



EPS

Escola Politècnica

UdG Superior

Projecte/Treball Fi de Carrera

Estudi: Eng. Tècn. Informàtica de Gestió. Pla 2001

Títol: CORRECTOR DE DIAGRAMES DE CLASSE PER L'ACME

Document: Memòria

Alumne: Jordi Oliveras Rovira

Director/Tutor: Josep Soler / Ferran Prados
Departament: Informàtica i Matemàtica Aplicada
Àrea: LSI

Convocatòria (mes/any): 06/2009

ÍNDEX

1.Introducció.....	6
1.1.Diagrames de classes. Conceptes bàsics.....	7
1.2.Disseny conceptual de bases de dades utilitzant diagrames de classe.....	11
1.3.Exemple.....	11
2.Objectius.....	13
3.Metodologia.....	15
4.Eines utilitzades.....	16
4.1.Llenguatges de programació.....	16
4.2.Entorns de desenvolupament.....	19
4.3.Altres eines software usades.....	20
5.L'editor de diagrames de classes.....	22
5.1.Requeriments de l'editor.....	22
5.2.Decisions prèvies al desenvolupament de l'editor.....	22
5.3.Diagrama de casos d'us.....	23
5.4.Fitxes de casos d'us.....	25
5.5.Diagrama de classes.....	28
5.5.1.Classes auxiliars.....	31
Classe IntAux.....	31
Classe XMLauxiliar.....	31

Classe Parellalmatges.....	33
Classe ImagePack.....	34
5.5.2.Classes del model de dades.....	35
Classe Element.....	35
Classe Classe.....	37
Classe AssociacioBase.....	41
Classe AssociacioDosClasses.....	42
Classe ClasseAssociacio.....	46
Classe ConjuntElements.....	48
Interfície Atribuible.....	52
Classe VectorAtributs.....	53
Classe Atribut.....	54
Classe InfoAssocaicio.....	55
5.5.3.Classes de la interfície.....	57
Classe AppletClasses.....	57
Classe Principal.....	59
Classe AbstractCanvas.....	66
Classe MyCanvas.....	67
Classe PetitCanvas.....	70
Classe Traduccio.....	71
Classe ConfigAssociacions.....	72
5.6.Configuració de les associacions.....	74

5.6.1.Llista d'associacions.....	74
5.6.2.Imatges.....	76
5.7.Representació d'un dibuix.....	77
5.7.1.Part comuna als diferents tipus.....	77
5.7.2.Llista de classes.....	78
Llista d'atributs.....	78
5.7.3.Llista d'associacions.....	79
5.7.4.Llista de classes associació.....	80
5.8.L'editor.....	81
5.9.Exemples d'us de l'editor.....	88
6.Definició d'un problema.....	91
6.1.Estructura d'un problema.....	91
6.2.Estructura d'una solució.....	92
6.2.1.Classe.....	93
6.2.2.Associació.....	94
6.2.3.Classe associació.....	96
6.3.Exemple d'un problema.....	97
6.4.Interpretació d'una solució.....	98
6.5.Errors en llegir un fitxer d'un problema.....	101
6.5.1.Errors en interpretar una solució.....	102
7.Corrector de diagrames de classes.....	105
7.1.Entrada i sortida del corrector.....	105

7.2. Correcció.....	105
7.2.1. Càlcul de la puntuació.....	106
7.2.2. Estratègia de correcció.....	107
7.2.3. Missatges d'error indicats pel corrector.....	109
8. Integració amb la plataforma ACME.....	111
8.1. Paràmetres que es poden passar a l'editor.....	114
8.2. Funcions per a interactuar amb l'editor des d'una pàgina web.....	115
9. Conclusions.....	117
10. Propostes de futures millores.....	118
11. Bibliografia / Webgrafia.....	119
12. Figures.....	120

1. INTRODUCCIÓ

En els últims anys han aparegut gran quantitat de plataformes de e-learning. L'objectiu de totes aquestes plataformes és facilitar l'aprenentatge i donar suport a un curs o matèria. Conscients d'això, la majoria d'Universitats, disposen d'aquest tipus de plataformes amb més o menys prestacions. La plataforma de e-learning ACME ha sigut desenvolupada per un grup de professors del departament de Informàtica i Matemàtica Aplicada de la Universitat de Girona amb la col·laboració de varis alumnes que hi han aportat els seus projectes final de carrera. La plataforma assigna problemes als alumnes i aquests envien les seves solucions que són corregides immediatament. La plataforma té com a característica principal la correcció de forma totalment automàtica de gran varietat de problemes no trivials.

Per una altra part, la matèria de bases de dades és una de les més importants en qualsevol Enginyeria Informàtica. Competències com el disseny conceptual i lògic d'una base de dades relacional o la resolució de consultes SQL o àlgebra relacional apareixen en tots els cursos de bases de dades. La importància d'un bon disseny és un dels punts clau en el desenvolupament de qualsevol aplicació informàtica.

Actualment la plataforma ACME permet fer varis tipus d'exercicis de l'àrea de base de dades, com són:

- Consultes SQL.
- Diagrames Entitat – Relació.
- Esquemes de bases de dades relacionals.
- Normalització de bases de dades.
- Consultes d'Àlgebra relacional.

L'objectiu principal d'aquest projecte és afegir un nou tipus d'exercici a la plataforma ACME. Aquest nou tipus d'exercici ha de permetre fer dissenys conceptuals de bases de dades mitjançant diagrames de classes.

1.1. Diagrames de classes. Conceptes bàsics

Dins del marc de l'anàlisi i disseny orientat a objectes, la notació UML ha esdevingut un estàndard per al modelat de sistemes de software.

Dins de UML s'utilitzen diferents tipus de diagrames. Entre aquest diagrames trobem els diagrames de classe que ens permeten descriure l'estructura d'un sistema.

En els diagrames de classe hi trobem diferents elements com:

- **Classe:** Una classe descriu un grup de objectes amb propietats comunes (atributs), comportament comú (operacions), relacions comunes amb altres objectes i la mateixa semàntica. Mitjançant UML una classe es representa com un rectangle dividit en tres parts que contenen:
 - En la part superior el nom de la classe.
 - En la part central els atributs de la classe.
 - En la part inferior les operacions de la classe.

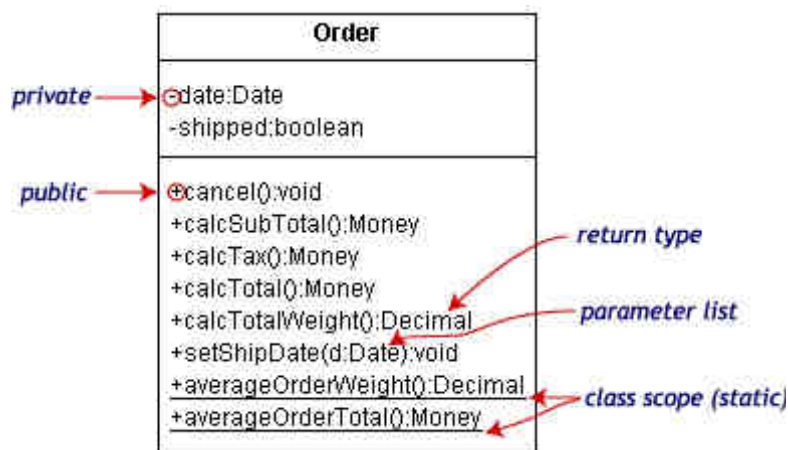


Figura 1.1: Exemple d'una classe.

- **Relació:** existeixen tres tipus diferents de relacions:
 - Associacions: connexions entre classes.
 - Dependències: relacions d'us.

- Especialitzacions/Generalitzacions: relacions d'herència.

- **Associació:** modela una connexió semàntica entre classes. És una relació estructural que especifica que els objectes d'una classe estan connectats als objectes de l'altre classe. Es pot navegar des d'un objecte d'una classe a un de l'altre. Es representa per una línia continua entre les classes.

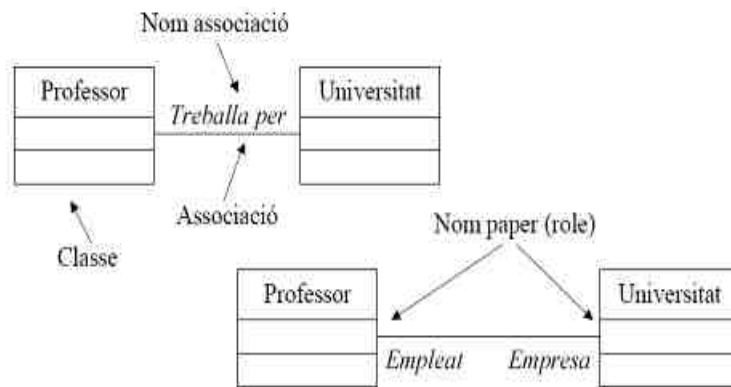


Figura 1.2: Exemple d'una associació.

- **Agregació:** és una forma especial d'associació que modela una relació “tot/part” o de “contenedor/contingut” entre un agregat (el tot) i les seves parts. Es representa mitjançant un rombe blanc en la part del tot.

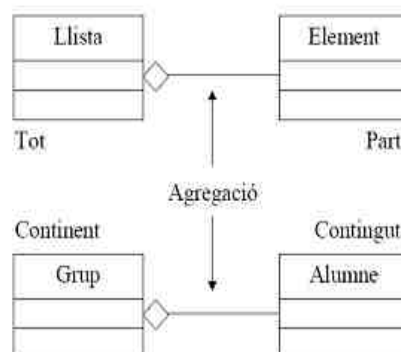


Figura 1.3: Exemple d'una agregació.

- **Composició:** és una forma d'agregació amb un propietari fort i temps de vida coincidents. Les parts no tenen sentit sense el tot. Les parts només es poden crear després de l'agregat (el tot) el qual pertanyen, però un cop creades viuen i moren amb l'agregat. Les parts només poden formar part d'un agregat.

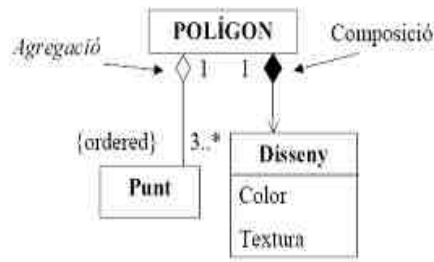


Figura 1.4: Exemple d'una composició.

- **Multiplicitat:** la multiplicitat defineix quants objectes participen en una relació, és a dir el número d'instàncies d'una classe relacionades amb una instància de l'altre classe. S'especifiquen mitjançant un rang en cada final d'associació i poden tenir els següents valors: 1..*, 1..1 (1), 0..1, 0..* (*), n..m.

- **Navegació:** les associacions són bidireccionals per defecte però a vegades és desitjable restringir la navegació en una sola direcció. En aquest cas a la línia de l'associació se li afegeix una fletxa que indica el sentit de la navegació.

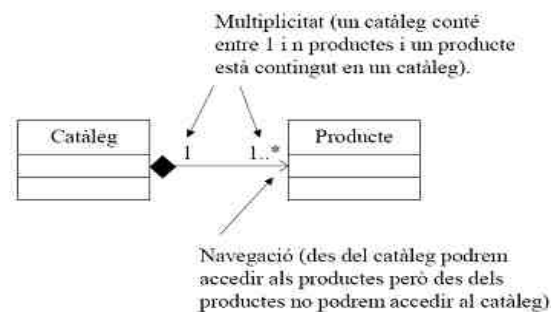


Figura 1.5: Exemple de multiplicitat i navegació.

- **Associació qualificada:** denota un atribut o llista d'atributs els quals el seu valor seleccionen un o varis objectes relacionats en la classe destí. Entre la classe i l'associació s'especifica l'atribut qualificador de l'associació.

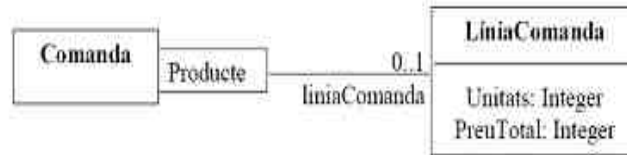


Figura 1.6: Exemple 1 d'una associació qualificada.

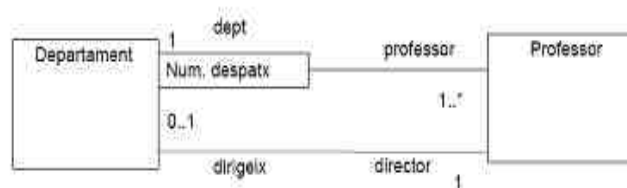


Figura 1.7: Exemple 2 d'una associació qualificada.

- **Classe associació:** una classe associació afegeix una restricció: només pot existir una instància de l'associació entre qualsevol parell d'objectes participants. Representa propietats que depenen dels dos objectes implicats en una associació.

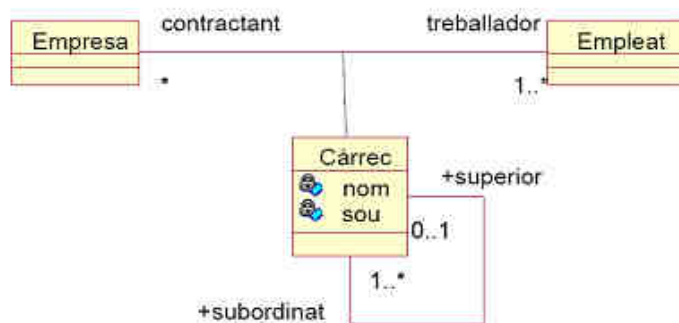


Figura 1.8: Exemple d'una classe associació.

- **Generalització:** és una relació entre classes on la classe especialitzada comparteix estructura i comportament amb la classe general.

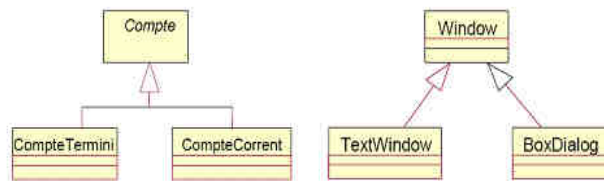


Figura 1.9: Exemple de generalització.

1.2. Disseny conceptual de bases de dades utilitzant diagrames de classe

Degut a l'auge de UML en l'anàlisi i programació orientada a objectes, cada vegada més s'estan utilitzant els diagrames de classe com a eina pel disseny conceptual de bases de dades.

Tot i que UML es va desenvolupar principalment per al disseny de software, una part molt important de d'aquest disseny consisteix en el disseny de bases de dades a les que es tindrà accés mitjançant mòduls de software. Per tant una part important d'aquestes metodologies, principalment els diagrames de classes són similars en molts aspectes als diagrames d'Entitat Relació Estès (EER), tot i que la terminologia sigui diferent. El concepte de "tipus d'entitat" del model Entitat Relació (ER) es representa com una classe i una "entitat ER" es correspon a un objecte en UML. Els "tipus de relacions" es denominen associacions o relacions en UML i com un cas particular d'aquestes es defineixen les agregacions. Les restriccions de cardinalitat i participació es passen a anomenar-se multiplicitats.

1.3. Exemple

A continuació un exemple d'us d'un diagrama de classes per en disseny conceptual de bases de dades. L'enunciat està en castellà:

- *La empresa está organizada en departamentos. De cada departamento nos interesa saber su número, el nombre y el teléfono. Cada departamento tiene asignados empleados de los que nos interesa saber su nombre completo, NIF,*

direcció completa, sueldo, sexo y fecha nacimiento. El trabajo de cada empleado está supervisado por otro empleado. Uno de los empleados del departamento desempeña la función de director del departamento.

- *Cada departamento controla un cierto número de proyectos. De cada proyecto nos interesa saber su código, su nombre y la ciudad donde se realiza. Cada proyecto es controlado por un único departamento.*
- *Un empleado puede trabajar en varios proyectos. Nos interesa saber el número de horas que cada empleado dedica a cada proyecto.*
- *Un empleado puede tener varias personas dependientes que figuran en su cartilla de la seguridad social. De cada una de estas personas nos interesa saber su número de orden dentro de la cartilla, su parentesco con el empleado, su nombre y su fecha de nacimiento.*

El sistema diseñado nos debe permitir:

- *A partir de un departamento saber todos los empleados que tiene asignados, saber quien es su director y los proyectos que controla.*
- *A partir de un empleado saber sus datos personales, los datos de su supervisor; en que proyectos ha trabajado y cuantas horas ha dedicado a cada proyecto. También nos interesa saber las personas dependiente que tiene a su cargo.*
- *A partir de un proyecto saber los datos de este, que departamento lo controla, los empleados que están trabajando en él y las horas dedicadas.*

A continuació un diagrama equivalent a l'enunciat anterior.

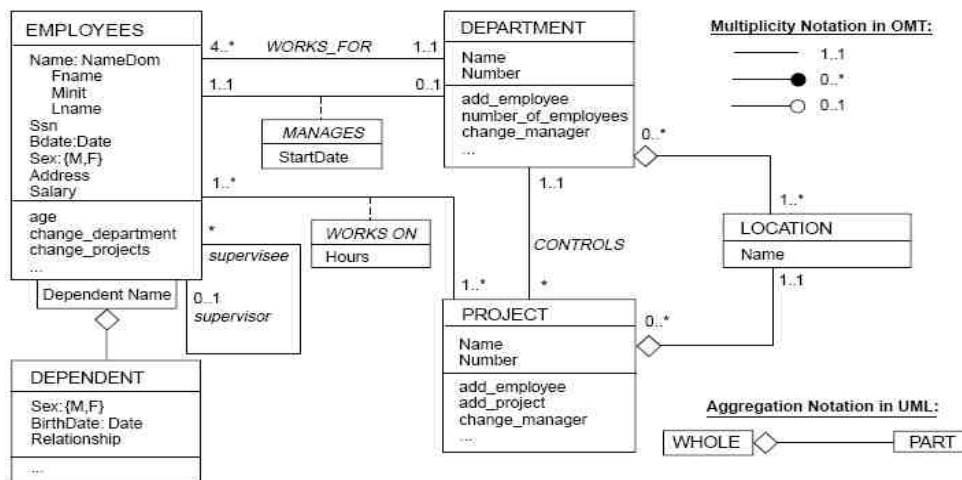


Figura 1.10: Exemple disseny conceptual de bases de dades.

2. OBJECTIUS

L'objectiu principal d'aquest Projecte Final de Carrera és desenvolupar l'anàlisi, disseny, implementació i integració dins la plataforma ACME d'un mòdul corrector de diagrames de classes, com a eina de treball en el disseny conceptual de bases de dades. Per complir aquest objectiu principal ha estat necessari:

- Crear un editor amb el que l'usuari pugui dibuixar diagrames de classes.
- Definir l'estructura d'un problema.
- Crear el corrector de diagrames de classes.
- Integrar l'editor i el corrector a l'ACME.
- Realitzar proves exhaustives.

Els objectius es poden resumir en el següent gràfic:

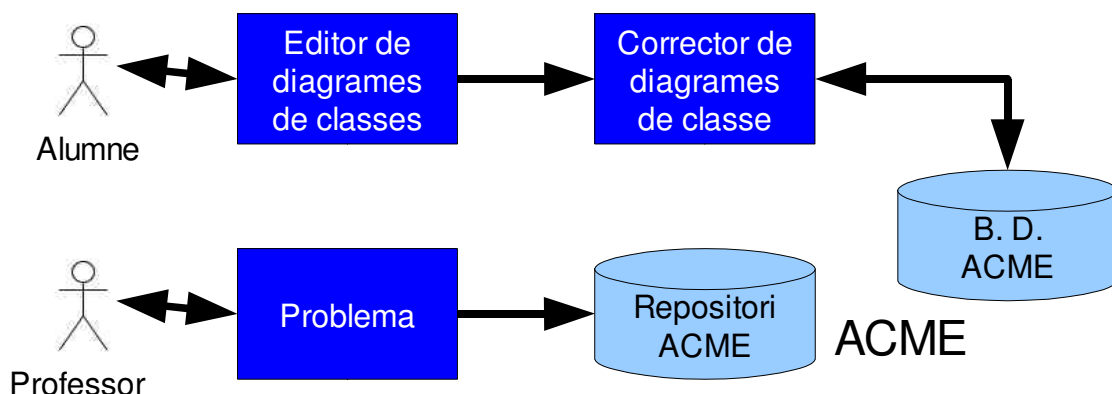


Figura 2.1: Esquema dels objectius.

El professor genera un problema amb un editor de text qualsevol seguint l'estructura definida en aquest Projecte (veure pàg. 91). Una vegada escrit el puja a l'ACME que comprova la seva correctesa segons el mòdul desenvolupat. Una vegada l'ACME detecte que és correcte l'incorpora al seu repositori i ja està apunt per ser assignat al quadern dels alumnes.

Quan un professor ha assignat un problema d'aquest tipus (diagrames de classes),

l'alumne visualitzarà l'editor desenvolupat (veure pàg. 22) i podrà dibuixar el corresponent diagrama de classes. Una vegada l'hagi acabat, l'enviarà a corregir. L'ACME a través del corrector desenvolupat en aquest Projecte (veure pàg. 105) comprovarà si la solució de l'alumne és equivalent a una de les solucions correctes del professor. Si és així donarà la solució per correcte i en cas contrari retornarà a l'alumne missatges d'ajuda per poder detectar els errors.

3. METODOLOGIA

Una metodologia és un seguit d'etapes i procediments utilitzats durant el procés de desenvolupament d'un sistema informàtic.

Les metodologies àgils s'agrupen en diferents tipus que permeten lliurar programari nou al final de cadascun dels blocs creats, aconseguint una aplicació final. Les taques que s'han de realitzar en cada bloc són: planificació, anàlisi, disseny, implementació, proves i documentació.

La metodologia àgil que és més apropiada per aquest projecte és l'anomenada Iterativa Incremental. La idea principal d'aquesta metodologia es desenvolupar un sistema de programes de manera incremental.

A continuació un esquema de com s'ha utilitzat la metodologia.

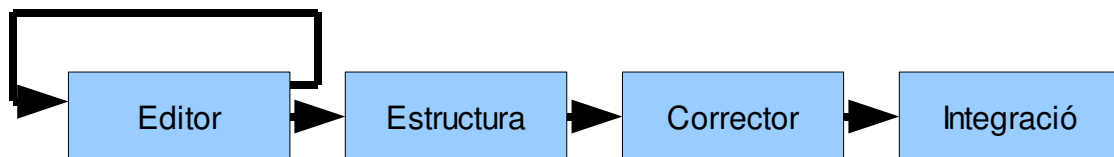


Figura 3.1: Esquema de les diferents etapes.

Primer de tot es va crear l'editor. Per a crear-lo es van anar fent varies iteracions on en cada una s'implementava una funcionalitat i es provava la funcionalitat i la seva integració amb la resta.

Quan l'editor ja quasi estava acabat, es va definir l'estructura d'una solució. Després de definir l'estructura d'una solució es va acabar l'editor. Una vegada acabat l'editor i l'estructura d'una solució es va definir l'estructura d'un problema.

Tot seguit es va implementar el corrector. Aquest també es va anar fent per parts i provant-lo.

I per acabar es va integrar en la plataforma ACME.

4. EINES UTILITZADES

Per dur a terme el projecte s'han d'utilitzar diferents eines per tal de poder desenvolupar-lo. Aquestes és poden agrupar en:

- *Llenguatges de programació*: aquestes eines s'han utilitzat en l'etapa d'implementació del projecte.
- *Entorns de desenvolupament*: aquests entorns són necessaris per tal d'agilitzar l'etapa d'implementació, ja que fan més còmode la tasca de programar.
- *Altres tipus d'eines*: per crear el document de la memòria, s'han utilitzat varis programes d'edició d'imatges,... etc.

4.1. Llenguatges de programació

Els llenguatges que s'han fet servir en aquest projecte són:

- **Java**: és un llenguatge de programació semi-interpretat, ja que al compilar es genera codi per a una màquina virtual que ha de ser interpretat. És un llenguatge orientat a objectes, robust i segur. Té parts que són codi obert i es pot usar des de la majoria de sistemes operatius sense necessitat de tornar a compilar.

Aquest llenguatge s'ha utilitzat per a crear l'editor de diagrames de classes.

- **Javadoc**: és el "llenguatge" que s'utilitza normalment per a documentar les classes escrites en Java. Està basat en comentaris de bloc: obertura `/**` en lloc de `/*` (inici normal d'un comentari multilínia de Java), cada línia que forma part de la documentació comença per `*` i el bloc s'acaba per `*/`. Utilitza una serie de paraules clau que comencen per `@` per indicar paràmetres, el retorn o crear enllaços. Permet crear enllaços a altres parts de la documentació.

Usat per a crear la documentació de les classes de l'editor i de les funcions del corrector.

- **PHP (*PHP Hypertext Pre-processor*)**: és un llenguatge de programació

interpretat. Va ser dissenyat originalment per la creació de pàgines web dinàmiques, es pot incrustar dins el codi *HTML*. El seu funcionament és molt senzill s'executa en un servidor web que contingui el intèrpret PHP, el que fa el servidor web és agafar el codi en PHP com l'entrada i l'interpreta, i com a resultat obtenim pàgines web.

Es pot utilitzar en la majoria dels sistemes operatius del mercat, incloent Linux, moltes varietats de Unix, Microsoft Windows, Mac OS X,..., igual que suporta la majoria de servidors web d'avui dia, incloent Apache, Internet Information Server,...; sense cost ja que és un llenguatge Open Source. D'aquesta manera, si programem amb aquest llenguatge tenim la llibertat de poder escollir el sistema operatiu i servidor web que més ens convingui.

També té la possibilitat d'utilitzar programació orientada a objectes o bé programació de procediments, tot i que no totes les característiques estàndards de la programació orientades a objectes estan implementades a la versió 4.0, però sí a la versió 5.0 (*L'ACME* utilitza la versió 4.0). Aquest llenguatge, a través de les múltiples llibreries que incorpora ens deixa manipular en temps real la creació i manipulació d'imatges, arxius PDF i pel·lícules Flash.

Els principals avantatges d'aquest llenguatge són els següents:

- És un llenguatge multiplataforma.
- Té capacitat de connexió amb la majoria de sistemes de gestió de base de dades que s'utilitzen a l'actualitat.
- Llegir i manipular dades de diverses fonts, incloent dades que poden introduir els usuaris als formularis *HTML*.
- Capacitat d'expandir el seu potencial amb mòduls o extensions.
- Trobarem una àmplia documentació d'aquest llenguatge a la web.
- Llenguatge Orientat a Objectes.

La plataforma *ACME*, està basada en aquest llenguatge en la seva versió 4.0, així que per fer aquest Projecte Final de Carrera és el que s'ha utilitzat per tal de

poder integrar el tipus de problema en la plataforma i per fer el corrector.

- **Javascript:** és un llenguatge de programació interpretat, és a dir, que no requereix que sigui compilat. S'utilitza principalment en pàgines web, amb una sintaxis molt semblant a la del llenguatge *Java*. Igual que *Java*, *Javascript* és un llenguatge orientat a objectes, ja que disposa de les eines mínimes per realitzar l'orientació a objectes com poden ser herència, encapsulació, ... S'utilitza en pàgines web *HTML*, per realitzar tasques des del client, de tal manera que realitzant validacions des de la part del client, estalviant així viatges al servidor. La majoria dels navegadors interpreten el codi *Javascript* integrat dins les pàgines web, tot i que a vegades ens podem trobar amb alguna dificultat.

S'utilitza per interactuar amb l'editor des de les pàgines web de l'ACME.

- **HTML (*Hyper Text Markup Language*):** és un llenguatge de marques molt senzill que s'utilitza per crear els textos i pàgines web. Aquest es basa en marques/etiquetes que defineixen estils per crear els hipertextos. Aquesta definició es deu a que està compost per etiquetes que defineixen l'estructura i el format del document que l'usuari visualitzarà a través de la web. El navegador s'encarrega de llegir les etiquetes i interpretar-les. En quan a la creació d'arxius *HTML*, són fitxers plans de tal manera que amb un editor senzill és poden crear.

S'ha utilitzat en la integració de l'editor de diagrames de classes amb l'ACME.

- **XML:** és un altre llenguatge de marques més estricte que l'html usat per intercanviar informació de forma senzilla. De fet XML és un conjunt de regles per a definir, juntament amb un DTD o un XML schema, l'estructura d'un document. Un DTD o un XML schema és un document que descriu l'estructura d'un arxiu XML. La principal diferència entre DTD i XML schema és que el segon és de fet un document XML.

S'ha utilitzat per a crear l'arxiu de configuració de les associacions i per a guardar i recuperar un dibuix generat per l'editor. També s'ha usat per a crear els XML schema dels fitxers de configuració i de l'estructura per a guardar i recuperar un dibuix.

4.2. Entorns de desenvolupament

Els entorns de desenvolupament usats en el projecte són:

- **NetBeans IDE 6.5:** és un entorn de desenvolupament integrat per a programar Java i altres llenguatges que, entre altres coses, permet crear interfícies gràfiques de forma senzilla. L'entorn és ampliable mitjançant plugins.

S'ha utilitzat per crear la interfície gràfica de l'editor de diagrames de classes. També s'ha usat el plugin de UML del NetBeans per generar els gràfics UML d'aquest document.

- **Eclipse 3.4:** és un altre entorn de desenvolupament integrat per a programar Java. A diferència de NetBeans no és tan senzill crear interfícies gràfiques, però té un millor editor i debugador (software que permet interrompre l'execució d'un programa per veure què està passant en un moment concret).

S'ha utilitzat per a programar i provar l'editor de diagrames de classes. Ja que tot i que el NetBeans també permet programar en Java, no permet provar els applets de Java. També s'ha utilitzat el plugin de XML de l'Eclipse per dissenyar i escriure els arxius en XML de la configuració i el disseny de l'estructura d'un dibuix.

- **PHP 4.3/4.4 des de consola:** l'interpret de PHP és un executable que funciona en mode consola. Quan s'utilitza des d'un servidor web, com per exemple Apache, aquest crea un pipe on li passa l'entrada al intèrpret i aquest retorna la sortida al servidor web.

S'ha utilitzat l'executable de PHP directament des de consola per a provar les funcions del corrector.

- **Navegador web:** un navegador web és una aplicació software que permet a l'usuari recuperar i visualitzar documents d'hipertext (documents *HTML*). Aquests són necessaris per visualitzar i provar el nou tipus de problema un cop integrat en l'ACME. Avui en dia existeixen diversos navegadors: Internet Explorer, Mozilla Firefox, Opera, Safari, Chrome,...etc. Cadascun d'ells té una interpretació pròpia del llenguatge *HTML* i del *Javascript* és per això que a l'hora de fer les proves de

visualització dels nous mòduls s'han triat els dos més utilitzats. En aquest projecte s'han utilitzat els navegadors:

- Internet Explorer 7.0
- Mozilla Firefox 3.0
- **SSH Secure Shell 3.2.9** : SSH és un protocol segur de comunicació a través de la xarxa. SSH Secure Shell és un conjunt d'eines que van ser dissenyades des d'un inici per oferir una seguretat màxima i permetre l'accés remot als servidors de manera segura utilitzant el protocol SSH. Per realitzar les transferències de fitxers al servidor, s'ha utilitzat aquest software que ens permetia connectar-nos al servidor de l'ACME a través de SSH.
- **Notepad++ 5.3.1**: és una eina per editar, ens permet editar varis arxius a la vegada de mida il·limitada. Aquest editor resalta el text per tal que l'edició sigui més còmoda pel programador. També permet afegir-li plugins (afegits) per estendre'n les funcionalitats com per exemple mostrar una llista de les funcions que conté l'arxiu que s'està editant. S'ha utilitzat aquest editor per redactar els fitxers de PHP.

4.3. Altres eines software usades

A més de les eines indicades a l'apartat anterior, s'ha usat altre software:

- **Microsoft Paint**: és una eina dibuix i edició d'imatge digitals integrada en el Windows. Té suport per varis tipus de formats de imatge. Usat per crear les imatges simples que s'utilitzen en l'editor de diagrames.
- **Adobe Photoshop CS2**: és una eina d'edició d'imatges digitals amb suport per a múltiples formats de imatge. Conté eines de selecció, filtres, color, ... etc. Usat en crear les imatges dels botons i per eliminar els fons de les imatges creades amb el Microsoft Paint.
- **Javadoc**: és una eina inclosa en l'entorn de desenvolupament de Java (SDK) que permet generar documentació en HTML a partir de la documentació escrita en

Javadoc. Només funciona si el codi font usat per generar la documentació és Java. S'ha utilitzat des de eclipse per generar la documentació de les classes de l'editor.

- **Doxygen:** és una eina de codi obert que permet generar documentació en HTML i altres tipus de sortida a partir de Javadoc. Si es configura per tal que usi el conjunt d'eines Graphviz (un conjunt d'eines per a crear grafs), permet incloure dibuixos de les herències de les classes i de les crides de funcions. A més de Java, també suporta la generació de documentació a partir de PHP, C++, C# i altres llenguatges.

S'ha utilitzat per generar la documentació de les classes de l'editor i de les funcions del corrector.

5. L'EDITOR DE DIAGRAMES DE CLASSES

L'editor de diagrames de classes és una eina feta amb Java que ha de servir per a dibuixar diagrames de classes.

Es va escollir Java com a llenguatge de programació perquè a l'ACME ja hi ha altres editor fets en Java que es podien usar de base, com són l'editor de diagrames Entitat – Relació i l'editor de Circuits Elèctrics. A més a més ja tenia coneixement de Java i de creació d'applets en Java.

5.1. *Requeriments de l'editor*

Abans de començar el desenvolupament de l'editor, ens vam reunir per a desidir que havia de fer.

- Permetre dibuixar diagrames de classes fàcilment, amb els elements de classes, associacions, agregacions, composicions i classes associació.
- Poder guardar un dibuix fet per l'editor i poder restaurar-lo.
- Que es pugui usar el mateix editor per a visualitzar solucions enviades per l'alumne.
- Que sigui el màxim fàcil possible afegir nous tipus de dibuixos, sobretot d'associacions.
- Facilitar el màxim la feina d'una possible persona que en el futur hagi de modificar i/o ampliar les funcionalitats l'editor.
- Fer el màxim de fàcil l'ús de l'editor.

5.2. *Decisions prèvies al desenvolupament de l'editor*

Ja que els diagrames de classes generats anaven encarats a definir bases de dades, es van prendre les següents decisions:

- No es permeten associacions n-àries.
- Una classe pot tenir un màxim de deu relacions.
- Una classe s'identifica per un o més atributs clau.
- No es permeten interfícies.

5.3. Diagrama de casos d'us

Els diagrames de casos d'us s'utilitzen per a mostrar gràficament les funcionalitats d'un programa, en aquest cas l'editor de diagrames de classes.

En el diagrama es pot veure com l'usuari fonamentalment només pot editar un diagrama. Per altre part, el servidor ACME pot enviar o recuperar un diagrama.

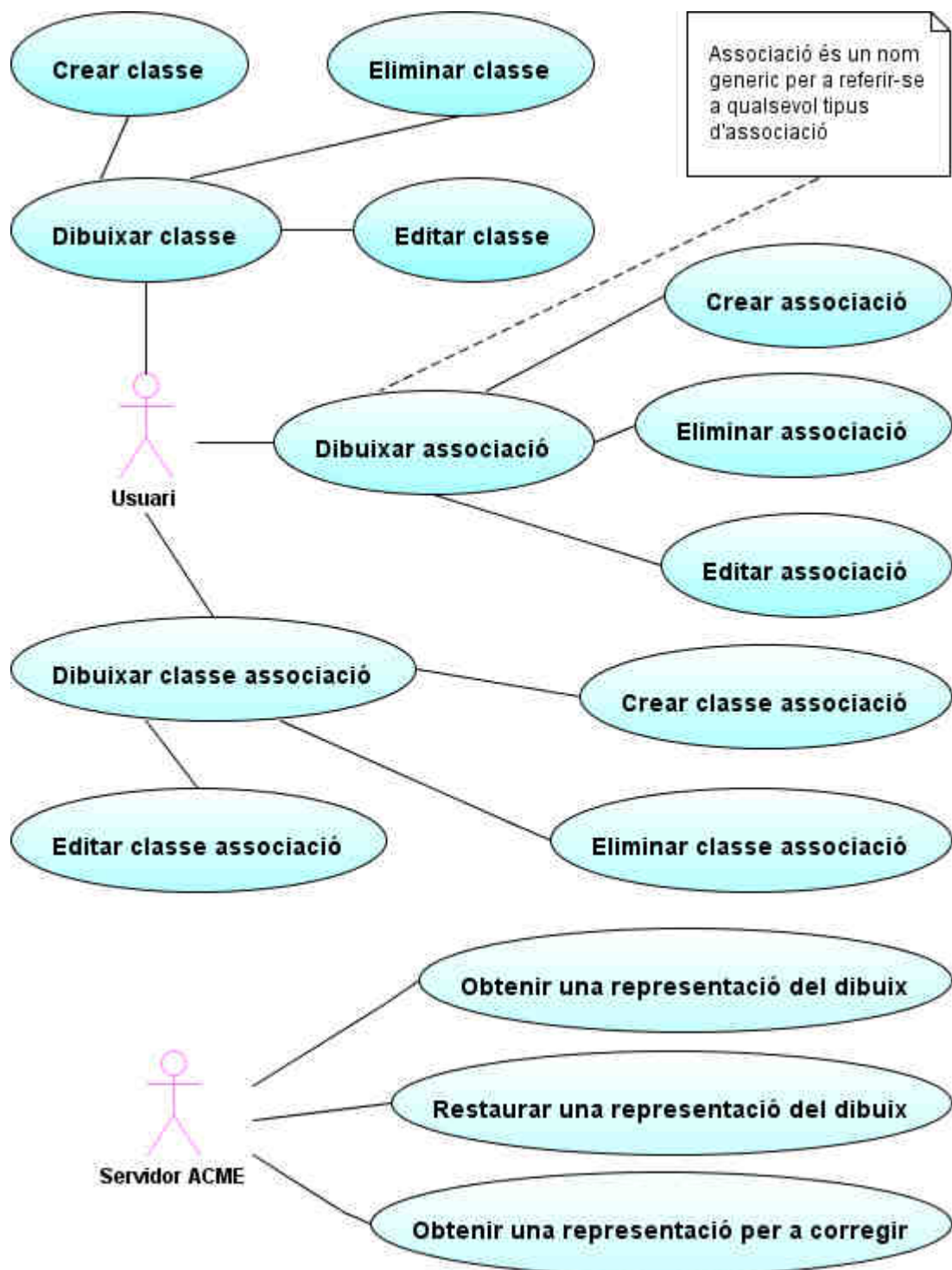


Figura 5.1: Diagrama de casos d'us de l'editor

5.4. Fitxes de casos d'us

Una fitxa de cas d'us és una explicació més acurada del que ha de fer un cas d'us. A continuació hi ha les fitxes de casos d'ús del diagrama anterior.

Totes les precondicions no indicades són certes. Les postcondicions, són que s'ha fet l'acció indicada. Els casos d'us que l'actor és l'usuari, poden ser cancel·lats a mitja acció.

Nom:	<u>Crear una classe</u>
Actor:	Usuari
Flux principal:	<ol style="list-style-type: none"> 1. Seleccionar botó de creació d'una classe. 2. Seleccionar una zona on no hi hagi cap classe en la zona de dibuix.
Flux alternatiu:	

Nom:	<u>Eliminar una classe</u>
Actor:	Usuari
Flux principal:	<p>Primera forma:</p> <ol style="list-style-type: none"> 1. Seleccionar la classe a eliminar. 2. Seleccionar el botó d'eliminar. <p>Segona forma:</p> <ol style="list-style-type: none"> 1. Seleccionar el botó d'eliminar. 2. Seleccionar la classe a eliminar.
Flux alternatiu:	

Nom:	<u>Editar una classe</u>
Actor:	Usuari
Flux principal:	<ol style="list-style-type: none"> 1. Seleccionar la classe a editar. 2. Fer alguna d'aquestes accions: <ul style="list-style-type: none"> • Modificar el nom. • Modificar els atributs.

Flux alternatiu:	<p>Modificar el nom.</p> <ol style="list-style-type: none"> 1. Canviar el nom. 2. Seleccionar el botó de modificar el nom. <p>Modificar els atributs.</p> <ol style="list-style-type: none"> 1. Modificar els atributs.
-------------------------	--

Nom:	<u>Crear una associació</u>
Actor:	Usuari
Flux principal:	<p>Primera forma:</p> <ol style="list-style-type: none"> 1. Seleccionar un botó de creació d'una associació. 2. Seleccionar una classe. 3. Seleccionar una altra classe (pot ser la mateixa que l'anterior). <p>Segona forma:</p> <ol style="list-style-type: none"> 1. Anar a la pestanya d'edició d'una associació. 2. Escollir les dades de l'associació. 3. Seleccionar el botó de crear una nova associació.
Flux alternatiu:	

Nom:	<u>Eliminar una associació</u>
Actor:	Usuari
Flux principal:	<p>Primera forma:</p> <ol style="list-style-type: none"> 1. Seleccionar l'associació a eliminar. 2. Seleccionar el botó d'eliminar. <p>Segona forma:</p> <ol style="list-style-type: none"> 1. Seleccionar el botó d'eliminar. 2. Seleccionar l'associació a eliminar.
Flux alternatiu:	

Nom:	<u>Editar una associació</u>
Actor:	Usuari
Flux principal:	<ol style="list-style-type: none"> 1. Seleccionar una associació. 2. Editar les seves propietats. 3. Seleccionar el botó de modificar una associació.

Flux alternatiu:	
-------------------------	--

Nom:	<u>Crear una classe associació</u>
Actor:	Usuari
Flux principal:	<p>Primera forma:</p> <ol style="list-style-type: none"> 1. Seleccionar el botó de creació d'una classe associació. 2. Seleccionar una associació o una classe. 3. Seleccionar l'altre element (una classe si abans s'ha seleccionat una associació, o una associació si abans s'ha seleccionat una classe). <p>Segona forma:</p> <ol style="list-style-type: none"> 1. Anar a la pestanya d'edició d'una classe associació. 2. Escollir les dades de la classe associació. 3. Seleccionar el botó de crear una nova classe associació.
Flux alternatiu:	

Nom:	<u>Eliminar una classe associació</u>
Actor:	Usuari
Flux principal:	<p>Primera forma:</p> <ol style="list-style-type: none"> 1. Seleccionar la classe associació a eliminar. 2. Seleccionar el botó d'eliminar. <p>Segona forma:</p> <ol style="list-style-type: none"> 1. Seleccionar el botó d'eliminar. 2. Seleccionar la classe associació a eliminar.
Flux alternatiu:	

Nom:	<u>Editar una classe associació</u>
Actor:	Usuari
Flux principal:	<ol style="list-style-type: none"> 1. Seleccionar una associació. 2. Editar les seves propietats. 3. Seleccionar el botó de modificar una associació.
Flux alternatiu:	

Nom:	<u>Obtenir una representació del dibuix</u>
Actor:	Servidor ACME

Flux principal:	Per cada element del dibuix obtenir la seva representació i ajuntar-les en una única cadena de caràcters.
Flux alternatiu:	

Nom:	<u>Restaurar una representació del dibuix</u>
Actor:	Servidor ACME
Precondicio:	La cadena de caràcters de la representació és correcta o està buida.
Flux principal:	Llegir cada element de la cadena i restaurar-lo.
Flux alternatiu:	Si hi ha algun error greu descartar l'element.

Nom:	<u>Obtenir una representació per a corregir</u>
Actor:	Servidor ACME
Flux principal:	Per cada element del dibuix obtenir la seva representació de correcció i ajuntar-les en una única cadena de caràcters.
Flux alternatiu:	

5.5. Diagrama de classes

Per tal de poder desenvolupar l'editor seguint les especificacions descrites en el diagrama de casos d'us, s'han creat les classes necessàries.

Per tal de fer-lo més llegible, el diagrama de classes següent no té totes les relacions entre les classes, falten dependències entre classes. Tampoc es mostren les classes privades que tenen algunes de les classes mostrades.

Les classes IntAux i XMLauxiliar estan repetides en el diagrama per a facilitar-ne la interpretació.

Els datatypes (Canvas, JPanel i JApplet) corresponen a classes que formen part de Java.

Les classes del diagrama es poden separar en dos grups principals. Els dos grups principals són: classes que formen la interfície gràfica de l'editor i classes que formen el model de dades usat en l'editor. El primer grup està compost per les classes

AbstractCanvas, PetitCanvas, MyCanvas, Principal, Traduccio i ConfigAssociacions. El segon grup està compost per les classes Element, Classe, AssociacioBase, AssociacioDosClasses, ClasseAssociacio, ConjuntElements, Atribut, VectorAtributs, InfoAssociacio i la interfície Atribuible. A més a més d'aquests dos grups principals, hi ha classes que són auxiliars entre els dos grups: Parellalmatges, ImagePack, IntAux i XMLauxiliar.

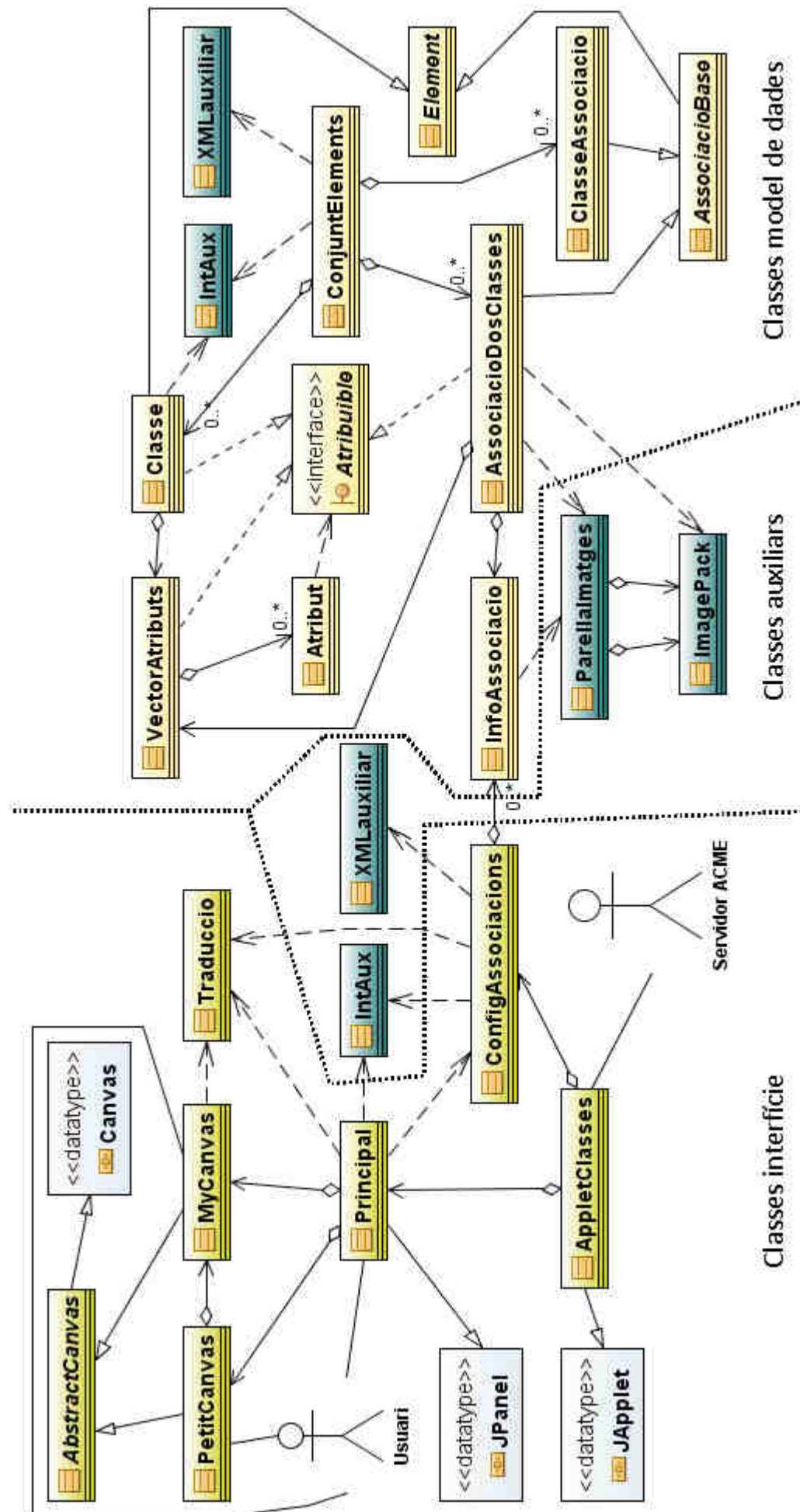


Figura 5.2: Diagrama de classes parcial de l'editor (falten dependències). Les línia puntejades separen els diferents grups de classes.

5.5.1. Classes auxiliars

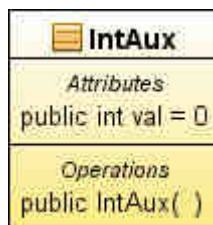
Les classes auxiliars són classes que s'han definit per a facilitar altres classes i per no repetir codi.

A continuació hi ha l'explicació de les classes:

- IntAux
- XMLauxiliar
- ParellaImatges
- ImagePack

Classe IntAux

Classe usada en els mètodes que necessiten retornar dos valor (mínim d'un d'ells enter), com per exemple en cercar connexions a usar en una Classe.



*Figura 5.3:
Classe IntAux.*

Classe XMLauxiliar

Classe no instanciable (els mètodes són estàtics, és a dir no necessiten una instància per funcionar) amb mètodes usats en llegir des d'un fitxer XML.

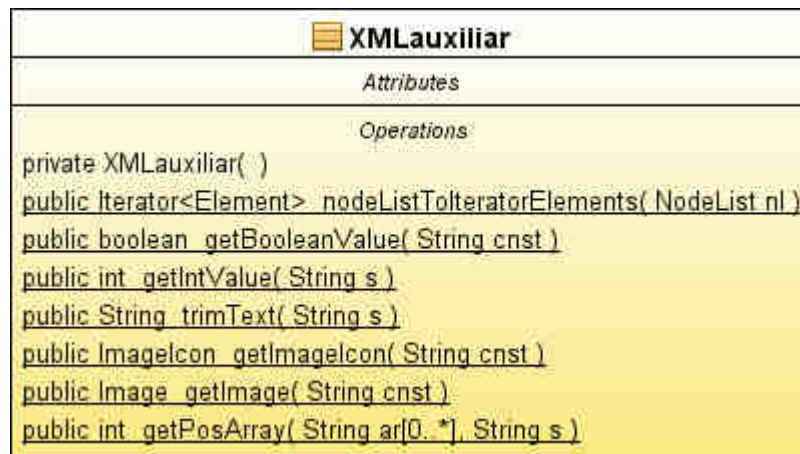


Figura 5.4: Classe XMLauxiliar.

Mètodes

Els mètodes de la classe són:

- **nodeListToIteratorElements:** a partir d'un objecte org.w3c.dom.NodeList retorna un iterador de org.w3c.dom.Element eliminant els objectes que no siguin de tipus org.w3c.dom.Element del paràmetre.
- **getBooleanValue:** a partir d'una cadena de caràcters retorna un valor booleà. Es considerarà cert si el text val "true" (sense tenir en compte majúscules i minúscules).
- **getIntValue:** a partir d'una cadena de caràcters retorna el número enter que conté. Si es produeix un error en interpretar la cadena retorna zero.
- **trimText:** elimina els espais, tabulacions, salts de línia i retorns de carro de l'inici i fi del paràmetre i el retorna. Si el paràmetre és nul o hi ha un error, retorna nul.
- **getImageIcon:** a partir de la direcció d'una imatge del paràmetre retorna la imatge com un objecte ImageIcon (una de les imatges usades en el botons de la interfície gràfica de l'editor).
- **getImage:** a partir de la direcció d'una imatge del paràmetre retorna la imatge com un objecte Image (una de les imatges usades en dibuixar).

- **getPosArray**: a partir d'un vector de cadenes de caràcters retorna la posició on es troba la cadena indicada (o -1 en cas de no trobar-la).

Classe Parellalimatges

Classe que controla les imatges de les associacions. Una d'elles és la imatge que s'utilitza quan l'associació està seleccionada i l'altre la imatge normal.

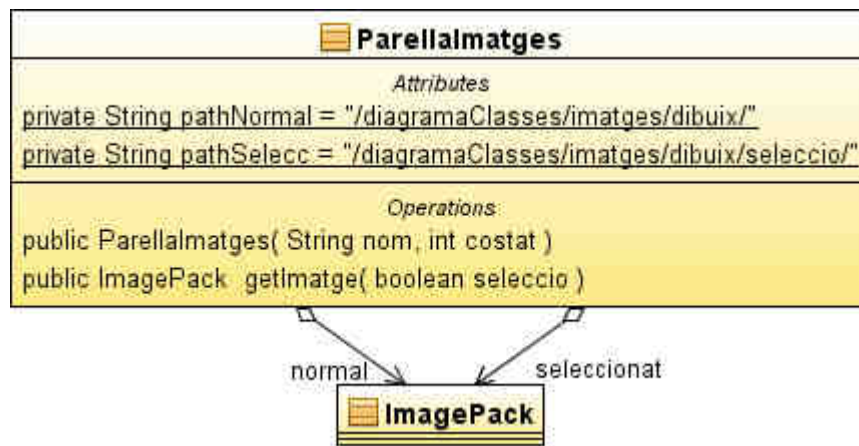


Figura 5.5: Dependències de la classe ParellaImatges.

Atributs

Els atributs de la classe són:

- **normal**: objecte de tipus ImagePack amb la imatge que s'utilitza quan l'associació no està seleccionada.
- **seleccionat**: objecte de tipus ImagePack amb la imatge que s'utilitza quan l'associació està seleccionada.

Mètodes

Mètode de la classe:

- **getImage**: retorna l'atribut "normal" si el paràmetre és fals, o l'atribut "seleccionat" si el paràmetre és cert.

Classe ImagePack

Classe per a facilitar els càlculs a l'hora de dibuixar les línies que formen una associació.



Figura 5.6: Classe ImagePack.

5.5.2. Classes del model de dades

A continuació hi ha l'explicació de les classes que formen el model de dades:

- Element
- Classe
- AssociacioBase
- AssociacioDosClasses
- Associacio
- ConjuntElements
- Atribuible
- VectorAtributs
- Atribut
- InfoAssocaicio

Classe Element

Classe abstracte de la qual deriven tots els objectes que es poden dibuixar. Conté la definició dels atributs bàsics i de les operacions comunes a tots els objectes que es poden dibuixar.



Figura 5.7: Classe abstracte Element

Atributs

Els atributs de la classe són:

- **CODI_NO_VALID**: constant que indica que el codi d'element no és vàlid.
- **codi**: identificador numèric de l'objecte. Usat per a controlar els objectes en

l'editor. Només pot ser assignat en crear-se un objecte derivat d'Element si aquest crida el constructor d'Element.

- **nom**: nom de l'objecte, per exemple en una classe, serà el nom de la classe. També és un identificador, és a dir, que en un mateix dibuix, no es pot repetir. És usat com a identificar en una solució.
- **puntBase**: punt superior esquerra en coordenades de dibuix on comença el rectangle que es pot usar en seleccionar un objecte des de la zona de dibuix.
- **Amplada i altura**: amplada i altura del rectangle on hi ha la zona de selecció en la zona de dibuix.

Mètodes

Els mètodes més importants de la classe són:

- **estaDintre**: comprova si el punt (en coordenades de dibuix) passat per paràmetre està dintre l'Element.
- **remove**: fa les accions necessàries abans ser eliminat d'un objecte ConjuntElements.
- **dibuixar**: dibuixa l'element sobre l'objecte del primer paràmetre i amb el zoom indicat al segon. El tercer paràmetre indica si cal recalcular les mides de l'objecte i l'últim si l'Element està seleccionat.
- **getCadenaCorreccio**: retorna la part de cadena de correcció corresponent a l'Element.
- **getCadenaDibuix**: retorna la part de cadena de dibuix corresponent a l'Element.

Classe Classe

Classe que implementa la representació d'una classe de UML. Pot tenir un màxim de deu associacions (dues a la part superior i inferior i tres a dreta i esquerra) o una sola

classe associació. També manté els seus atributs.

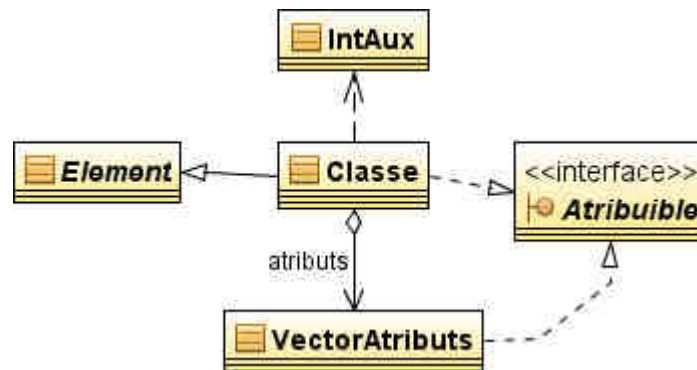


Figura 5.8: Dependències de la classe Classe.

Atributs

Els atributs més importants de la classe són:

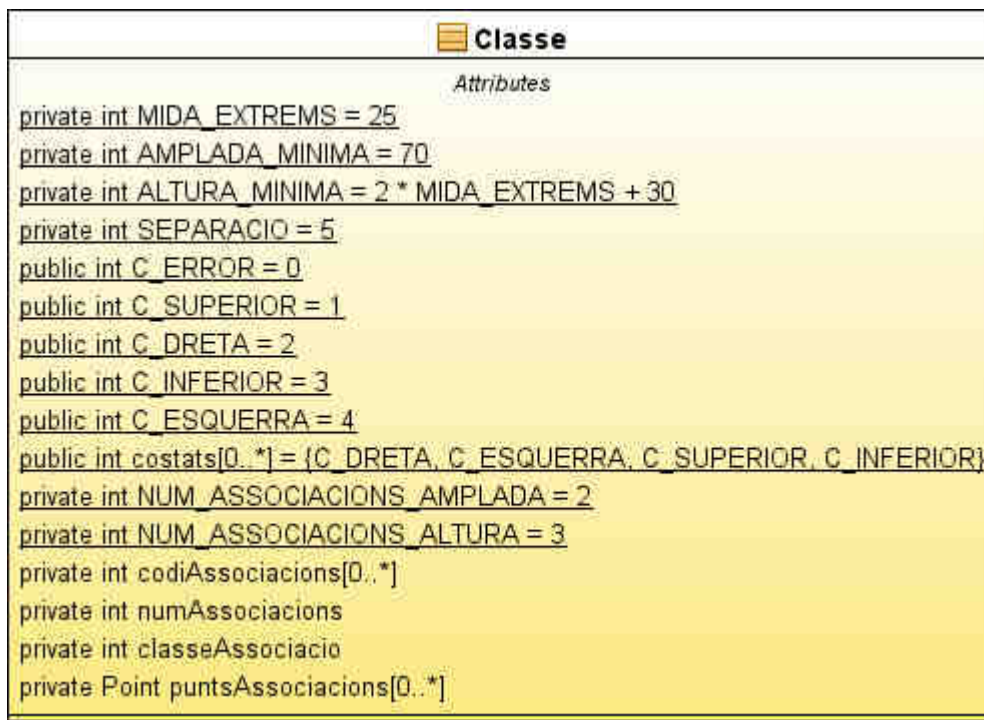


Figura 5.9: Atributs de la classe Classe

- **atributs:** objecte VectorAtributs que conté els atributs de la classe.
- **codiAssociacions:** vector que conté els codis de les associacions que estan connectades a la classe. Les posicions comencen a la connexió de més a l'esquerra de la part superior i segueixen el sentit de les agulles del rellotge. Si en

alguna posició no hi ha cap associació, el seu valor és la constant `CODI_NO_VALID` definida a `Element`.

- **puntsAssociaions**: vector amb els punts (en coordenades de dibuix) on es troben les connexions de les associacions de la mateixa posició de `codiAssociaions`.
- **numAssociaions**: número d'associacions que hi ha en la classe.
- **classeAssociaio**: posició del vector `codiAssociaions` on hi ha una classe associació o -1 en cas que no en tingui cap.
- **C_ERROR, C_SUPERIOR, C_DRETA, C_INFERIOR i C_ESQUERRA**: constant que indica a quin costat de la classe està la connexió demanada.
- **NUM_ASSOCIACIONS_AMPLADA i NUM_ASSOCIACIONS_ALTURA**: constants que indiquen el número d'associacions de la part superior i inferior i de la dreta i esquerra respectivament.

Mètodes

A la pàgina següent hi ha un esquema de tots els mètodes de la classe. Els mètodes més importants de la classe són:

- **buscarConnexions(Classe, IntAux)**: cerca quines connexions són les millors per a connectar dues classes (la classe sobre la que es crida el mètode i la classe del paràmetre) amb una associació. Si la classe sobre la que es crida el mètode i la classe del paràmetre són la mateixa, crida a `buscarConnexions(IntAux)`. Retorna la connexió sobre la que connectar l'associació a la classe sobre la que es crida el mètode i a l'objecte `IntAux` guarda la millor connexió a usar de la classe del paràmetre. Retorna -1 en cas que no es pugui trobar una parella de connexions. El seu funcionament, es el següent: comprova totes les parelles possibles de connexions cercant la parella que està a mínima distància (no té en compte les connexions usades).

 Classe
<i>Operations</i>
<pre> public Classe() public Classe(int codi, String nom) private void partComunaConstructors() public Atribut[0..*] getAtributs() public void addAtribut(Atribut a) public Atribut getAtribut(int i) public int getNumAtributs() public void removeAtribut(int i) public void removeAtribut(Atribut a) public int buscarConnexio(Classe c, IntAux ia) public int buscarConnexio(Point p) public int buscarConnexio(IntAux i) private int buscaConnexio(int posIni, int posFi, IntAux i) public int costatConnexio(int connexio) <u>public boolean connexionsEnVertex(int cos1, int cos2)</u> <u>public boolean costatValid(int costat)</u> public Point puntConnexio(int connexio) public void setAssociacio(AssociacioBase ab, int connexio) public void removeAssociacio(int adc, int connexio) public boolean teAssociacions() public boolean teClasseAssociacio() public Iterable<Integer> getAssociacions() <u>public int getAmpladaDefecte()</u> <u>public int getAlturaDefecte()</u> public double arealInterseccio(Classe c) public double arealInterseccio(int x, int y, int amplada, int altura) private void recalculaMides(Graphics2D g, float zoom, boolean seleccionat) <u>public void dibuixar(int x, int y, Graphics2D g, String nom, boolean interseccia)</u> <u>private void dibuixar(int x, int y, int amplada, int altura, String nom, Graphics2D g)</u> private String getCadenaCorreccioAtributs(boolean esID) </pre>
<i>Operations Redefined From Element</i>
<pre> public void remove() public void dibuixar(Graphics2D g, float zoom, boolean esCanvasGran, boolean seleccionat) public String getCadenaCorreccio() public String getCadenaDibuix() </pre>

Figura 5.10: Mètodes de la classe Classe

- **buscarConnexions(IntAux):** cerca un parell de punts de connexió de la classe per a connectar una associació en la classe. Retorna una connexió i l'altre es guarda a l'objecte del paràmetre. Si no es pot trobar cap parella de connexions, retorna -1. El seu funcionament és el següent:


```

Cerca les connexions en els vertex.
Si no s'han trobat.
    Cerca les connexions en un costat.
    Si no s'han trobat.
        Cerca les connexions a qualsevol lloc de la classe.
    Fsi
Fsi
    
```

- **buscarConnexions(Point)**: cerca la connexió més pròxima al punt del paràmetre. Usat en cercar la connexió en una classe associació.
- **remove**: mètode cridat des de la classe ConjuntElements quan s'elimina un element del conjunt. Indica que també s'eliminïn les associacions o classe associació que pot tenir la Classe.
- **dibuixar(Graphics2D, float, boolean, boolean)**: dibuixa la classe sobre l'objecte Graphics2D passat. Si el tercer paràmetre és cert, es recalculen les mides de la Classe. I si el tercer i quart paràmetres són cert, es dibuixa la classe com a seleccionada.

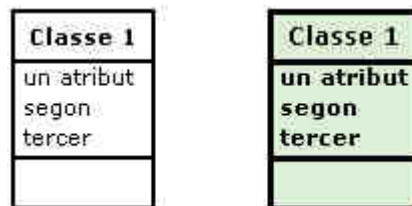


Figura 5.11: Exemple d'una classe sense seleccionar i la mateixa classe seleccionada.

Classe AssociacioBase

Classe abstracte amb operacions comunes a AssociacioDosClasses i ClasseAssociacio.

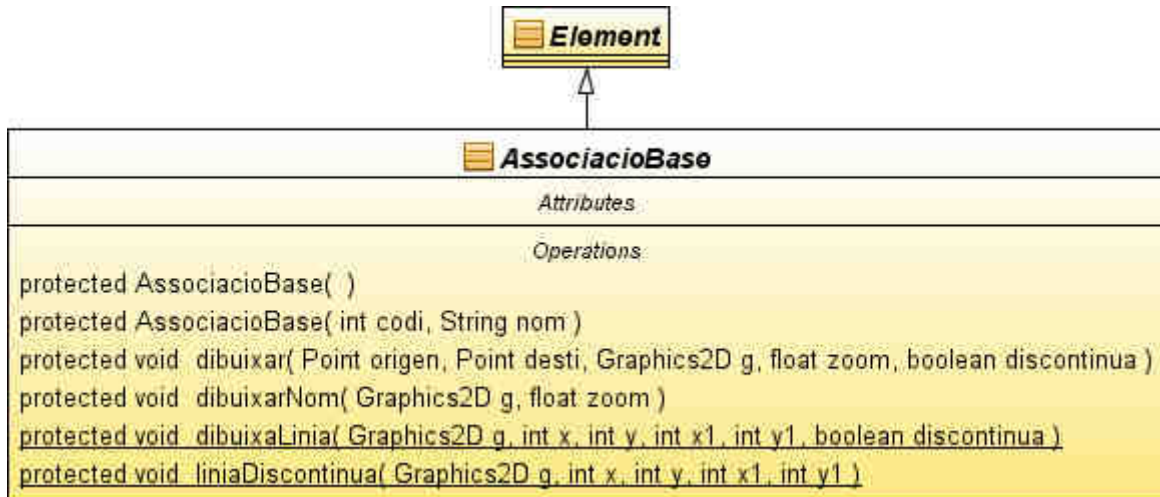


Figura 5.12: Herència de la classe abstracte *AssociacioBase*.

Mètodes:

Els mètodes de la classe són:

- **dibuixar**: dibuixa una línia de l'associació entre els punts indicats sobre l'objecte `Graphics2D` passat. També dibuixa el nom a mitja distància de la línia.
- **dibuixarNom**: dibuixa el nom de l'associació al punt `puntBase` heretat de la classe `Element`. Cal modificar-lo abans de cridar al mètode.
- **dibuixarLinia**: dibuixa una línia entre els punts indicats sobre l'objecte `Graphics2D` passats tinguen en compte el paràmetre `discontinua`.
- **liniaDiscontinua**: dibuixa una línia recte discontinua entre els punts indicats sobre l'objecte `Graphics2D` passat.

Classe AssociacioDosClasses

Classe que implementa la representació d'una relació genèrica de UML. Aquesta classe representa una associació entre dues classes qualsevol, el tipus concret d'associació ve determinat per l'atribut `infoAss`.

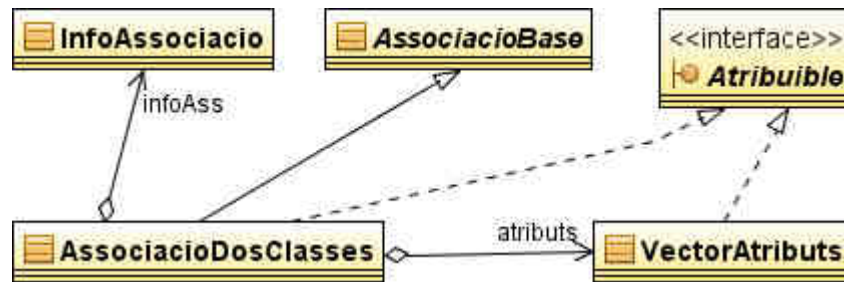


Figura 5.13: Dependències de la classe AssociacioDosClasses.

Les associacions per a distingir les dues classes, es segueix el següent conveni: es distingeix la primera classe de la segona classe, siguen aquesta última la que en el dibuix conté el dibuix que representa l'associació (les fletxes de navegació no compte):

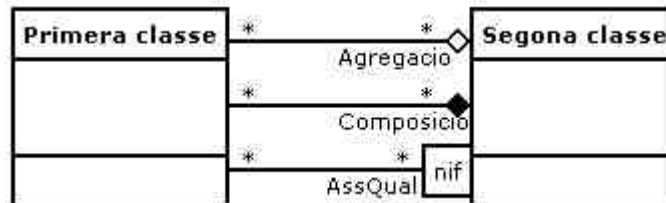


Figura 5.14: Exemple per il·lustrar la diferència entre primera i segona classe en una associació.

Atributs

Els atributs més importants de la classe són:

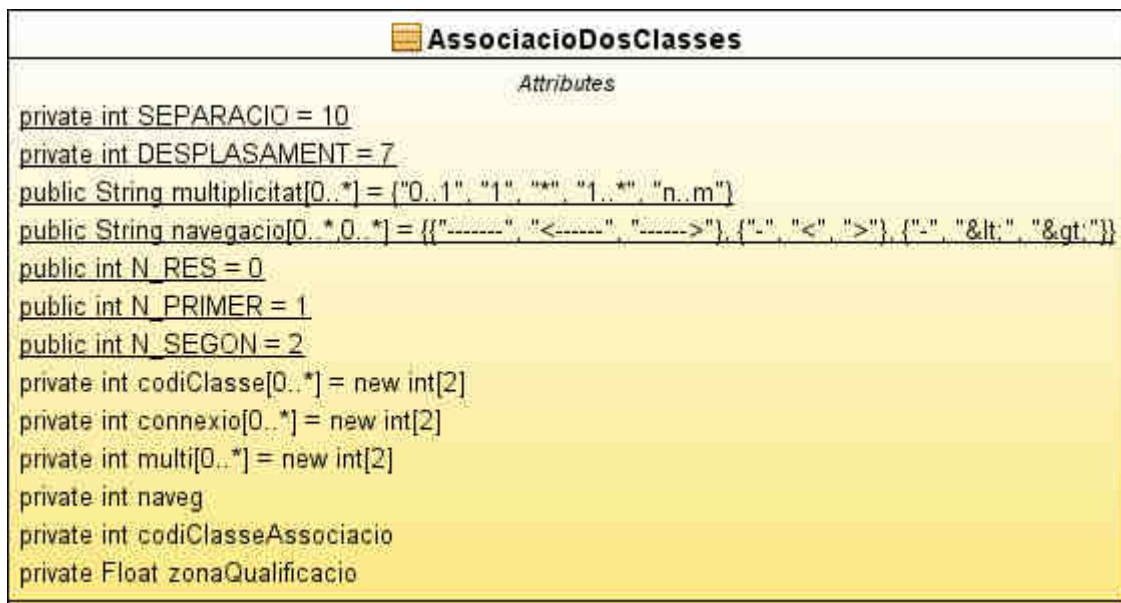


Figura 5.15: Atributs de la classe AssociacioDosClasses.

- **infoAss:** objecte de tipus InfoAssociacio amb la configuració del tipus d'associació concret obtingut a partir d'un fitxer de configuració.
- **atributs:** en cas que l'associació sigui qualificable, conté els atributs qualificadors.
- **codiClasse:** conté els codis de les classes dels extrems. En la posició zero està el codi de la primera classe i en la posició u la de la segona classe.
- **connexio:** número de connexió de la classe on està connectada l'associació, l'ordre és el mateix que en codiClasse.
- **multi:** multiplicitats de les connexions, guarda la posició de l'atribut multiplicitat que conté la multiplicitat desitjada o -1 en cas que no tingui multiplicitat. L'ordre de les multiplicitats és el mateix que en els atributs codiClasse i connexio.
- **naveg:** guarda el tipus de navegació que té l'associació. Els seus valors són N_RES si té navegació bidireccional o no es pot definir, N_PRIMER si la navegació està en la primera classe o N_SEGON si està en la segona classe.
- **codiClasseAssociacio:** guarda el codi de la ClasseAssociacio que pot tenir (que en pugui tenir o no depèn de la configuració).
- **zonaQualificacio:** en cas que l'associació sigui qualificable, conté el rectangle que envolta els atributs qualificadors en el dibuix per tal de poder usar-lo en seleccionar un element.

Mètodes

A la pàgina següent hi ha un esquema de tots els mètodes de la classe. Els mètodes més importants de la classe són:

- **estaDintre:** comprova si el punt passat pel paràmetre està dintre la zona del nom de l'associació o en cas que l'associació sigui qualificable si el punt està dintre del rectangle zonaQualificacio.


 AssociacioDosClasses
<i>Operations</i>
<pre> public AssociacioDosClasses(InfoAssociacio ia) public AssociacioDosClasses(int codi, String nom, InfoAssociacio ia) private void partComunaConstructors(InfoAssociacio ia) public void setTipus(InfoAssociacio ia) public void setNavegacio(int val) public int getNavegacio() public int getMultiplicitat1() public void setMultiplicitat1(int multi) public int getMultiplicitat2() public void setMultiplicitat2(int multi) public int getCodiClasse1() public int getConnexio1() public void setClasse1(int codiClasse, int connexio) public int getCodiClasse2() public int getConnexio2() public void setClasse2(int codiClasse, int connexio) public void eliminarConnexions() public void setClasseAssociacio(int classeAssociacio) public int getClasseAssociacio() public String getTipus() public String getNomTipus() public void girarDades() public boolean isQualificable() public boolean isClasseAssociacio() public Atribut[0..*] getAtributs() public void addAtribut(Atribut a) public Atribut getAtribut(int i) public int getNumAtributs() public void removeAtribut(int i) public void removeAtribut(Atribut a) public boolean estaDintreQualificacio(Point p) private void dibuixaMulti(int i, Point p, int costat, Graphics2D g) private void dibuixarTrosLinia(int costat, Point p, int desplaament, Graphics2D g) private void dibuixarQualificacio(Graphics2D g, Point p, int costat, boolean esCanvasGran, boolean seleccionat) private String getCadCorMulti(int i) private String getCadCorAtrQual() private String getCadenaDibuixConnexio(int i) </pre>
<i>Operations Redefined From Element</i>
<pre> public boolean estaDintre(Point p) public void remove() public void dibuixar(Graphics2D g, float zoom, boolean esCanvasGran, boolean seleccionat) public String getCadenaCorreccio() public String getCadenaDibuix() </pre>

Figura 5.16: Mètodes de la classe AssociacioDosClasses.

- **remove**: mètode cridat des de la classe ConjuntElements quan s'elimina un element del conjunt. Indica a les classes dels seus extrems que eliminin les connexions a l'associació i indica a la possible classe associació que s'elimini.

- **dibuixar(Graphics2D, float, boolean, boolean)**: dibuixa l'associació sobre l'objecte Graphics2D passat. Si el tercer paràmetre és cert, es recalculen les mides de la zona on es dibuixa el nom. I si el tercer i quart paràmetres són cert, es dibuixa l'associació com a seleccionada.

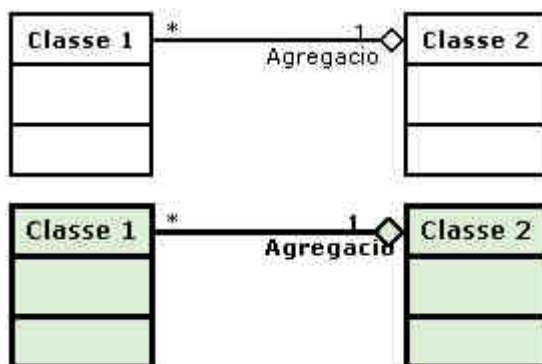


Figura 5.17: Exemple d'una agregació sense seleccionar i la mateixa agregació seleccionada.

Classe ClasseAssociacio

Classe que implementa la representació de una classe associació de UML, de fet, només la unió de una relació amb una classe.

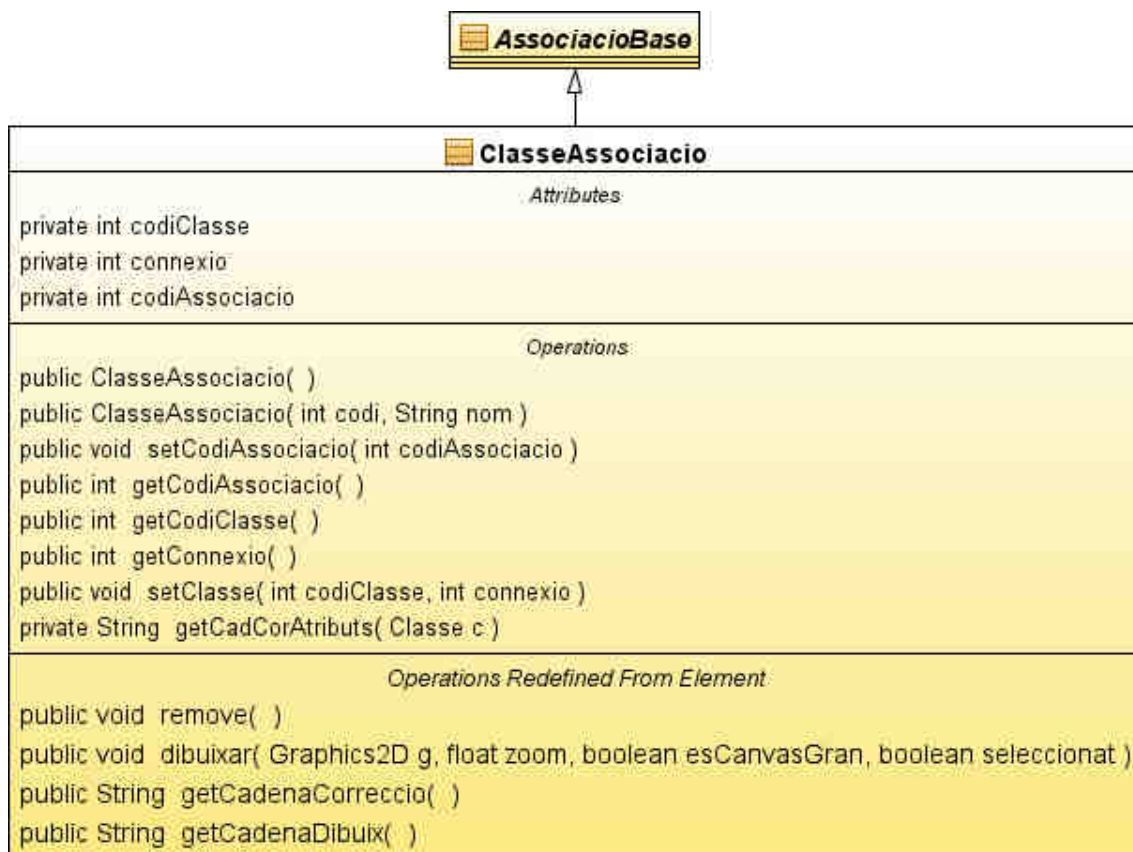


Figura 5.18: Herència de la classe ClasseAssociacio.

Atributs

Els atributs de la classe són:

- **codiClasse:** codi de la classe que està unida a la classe associació.
- **connexio:** número de connexió de la classe amb codi codiClasse on està unida la classe associació.
- **codiAssociacio:** codi de la classe associació que està unida a la classe associació.

Mètodes

Els mètodes més importants de la classe són:

- **remove:** mètode cridat des de la classe ConjuntElements quan s'elimina un element del conjunt. Indica a la classe i a l'associació dels seus extrems de la

classe associació que eliminin les connexions a la classe associació.

- **dibuixar(Graphics2D, float, boolean, boolean)**: dibuixa la classe associació sobre l'objecte Graphics2D passat. Si el tercer paràmetre és cert, es recalculen les mides de la zona on es dibuixa el nom. I si el tercer i quart paràmetres són cert, es dibuixa l'associació com a seleccionada.

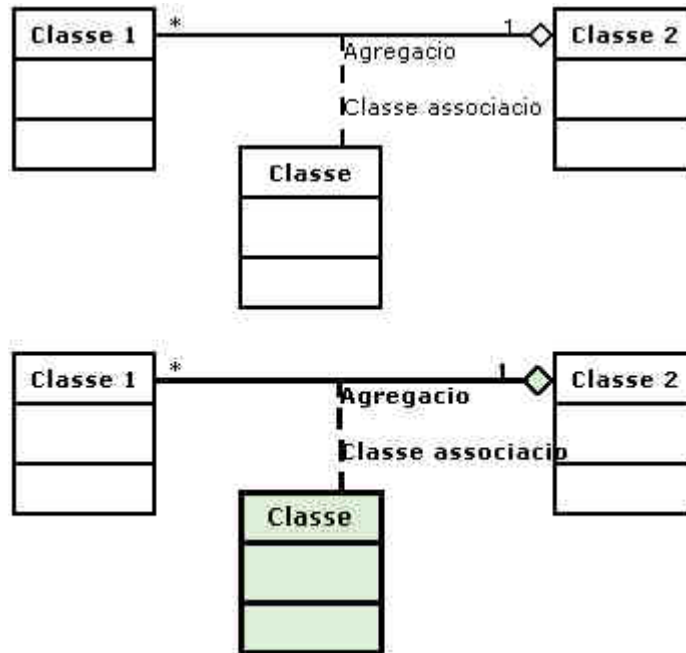


Figura 5.19: Exemple d'una classe associació sense seleccionar i la mateixa classe associació seleccionada.

Classe ConjuntElements

Classe que conté tots els elements (classes, associacions i classes associació) que formen un diagrama de classes. També és la classe que retorna les cadenes de caràcters de la correcció i del dibuix i també restaura aquesta última. Hi ha un instància d'aquesta classe que s'utilitza en tot l'editor (és un dels atributs estàtics de AppletClasses). En aquesta classe, el tipus Element es refereix a la classe org.w3c.dom.Element de Java i no a la classe Element de l'editor.

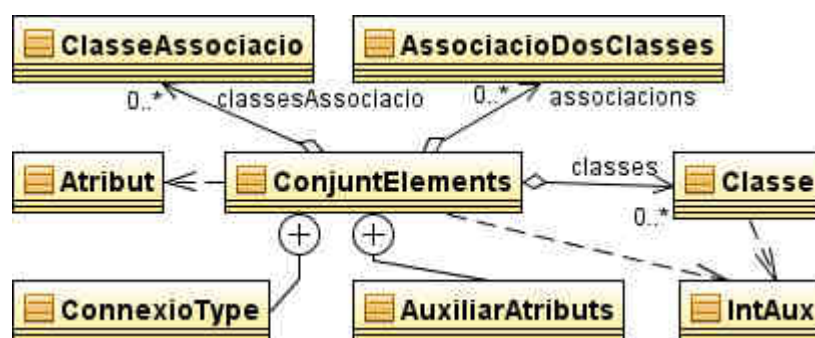


Figura 5.20: Dependències, atributs i classes privades de la classe *ConjuntElements*.

Atributs

Els atributs de la classe són:

- **classes**: aquest atribut guarda els objectes de tipus Classe que s'han anat afegint a un diagrama.
- **associacions**: aquest atribut guarda els objectes de tipus AssociacioDosClasses que s'han anat afegint a un diagrama.
- **classesAssociacio**: aquest atribut guarda els objectes de tipus ClasseAssociacio que s'han anat afegint a un diagrama.
- **mapElements**: mapa (parella clau – valor) que guarda els elements afegits a la classe associats amb el seu codi (clau).
- **mapNoms**: mapa (parella clau – valor) que guarda els noms dels objectes de l'estructura.

Mètodes

A la pàgina següent hi ha un esquema de tots els mètodes de la classe. Els mètodes més importants de la classe són:

- **add**: mètodes per a afegir objectes de tipus Classe, AssociacioDosClasses o ClasseAssociacio.
- **IteratorClasse**, **iteratorAssociacio** i **iteratorClasseAssociacio**: retornen un

iterador (classe que permet recórrer els elements d'una estructura sense saber-ne la implementació concreta) de objectes de tipus Classe, AssociacioDosClasses o ClasseAssociacio respectivament.

- **remove(Element), remove(int), removeClasse, removeAssociacio, removeClasseAssociacio**: eliminen un element del conjunt. Els dos primers eliminen l'element sigui quin sigui el tipus real. Els tres següents només eliminen l'element en cas que sigui del tipus indicat en el nom del mètode.
- **reset**: elimina tots el contingut del conjunt.
- **get, getClasse, getAssociacio, getClasseAssociacio**: retornen un element del conjunt (o nul en cas de no trobar-lo). El primer el retorna com a Element i els altres tres com al tipus indicat en el nom del mètode.
- **getCadenaCorreccio**: retorna una cadena de caràcters amb la definició d'una solució preparada per a corregir. Mirar la secció de “Estructura d'una solució” (veure pàg. 92) del capítol “Definició d'un problema” per a veure l'estructura de la cadena de correcció.
- **getCadenaDibuix**: retorna una cadena de caràcters amb la cadena que representa un dibuix. Aquesta cadena és un fitxer XML generat
- **setCadenaDibuix**: a partir de la cadena que representa un dibuix del paràmetre, recupera el dibuix que representa. Aquest mètode és un parser XML que no té recuperació d'errors, per tant, si hi ha algun error, es descarta l'element que l'ha provocat (excepte si l'error està produït perquè algun atribut no ha estat passat a l'editor). En principi, si a l'editor se li passen els atributs, una cadena generada per l'editor sempre pot ser restaurada.

 ConjuntElements
<i>Operations</i>
<pre> public ConjuntElements() public void add(Classe c) public void add(AssociacioDosClasses as) public void add(ClasseAssociacio ca) public Iterator<Classe> iteratorClasse() public Iterator<AssociacioDosClasses> iteratorAssociacio() public Iterator<ClasseAssociacio> iteratorClasseAssociacio() public void remove(Element e) public void remove(int codi) public void removeClasse(int codi) public void removeAssociacio(int codi) public void removeClasseAssociacio(int codi) public void reset() public Element get(int codi) public Classe getClasse(int codi) public AssociacioDosClasses getAssociacio(int codi) public ClasseAssociacio getClasseAssociacio(int codi) public int getCodiPunt(Point p) private int getCodiPunt(Point p, Iterable v) public boolean conteCadenesProhibides(String s) public boolean interseca(Classe c) public boolean interseca(int x, int y, int amplada, int altura) public boolean containsNom(String s) public void removeNom(String s) public void addNom(String s) public void eliminaAtributs() public String getAtributs() public String getCadenaCorreccio() public String getCadenaDibuix() public String getCadenaDibuixBuit() public void setCadenaDibuix(String c) private void parserClasses(Element e, AuxiliarAtributs aa) private Classe parserClasse(Element e, AuxiliarAtributs aa) private Atribut[0..*] parserAtributsClasse(Element e, AuxiliarAtributs aa) private void parserAssociacions(Element e, AuxiliarAtributs aa) private AssociacioDosClasses parserAssociacio(Element e, AuxiliarAtributs aa) private ConnexioType parserConnexio(Element e, boolean teMultiplicitat) private Atribut[0..*] parserAtributsAssociacio(Element e, AuxiliarAtributs aa) private void parserClassesAssociacio(Element e) private ClasseAssociacio parserClasseAssociacio(Element e) private String parserAtributsElement(Element e, IntAux codi) </pre>

Figura 5.21: Mètodes de la classe ConjuntElements.

Classes privades

La classe `ConjuntElements` té i usa dues classes privades en recuperar una cadena de dibuix. Aquestes són:

- **ConnexioType**: conté les dades parcials en llegir part de la cadena de dibuix d'una associació. Usada en un mètode que es necessita retornar múltiples dades.
- **AuxiliarAtributs**: classe per a facilitar el control dels atributs mentre s'està reconstruint el diagrama a partir d'una cadena de dibuix.

Interfície Atribuible

Interfície que defineix els mètodes necessaris per tal que una classe pugui contenir objectes de tipus `Atribut`. La classe que implementi aquesta interfície ha de ser la classe `Element` o una classe derivada d'ella (o la classe especial `VectorAtributs`).



Figura 5.22: Mètodes de la interfície Atribuible.

Mètodes

Els mètodes de la interfície són:

- **addAtribut**: afegeix un atribut a l'objecte, aquest mètode a d'informar a l'atribut del codi de l'objecte que el conté.
- **getAtributs**: retorna un objecte amb tots els atributs que conté.

- **getAtribut**: retorna l'Atribut de la posició indicada de l'estructura. Si no existeix retorna nul. La numeració comença a zero.
- **getNumAtributs**: retorna el número d'Atributs de l'estructura.
- **removeAtribut(int)**: elimina l'Atribut de la posició indicada de l'estructura. Si no existeix la posició no fa res. La numeració comença a zero.
- **removeAtribut(Atribut)**: elimina l'Atribut indicat de l'estructura. Si no existeix no fa res.

Classe VectorAtributs

Classe que implementa la interfície Atribuible usada per tal de no haver de repetir codi en les classes Classe i AssociacioDosClasses.

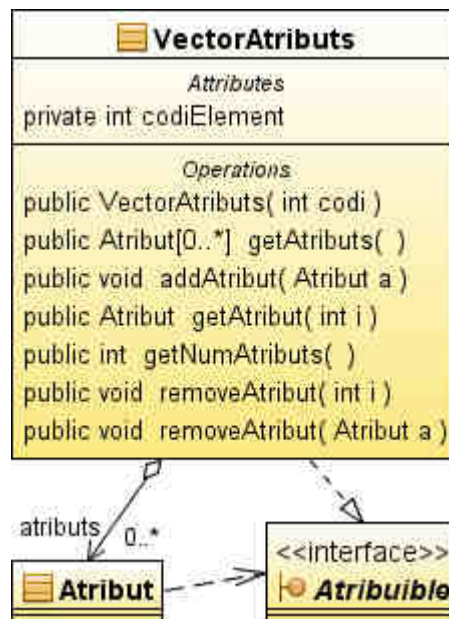


Figura 5.23: Dependències de la classe VectorAtributs

Atributs

Els atributs de la classe són:

- **codiElement**: codi de l'element que conté l'objecte.

- **atributs:** vector d'atributs que estan afegits a l'estructura.

Classe Atribut

Classe que guarda la informació d'un atribut d'una classe o una associació qualificada.

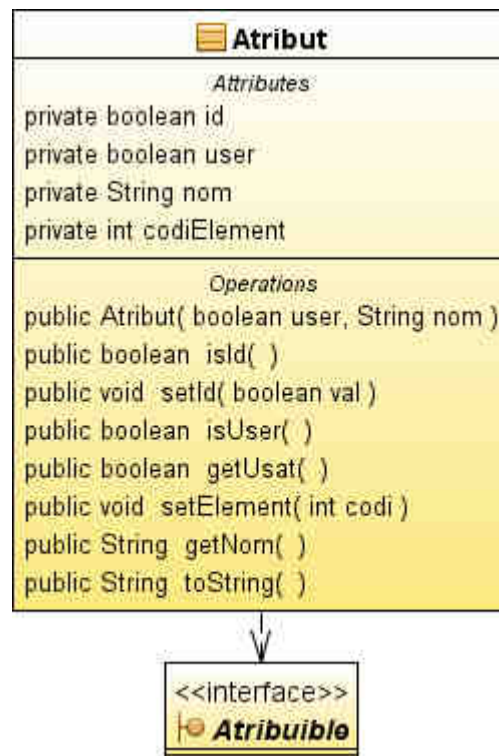


Figura 5.24: Dependències de la classe Atribut.

Atributs

Els atributs de la classe són:

- **id:** indica si l'atribut és un identificador o no. En cas que l'atribut sigui un identificador, es s'afegirà "K_" davant del nom en mostrar-lo.
- **user:** indica si l'atribut està definit per l'usuari o és un atribut definit en el problema.
- **nom:** nom de l'atribut.
- **codiElement:** codi de l'Element que el conté o la constant CODI_NO_VALID de

la classe Element en cas que no estigui dintre de cap Element.

Classe InfoAssociaicio

Classe que conté la informació d'un tipus de relació. La informació és llegida des d'un fitxer de configuració per la classe ConfigAssociacions. Una instància de la classe AssociacioDosClasses conté una referència a un objecte d'aquesta classe.



Figura 5.25: Dependències de la classe InfoAssociaicio.

Atributs

Els atributs més importants de la classe són:



Figura 5.26: Atributs de la classe InfoAssociaicio.

- **id**: identificador del tipus de relació que defineix.
- **mapImatges**: mapa (parella clau – valor) amb les imatges a mostrar en els

costats d'una classe (dreta, esquerra, part superior i inferior).

Mètodes

Els mètodes més importants de la classe són:

```

InfoAssociacio
Operations
public InfoAssociacio( )
public String getId( )
public void setId( String id )
public String getNom( )
public void setNom( String nom )
public boolean getMult1( )
public void setMult1( boolean mult1, int defecte )
public int getDefecteMulti1( )
public boolean getMult2( )
public void setMult2( boolean mult2, int defecte )
public int getDefecteMulti2( )
public boolean getMult( int i )
public int getDefecteMulti( int i )
public ImageIcon getImgIcona( )
public void setImgIcona( ImageIcon imgIcona )
public boolean getDirec1( )
public ImageIcon getImgIconaDirec1( )
public void setDirec1( boolean direc, ImageIcon icon )
public boolean getDirec2( )
public ImageIcon getImgIconaDirec2( )
public void setDirec2( boolean direc, ImageIcon icon )
public boolean isQualificable( )
public void setQualificable( boolean qualificable )
public boolean getLiniaDiscontinua( )
public void setLiniaDiscontinua( boolean liniaDiscontinua )
public boolean isClasseAssociacio( )
public void setClasseAssociacio( boolean classAss )
public void setImgTot( String imgTot )
public void setImgCostat( String img, int costat )
public ImagePack getImage( int costat, boolean seleccionat )
public boolean teMinimsAtributsNecessaris( )
public boolean telmatgeSegonaConnexio( )
private void recalculaExistencialmatges( )

```

Figura 5.27: Mètodes de la classe InfoAssociacio.

- **teMinimsAtributsNecessaris**: indica si l'objecte té els atributs mínims

necessaris per a poder ser usat. Els atributs necessaris són un identificador de tipus, un nom de tipus, una imatge per posar en el boto de creació en la interfície gràfica de l'editor i o té imatges definides per a tots els costats o no tenir imatges per a cap dels costats.

- **getImage**: retorna la imatge de decoració (i que identifica gràficament) un tipus d'associació a partir del costat (dreta, esquerra, part superior o inferior) d'un classe on s'ha de dibuixar i si l'associació en concret està seleccionada o no.

5.5.3. Classes de la interfície

A continuació hi ha l'explicació de les classes que formen part de la interfície gràfica de l'editor. Aquestes classes són:

- AppletClasses
- Principal
- AbstractCanvas
- MyCanvas
- PetitCanvas
- Traduccio
- ConfigAssociacions

Classe AppletClasses

Classe que deriva de la classe que incorpora el Java javax.swing.JApplet. Crea l'editor, també serveix per interactuar amb el servidor de l'ACME mitjançant una pàgina web o Javascript. Aquesta classe llegeix la informació que rep en el document web i crea la finestra principal i la configuració de les associacions.

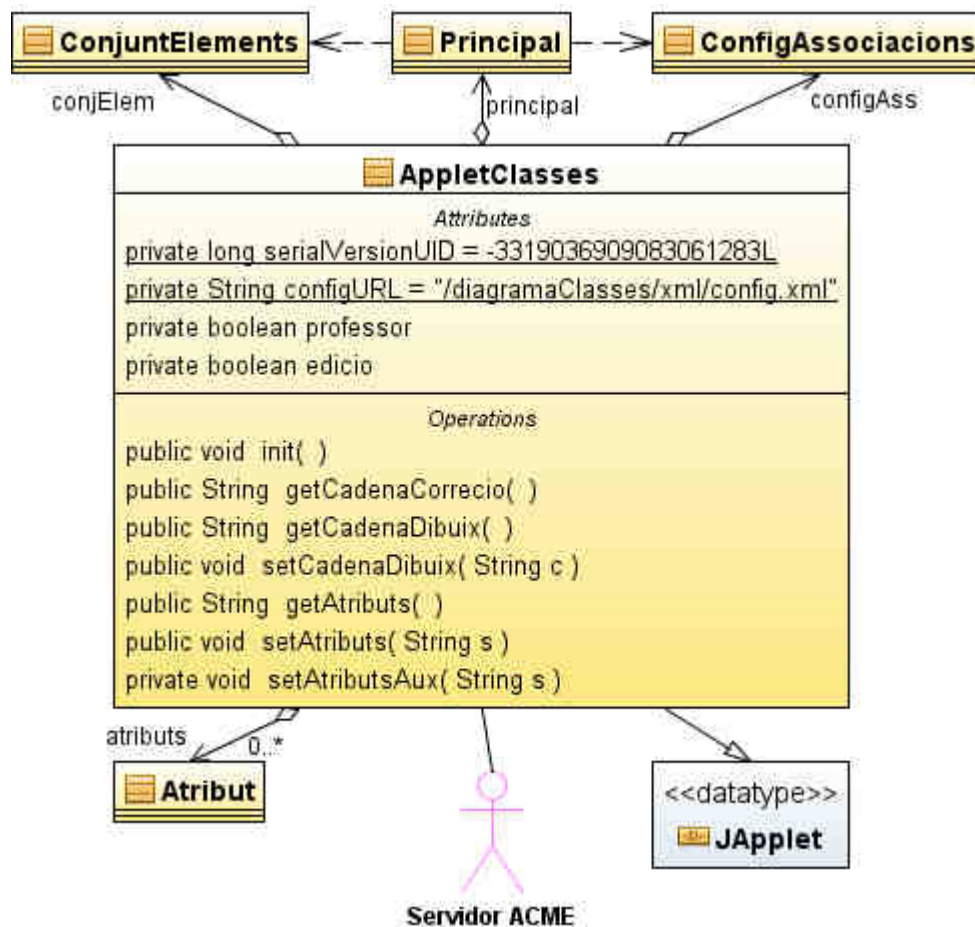


Figura 5.28: Classe *AppletClasses* amb totes les dependències, atributs i mètodes.

Atributs

Els atributs més importants de la classe són:

- **Vector `atributs`:** guarda els atributs definits en el problema i que són els que tindrà disponibles l'alumne per defecte. Es un atribut estàtic public de la classe, qualsevol classe de l'editor en pot tenir accés.
- **`configAss`:** objecte el qual crea la configuració de les associacions a partir del fitxer indicat a l'atribut `configURL`.
- **`conjElem`:** conjunt d'elements usat en l'applet. És un atribut estàtic de la classe `AppletClasses`. Totes les altres classes que l'utilitzen, en fan una importació estàtica (accedeixen a aquest atribut directament amb el seu nom sense usar el nom de la classe `AppletClasses`).

- **principal**: objecte que crea la finestra principal de l'editor.

Mètodes

- **init**: mètode cridat des del navegador que hagi obert una pàgina web que contingui l'editor.
- Els altres mètodes serveixen per interactuar amb l'editor des de codi Javascript inclòs en la pàgina web que conté l'editor. Estan explicats a la secció “Funcions per a interactuar amb l'editor des d'una pàgina web” (veure pàg. 115) del capítol de “Integració amb la plataforma ACME”.

Classe Principal

Classe que deriva de la classe que incorpora el Java javax.swing.JPanel. Aquesta classe crea la major part de la interfície gràfica, la resta la formen MyCanvas i PetitCanvas. I conté els mètodes que donen resposta als events de la interfície (inclosos els de l'objecte canvasG).

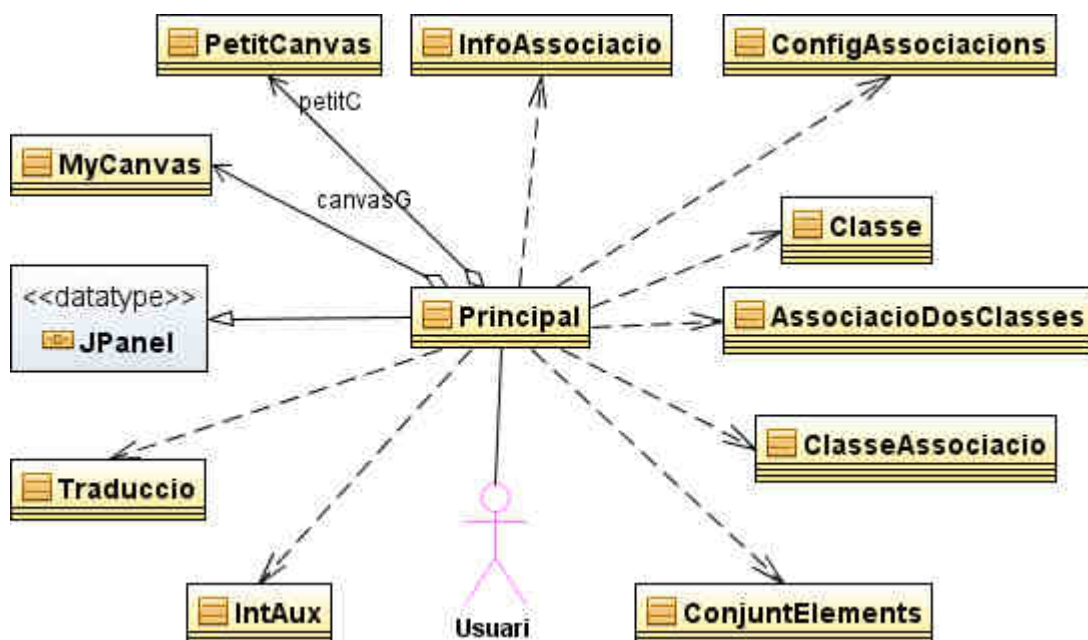


Figura 5.29: Classe Principal amb les seves dependències.

Atributs

A les següents pàgines es mostren tots els atributs de la classe. Atributs més importants de la classe.

- **canvasG**: instància de MyCanvas, és la zona de dibuix principal. Els events d'aquest objecte es tracten a principal per tal de facilitar el codi.
- **petitC**: instància de PetitCanvas, és la zona on es veu la miniatura del dibuix.
- **accio**: controla que s'ha de fer en pulsar sobre l'objecte canvasG. Els seus valors son els de les constants definides a principal que el seu nom comença per "DIBUIX_".
- **canvis**: vector usat per implementar els botons de desfer i refer. Guarda cadenes de caràcter amb la representació del dibuix.

Principal

Attributes

```

private long serialVersionUID = 7013461121182195498L
private int DIBUIX_RES = 0
private int DIBUIX_CLASSE = 1
private int DIBUIX_ESBORRAR = 2
private int DIBUIX_ASSOCIACIO_2_CLASSES = 3
private int DIBUIX_ASSOCIACIO_2_CLASSES_PART2 = 4
private int DIBUIX_CLASSE_ASSOCIACIO = 5
private int DIBUIX_CLASSE_ASSOCIACIO_PART2_CLASSE = 6
private int DIBUIX_CLASSE_ASSOCIACIO_PART2_ASSOCIACIO = 7
private int TAB_CLASSE = 0
private int TAB_ASSOCIACIO = 1
private int TAB_QUALIFICAR = 2
private int TAB_CLASSE_ASSOCIACIO = 3
private int TIPUS_NO_VALID = -1
private int TIPUS_CLASS = 0
private int TIPUS_ASSOC = 1
private int TIPUS_CLA_AS = 2
private int accio = DIBUIX_RES
private String tipusAssoc2Class
private int navegAssoc2Class
private int elementSelec = CODI_NO_VALID
private int tipusElementSelec = TIPUS_NO_VALID
private int primSelec
private boolean edicio
private Point punt
private Point desplaament
private String canvis[0..*] = new Vector<String>()
private int index = 0
private int numBotoCrear = 2
private JButton associacio_btnEliminar
private JButton associacio_btnGirarDades
private JButton associacio_btnModificar
private JButton associacio_btnNou
private JButton associacio_btnQualificar
private JComboBox associacio_cmbClasse1
private JComboBox associacio_cmbClasse2
private JComboBox associacio_cmbMultiplicitat1
private JComboBox associacio_cmbMultiplicitat2
private JComboBox associacio_cmbNavegacio
private JComboBox associacio_cmbTipus
private JLabel associacio_lbClasse1
private JLabel associacio_lbClasse2
private JLabel associacio_lbIMultiplicitat1

```

Figura 5.30: Atributs de la classe Principal, part 1.

```
private JLabel associacio_iblMultiplicitat2
private JLabel associacio_iblNavegacio
private JLabel associacio_iblNom
private JLabel associacio_iblTipus
private JPanel associacio_panGeneral
private JTextField associacio_txtNom
private Canvas canvas
private JPanel canvas_panBorder
private JButton classeAssociacio_btnEliminar
private JButton classeAssociacio_btnModificar
private JButton classeAssociacio_btnNou
private JComboBox classeAssociacio_cmbAssociacio
private JComboBox classeAssociacio_cmbClasse
private JLabel classeAssociacio_iblAssociacio
private JLabel classeAssociacio_iblClasse
private JLabel classeAssociacio_iblNom
private JPanel classeAssociacio_panGeneral
private JTextField classeAssociacio_txtNom
private JButton classe_btnAtributsAfegirAtribut
private JButton classe_btnAtributsDesferId
private JButton classe_btnAtributsFerId
private JButton classe_btnAtributsNou
private JButton classe_btnAtributsTreureAtribut
private JButton classe_btnEliminarAssociacio
private JButton classe_btnNomModificar
private JComboBox classe_cmbEliminarAssociacio
private JLabel classe_iblAtributsClasse
private JLabel classe_iblAtributsDisponibles
private JLabel classe_iblAtributsNou
private JLabel classe_iblEliminarAssociacio
private JLabel classe_iblNom
private JList classe_istAtributsClasse
private JList classe_istAtributsDisponibles
private JPanel classe_panAtributs
private JPanel classe_panEliminarAssociacio
private JPanel classe_panNom
private JScrollPane classe_scriAtributsClasse
private JScrollPane classe_scriAtributsDisponibles
private JSeparator classe_sepAtributs
private JTextField classe_txtAtributsNou
private JTextField classe_txtNom
private JButton control_btnCancelar
private JButton control_btnDesfer
private JButton control_btnEsborrar
private JButton control_btnRefer
```

Figura 5.31: Atributs de la classe Principal, part 2.

```

private JButton control_btnReset
private JPanel control_pan
private JButton crear_btnClasse
private JButton crear_btnClasseAssociacio
private JPanel crear_pan
private JScrollPane crear_scrIPan
private JPanel informacio_panAssociacions
private JPanel informacio_panClasse
private JPanel informacio_panClasseAssociacio
private JPanel informacio_panQualificar
private JTabbedPane informacio_tab
private Canvas petit
private JButton qualificar_btnAtributsAfegir
private JButton qualificar_btnAtributsTreure
private JLabel qualificar_lblAtributsAssociacio
private JLabel qualificar_lblAtributsDisponibles
private JLabel qualificar_lblNom
private JList qualificar_lstAtributsAssociacio
private JList qualificar_lstAtributsDisponibles
private JPanel qualificar_panGeneral
private JScrollPane qualificar_scrAtributsAssociacio
private JScrollPane qualificar_scrAtributsDisponibles
private JTextField qualificar_txtNom
private JButton zoom_btnMenys
private JButton zoom_btnMes
private JPanel zoom_pan

```

Figura 5.32: Atributs de la classe Principal, part 3.

Els atributs que el nom del seu tipus comença per J (JButton, JLabel, JPanel...) més els atributs de tipus Canvas són els atributs que formen els objectes de la interfície gràfica de l'editor. Aquests atributs (excepte els dos de tipus Canvas), per a facilitar entendre el codi, segueixen unes regles en el nom. La primera part (abans del símbol de subratllat) és el nom del grup on pertanen (per exemple els objectes de la pestanya de modificació de classes és classe). Després del símbol de subratllat hi ha les inicials del tipus (btn és un JButton; pan, un JPanel; etc.) i tot seguit la seva funció.

Mètodes

A les següents pàgines hi ha un esquema amb els mètodes de la classe.


 Principal	
<i>Operations</i>	
<pre> public Principal(boolean edicio, Color fons) private void initComponents() private Canvas initCanvas() private void initAssociacions() private void control_btnResetActionPerformed(ActionEvent evt) private void control_btnEsborrarActionPerformed(ActionEvent evt) private void control_btnCancelarActionPerformed(ActionEvent evt) private void control_btnDesferActionPerformed(ActionEvent evt) private void control_btnReferActionPerformed(ActionEvent evt) private void crear_btnClasseActionPerformed(ActionEvent evt) private void crear_btnClasseAssociacioActionPerformed(ActionEvent evt) private void crear_btnAssociacioDosClasses(ActionEvent evt, String tipus, int naveg) private void zoom_btnMesActionPerformed(ActionEvent evt) private void zoom_btnMenysActionPerformed(ActionEvent evt) private void informacio_tabStateChanged(ChangeEvent evt) private void classe_btnNomModificarActionPerformed(ActionEvent evt) private void classe_btnEliminarAssociacioActionPerformed(ActionEvent evt) private void classe_btnAtributsNouActionPerformed(ActionEvent evt) private void classe_btnAtributsFerIdActionPerformed(ActionEvent evt) private void classe_btnAtributsDesferIdActionPerformed(ActionEvent evt) private void classe_btnAtributsTreureAtributActionPerformed(ActionEvent evt) private void classe_btnAtributsAfegirAtributActionPerformed(ActionEvent evt) private void classe_lstAtributsClasseMouseClicked(MouseEvent evt) private void classe_lstAtributsDisponiblesMouseClicked(MouseEvent evt) private void associacio_cmbTipusActionPerformed(ActionEvent evt) private void associacio_btnGirarDadesActionPerformed(ActionEvent evt) private void associacio_btnNouActionPerformed(ActionEvent evt) private void associacio_btnModificarActionPerformed(ActionEvent evt) private void associacio_btnEliminarActionPerformed(ActionEvent evt) private void associacio_btnQualificarActionPerformed(ActionEvent evt) private void qualificar_btnAtributsTreureActionPerformed(ActionEvent evt) private void qualificar_btnAtributsAfegirActionPerformed(ActionEvent evt) private void qualificar_lstAtributsAssociacioMouseClicked(MouseEvent evt) private void qualificar_lstAtributsDisponiblesMouseClicked(MouseEvent evt) private void classeAssociacio_btnNouActionPerformed(ActionEvent evt) private void classeAssociacio_btnModificarActionPerformed(ActionEvent evt) private void classeAssociacio_btnEliminarActionPerformed(ActionEvent evt) private void canvas_MousePressed(MouseEvent e) private void canvas_MouseReleased(MouseEvent e) private void canvas_MouseClicked(MouseEvent e) private void canvas_MouseDragged(MouseEvent e) private void canvas_MouseMoved(MouseEvent e) private void canvas_MouseWheelMoved(MouseWheelEvent e) </pre>	

Figura 5.33: Mètodes de la classe Principal, part 1.


```

private AssociacioDosClasses crearAssociacio( int classe1, int classe2, String tipus, int navegacio )
private void modificarConnexionsAssociacio( AssociacioDosClasses adc, int codi1, int codi2 )
private boolean crearClasseAssociacioDesdeCanvas( int codiAssociacio, int codiClasse )
private ClasseAssociacio crearClasseAssociacio( int codiAssociacio, int codiClasse )
private void modificarConnexioClasseAssociacio( ClasseAssociacio ca, int codiClasse, int codiAssociacio )
private void posarInfoTabClasse( Classe c )
private void posarInfoTabAssociacio( AssociacioDosClasses adc )
private void posarInfoTabQualificacio( AssociacioDosClasses adc )
private void posarInfoTabClasseAssociacio( ClasseAssociacio ca )
private DefaultComboBoxModel getComboModelAssociacionsPerClasse( Classe c )
private DefaultComboBoxModel getComboModelClassesPerAssociacions( )
private DefaultComboBoxModel getComboModelClassesPerClasseAssociacio( )
private DefaultComboBoxModel getComboModelAssociacionsPerClasseAssociacio( )
private void seleccionarItemComboCBS( JComboBox jcb, int codi )
public String arreglarString( String s )
private void emplenarAtributsDisponibles( )
private void passarCoordenadesDibuixClasse( Point p, Classe c )
private void gestionarCanvis( )
private void ocultarBotons( )
private void repinta( )
private void setAccio( int accio )
private void setElementSeleccionat( int codi )
private void setElementSeleccionat( Element e )
public void setCadenaDibuix( String c )
public void actualitzaAtributs( )
private void actualitza( )
public void afegirBotoCrear( String tipus, int init, ImagemIcon icon, String nom )

```

Figura 5.34: Mètodes de la classe Principal, part 2.

Mètodes més importants de la classe.

- **InitComponents:** mètode creat completament pel NetBeans, crea la interfície.

Classes privades:

Aquesta classe conté i utilitza quatre classes privades.

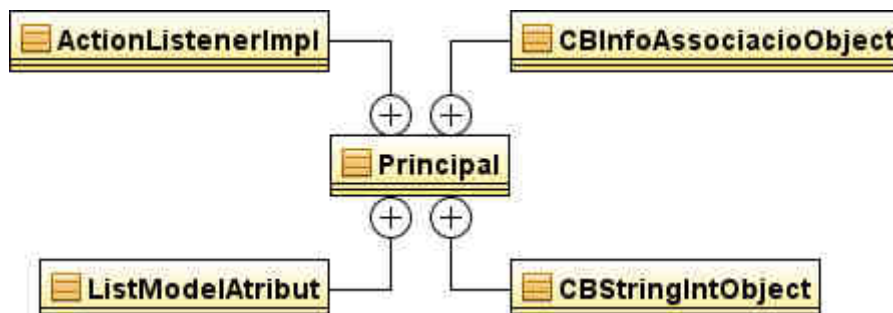


Figura 5.35: Classes privades de la classe Principal.

- **ActionListenerImpl**: implementació personalitzada de l'objecte que respon als events dels botons de creació de les associacions.
- **CBInfoAssociacioObject**: classe per poder crear llistes desplegable de objectes InfoAssociacio.
- **CBStringIntObject**: classe per a crear llistes desplegable amb la informació dels dibuixos (classes, associacions i classes associació).
- **ListModelAtribut**: implementació personalitzada d'una llista d'atributs.

Classe AbstractCanvas

Classe abstracte (no es pot instanciar) que deriva de la classe que incorpora el Java `java.awt.Canvas`. Defineix una zona de dibuix que no parpelleja la imatge en dibuixar-hi (com passa en la classe base).

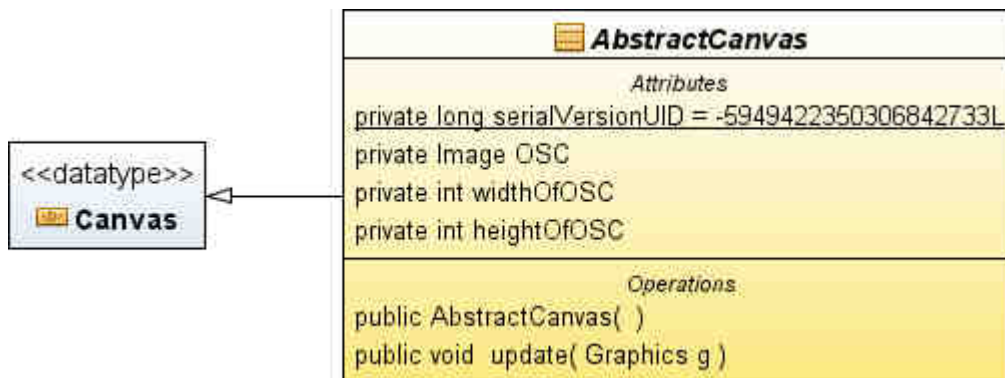


Figura 5.36: Classe abstracte AbstractCanvas.

Atributs

- **OSC**: imatge que s'utilitza per a dibuixar-hi abans de dibuixar sobre l'objecte.
- **widthOfOSC**: amplada de la imatge OSC.
- **heightOfOSC**: altura de la imatge OSC.

Mètodes

- **update**: sobreesciu el mètode de la superclasse. Aquest mètode és cridat per

Java sempre que cal redibuixar l'objecte. Aquí és on es prepara la imatge OSC per a dibuixar-hi i després es passa el dibuix de la imatge sobre l'objecte.

Classe MyCanvas

Classe que deriva de AbstractCanvas. Defineix la zona de dibuix principal de l'editor. L'usuari interectua amb una instància d'aquesta classe per a dibuixar en l'editor. Conté els mètodes per a canviar el zoom de la imatge i per a dibuixar sobre seu.

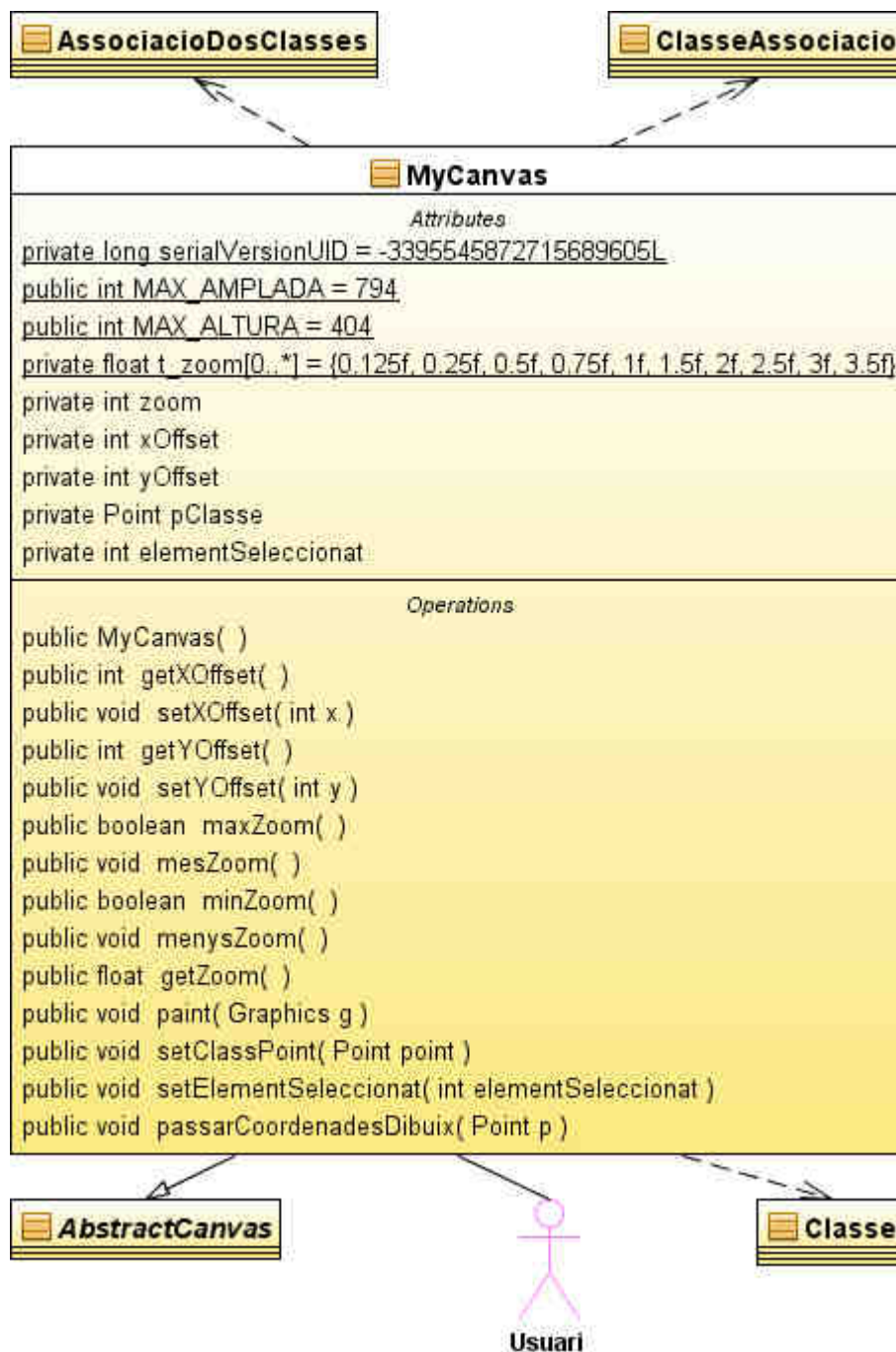


Figura 5.37: Dependències de la classe MyCanvas.

Atributs

- **t_zoom**: taula amb els valors que serveixen per a calcular l'escala a la que s'ha de dibuixar el dibuix.
- **zoom**: posició de t_zoom que s'està usant actualment.

- **xOffset** i **yOffset**: guarden les coordenades de dibuix de vèrtex superior esquerra de l'objecte.
- **pClasse**: punt on està la rata en cas que s'estigui dibuixant una classe o nul en cas que no s'estigui dibuixant una classe. Serveix per a dibuixar l'esquema d'una classe que segueixi la rata.
- **elementSeleccionat**: codi de l'element del dibuix seleccionat actualment. Usat per a pintar diferent l'element seleccionat.

Mètodes

- **paint**: dibuixa el dibuix sobre l'objecte passat com a paràmetre. El seu funcionament és el següent:

```
Si hi ha elementSeleccionat
  Preparar estructura amb els elements a marcar diferents.
Fsi
Per totes les classes del dibuixar-hi
  Si la classe està seleccionada
    Dibuirar més marcada.
  Altrament
    Dibuirar normal.
  Fsi
Fper
Per totes les associacions del dibuixar-hi
  Si l'associació està seleccionada
    Dibuirar més marcada.
  Altrament
    Dibuirar normal.
  Fsi
Fper
Per totes les classes associació del dibuixar-hi
  Si la classe associació està seleccionada
    Dibuirar més marcada.
  Altrament
    Dibuirar normal.
  Fsi
Fper
Si s'ha de dibuixar l'esquema d'una classe
  Dibuirar l'esquema de la classe.
Fsi
```

Classe *PetitCanvas*

Classe que deriva de *AbstractCanvas*. Defineix la zona on es veu la miniatura del dibuix. L'usuari hi interectua per a canviar la zona del dibuix que es veu i per a canviar el zoom del dibuix (amb la roda de la rata).

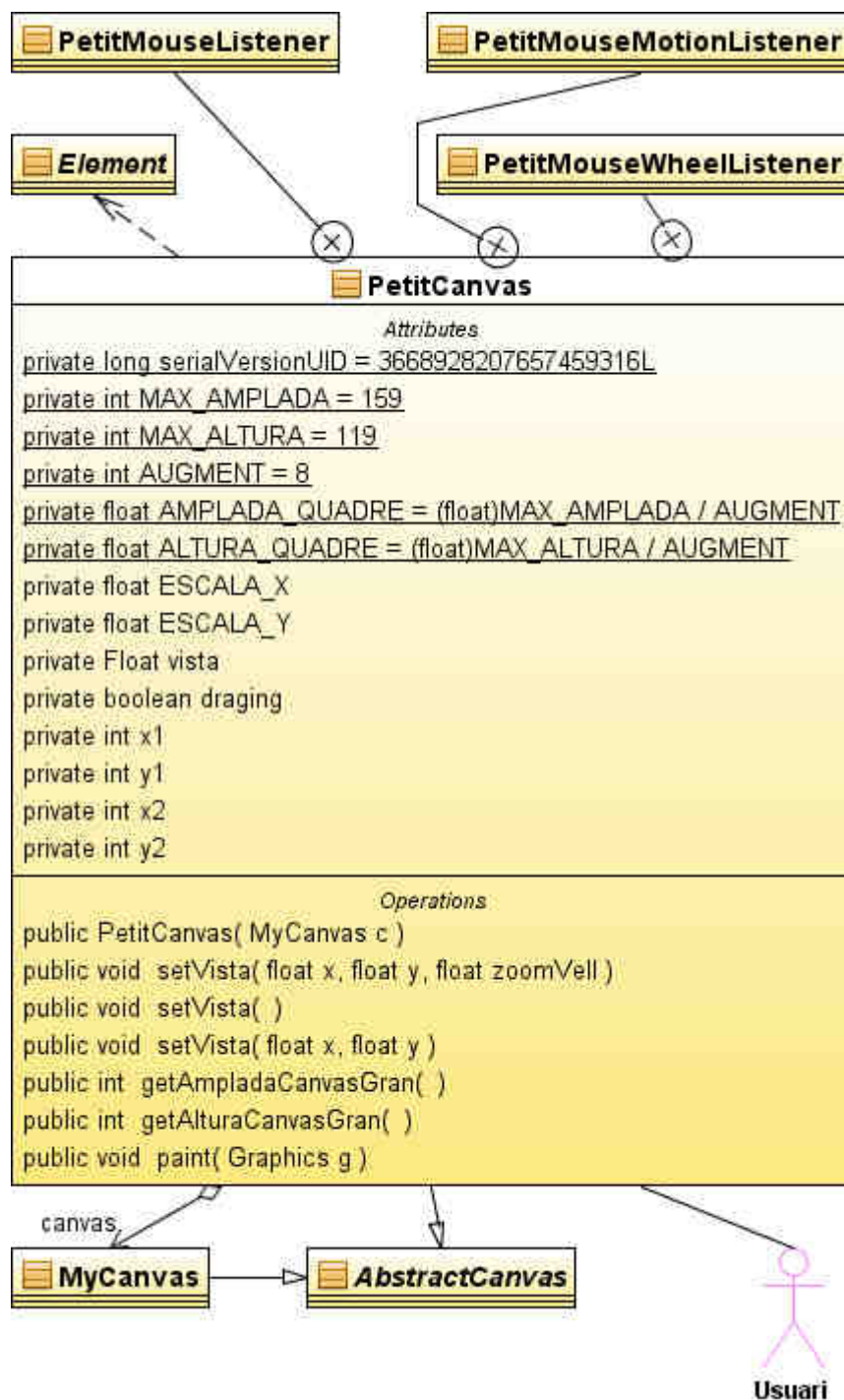


Figura 5.38: Dependències i classes privades de *PetitCanvas*.

Atributs

Els atributs més importants de la classe són:

- **canvas**: instància de MyCanvas sobre la que ha de controlar la zona visible.

Mètodes

El mètode més important és:

- **paint**: dibuixa el dibuix sobre l'objecte passat com a paràmetre. El seu funcionament és el següent:

```
Per totes les classes del dibuixar-hi
  Dibuir normal.
Fper
Per totes les associacions del dibuixar-hi
  Dibuir normal.
Fper
Per totes les classes associació del dibuixar-hi
  Dibuir normal.
Fper
```

Classes privades

Aquesta classe conté tres classes privades que serveixen per respondre als events de la rata sobre l'objecte. Quan es crea un objecte PetitCanvas, aquest afegeix un objecte de cada una d'aquestes classes per a respondre als seus events.

- **PetitMouseListener**: controla els events de pressionar un botó de la rata, de deixar-lo anar i de fer les dues accions sobre el mateix punt.
- **PetitMouseMotionListener**: controla l'event de arrossegar la rata sobre un objecte PetitCanvas.
- **PetitMouseWheelListener**: controla l'event de girar la roda de la rata sobre un objecte PetitCanvas.

Classe Traduccio

Classe no instanciable que serveix per a facilitar la traducció de la interfície.

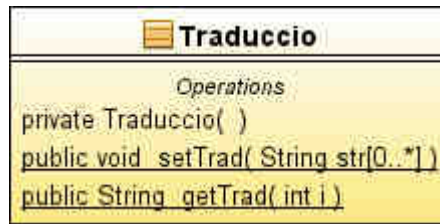


Figura 5.39: Classe Traduccio.

Atributs

- **catala**: taula amb els textos de la traducció per defecte en català.
- **act**: taula que s'utilitza actualment per a la traducció.
- Atributs que comencen per "T_": contenen la posició de taula amb els textos de traducció on hi ha el que defineixen. Son els possibles paràmetres del mètode getTrad.

Mètodes

- **setTrad**: canvia la taula de traducció actual per la passada.
- **getTrad**: retorna la cadena de caràcters demanada. El seu funcionament és el següent:

```

Si el paràmetre indica una posició vàlida de act
  Retorna l'element de la posició indicada de l'atribut act.
Altrament si el paràmetre indica una posició vàlida de l'atribut
catala
  Retorna l'element de la posició indicada de l'atribut catala.
Altrament
  Retorna un text d'error.
Fsi
  
```

Classe ConfigAssociacions

Llegeix el fitxer de configuració amb la informació dels diferents tipus de associacions i guarda la informació de cada tipus d'associació en objectes InfoAssociacio. En aquesta classe, el tipus Element es refereix a la classe org.w3c.dom.Element de Java i no a la classe Element de l'editor.

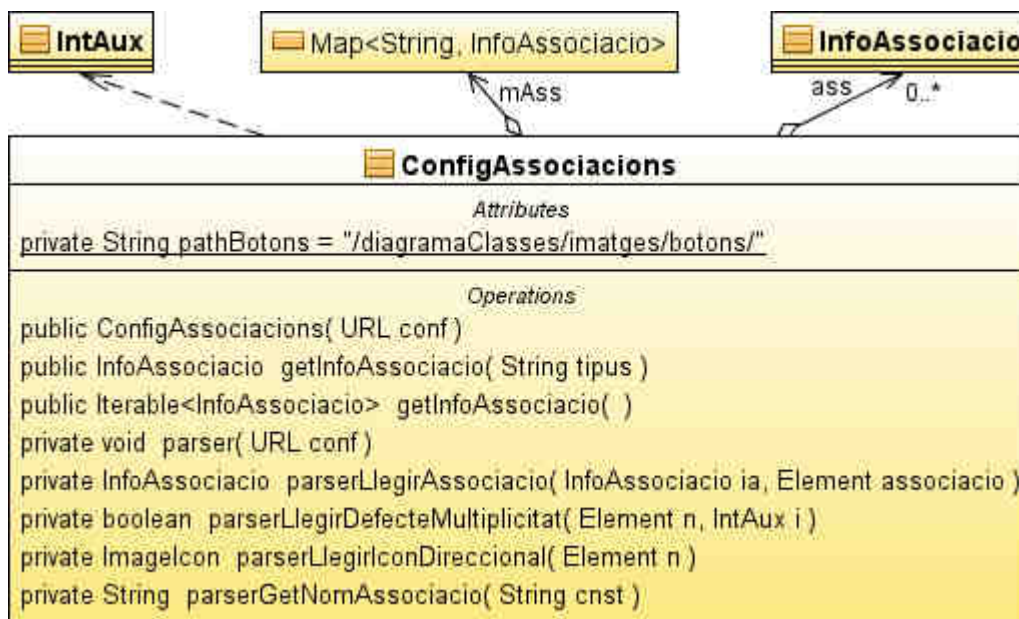


Figura 5.40: Dependències de la classe ConfigAssociacions

Atributs

Els atributs més importants de la classe són

- **ass**: vector que guarda la informació de les associacions amb l'ordre que estan en el fitxer de configuració.
- **mAss**: mapa (parella clau - valor) que guarda els mateixos objectes que l'atribut ass però accessible per el seu tipus (clau).

Mètodes

Els mètodes més importants de la classe són:

- **getInfoAssociacio(String)**: a partir del tipus d'una associació retorna l'objecte que conté la informació del tipus.
- **getInfoAssociacio**: retorna els objectes amb la informació dels tipus.
- **parser**: llegeix el fitxer de configuració que es troba a l'adreça indicada pel paràmetre. El parser no té cap mecanisme de recuperació d'errors, per tant, si al llegir la configuració d'un tipus d'associació es produeix un error, descarta el tipus.

5.6. Configuració de les associacions

A la secció "XML de configuració de les associacions" de l'"Annex A" hi ha el fitxer de configuració que usa l'editor.

La configuració de les associacions està definida en un fitxer XML amb la següent estructura (la part en negreta està explicada després):

```
<?xml version="1.0" encoding="UTF-8"?>
<associacions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="config_schema.xsd">
Llista d'associacions definides
</associacions>
```

5.6.1. Llista d'associacions

Una associació té la següent estructura (la part en negreta s'explica després):

```
<associacio id="identificador">
  <nom>nom</nom>
  <multiplicitat_1 defecte="valor">boolea</multiplicitat_1>
  <multiplicitat_2 defecte="valor">boolea</multiplicitat_2>
  <direccional_1 icon="imatge">boolea</direccional_1>
  <direccional_2 icon="imatge">boolea</direccional_2>
  <qualificable>boolea</qualificable>
  <linia_discontinua>boolea</linia_discontinua>
  <classe_associacio>boolea</classe_associacio>
  <icona>imatge</icona>
  Altres imatges
</associacio>
```

La majoria de propietats que s'han de definir són booleans, és a dir que poder tenir els valors cert o fals. En tots aquests casos es considerarà cert sí i només sí el valor *true* (sense tenir en compte majúscules i minúscules).

- Atribut **id**: és una atribut obligatori que conté un identificador del tipus d'associació. Ha de tenir tres lletres en majúscules, sense poder repetir una combinació. Aquest és el valor que s'utilitza en definir un tipus d'associació en una solució.
- **Nom**: nom de la constant de la classe Traduccio de l'editor que serveix per a obtenir el nom d'aquest tipus d'associació. Si el valor aquí definit no existeix és mostrarà el text *ERROR* en lloc del nom esperat.

- **multiplicitat_1**: defineixen si es podrà modificar la multiplicitat en la primera connexió de l'associació. És un valor booleà.
 - Atribut **defecte**: és un atribut opcional que hauria d'aparèixer obligatòriament en cas la multiplicitat tingui un valor cert. Els possibles valors d'aquest atribut són: "0..1", "1", "*", "1..*", "n..m". Si la multiplicitat té un valor cert però, o no té l'atribut definit o s'ha definit un valor incorrecte s'usarà el valor "0..1". Si està definit (o no ho està, però la multiplicitat té un valor cert) és obligatori que en una solució del problema que usi el tipus d'associació definit tingui definida la multiplicitat indicada. Si no està definit i la multiplicitat que el conté és fals, una solució no ha de tenir la multiplicitat indicada per aquest tipus d'associació.
- **multiplicitat_2**: defineixen si es podrà modificar la multiplicitat en la segona connexió de l'associació. És un valor booleà.
 - Atribut **defecte**: el mateix que l'atribut defecte de multiplicitat_1, però aplicat a la multiplicitat_2.
- **direccional_1**: indica si es podrà posar direccionalitat en la primera connexió de associació. És un valor booleà.
 - Atribut **icon**: atribut opcional per a tenir un botó en la interfície de l'editor per a crear l'associació amb la direccionalitat en la connexió indicada. Només es tindrà en compte si la direccionalitat és certa. Ha de ser el nom d'una imatge que es trobi a */diagramaClasses/imatges/botons/*. Si la imatge no existeix, es farà com sí no hagués estat definida.
- **direccional_2**: indica si es podrà posar direccionalitat en la segona connexió de associació. És un valor booleà.
 - Atribut **icon**: el mateix que l'atribut icon de direccional_1, però aplicat a direccional_2.
- **qualificable**: indica si en l'associació es poden definir atributs qualificadors. És un valor booleà.

- **linia_discontinua**: indica si en dibuixar una associació d'aquest tipus cal fer la línia discontinua o continua. És un valor booleà.
- **classe_associacio**: indica si l'editor a de permetre afegir una classe associació en una associació del tipus que es defineix. És un valor booleà.
- **icona**: nom de la imatge, que s'ha de trobar a */diagramaClasses/imatges/botons/*, que s'utilitzarà en el botó de la interfície de l'editor que servirà per a dibuixar aquest tipus d'associació. Si la imatge no existeix, no es podrà usar aquest tipus.

5.6.2. Imatges

El lloc on en l'anterior esquema està indicat com a *Altres imatges* indica quina imatge o imatges s'utilitzaran en la segona connexió de l'associació per a dibuixar el tipus. Aquestes imatges s'han de trobar en */diagramaClasses/imatges/dibuix/* i/o */diagramaClasses/imatges/dibuix/seleccio/*. Si no es troba una imatge en un dels dos directoris s'usarà la de l'altre. Aquest bloc pot ser buit, tenir definida aquesta línia:

```
<tot>imatge</tot>
```

- **tot**: la imatge definida aquí s'usarà en tots els costats d'una classe a l'hora de dibuixar la segona connexió d'una associació d'aquest tipus. Si la imatge no es troba a cap dels dos directoris, es farà el mateix que si no s'hagués definit la propietat.

o aquest conjunt:

```
<dreta>imatge</dreta>  
<esquerra>imatge</esquerra>  
<superior>imatge</superior>  
<inferior>imatge</inferior>
```

Si alguna de les imatges definides en aquest bloc no existeix, no es podrà usar la definició de l'associació, si són totes les que fallen, si que és podrà usar.

- **dreta**: imatge usada quan s'hagi de dibuixar la segona connexió de l'associació definida a la dreta d'una classe.

- **esquerra**: imatge usada quan s'hagi de dibuixar la segona connexió de l'associació definida a l'esquerra d'una classe.
- **superior**: imatge usada quan s'hagi de dibuixar la segona connexió de l'associació definida a la part superior d'una classe.
- **Inferior**: imatge usada quan s'hagi de dibuixar la segona connexió de l'associació definida a la part inferior d'una classe.

5.7. Representació d'un dibuix

Un dibuix es guarda en una cadena de caràcters amb estructura XML amb la següent estructura (la part en negreta està explicada després):

```
<?xml version="1.0" encoding="UTF-8"?>
<dibuix xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="dibuix_schema.xsd">
  <classes>
    Llista de classes.
  </classes>
  <associacions>
    Llista d'associacions.
  </associacions>
  <classesAssociacio>
    Llista de classes associació.
  </classesAssociacio>
</dibuix>
```

Les tres "llistes" són opcionals, però cal tenir en compte que per a poder definir una associació ha d'existir un mínim de la classe dels seus extrems. I per a poder definir una classe associació cal que com a mínim existeixi una associació i una altre classe que no estigui connectada a l'associació.

5.7.1. Part comuna als diferents tipus

Tant la representació d'una classe, d'una associació i d'una classe associació tenen una part comuna: en l'etiqueta d'obertura tenen uns atributs comuns:

- Atribut **nom**: nom de l'objecte, és el nom que mostra l'editor. Ha de ser únic, és a dir no hi pot haver cap altre classe, associació o classe associació que utilitzi un

nom ja usat.

- Atribut **id**: codi numèric intern que utilitza l'editor. En aquesta representació s'utilitza per a referenciar indicar les classes dels extrems de les associacions i per a indicar l'associació i classe dels extrems d'una classe associació. Igual que amb l'atribut nom, no es pot repetir.

5.7.2. Llista de classes

Una llista de classes està formada per la repetició del següent tros un per a cada classe. Els atributs nom i id ja estan explicats a abans.

```
<classe nom="nom_classe" id="numero">
  <x>numero</x>
  <y>numero</y>
  <atributs>
    Llista d'atributs
  </atributs>
</classe>
```

- x: coordenada x del punt superior esquerra de la classe.
- y: coordenada y del punt superior esquerra de la classe.

Llista d'atributs

La llista d'atributs està formada per elements amb la següent estructura:

```
<atribut tipus="user" esID="true">nom_atribut</atribut>
```

El mínim és de cap atribut. El valor nom_atribut ha de ser el nom de l'atribut definit.

- Atribut **tipus**: pot prendre els valor “user” (l'atribut l'ha definit l'usuari) o “prof” (és un atribut definit en el problema).
- Atribut **esID**: valor booleà, és a dir que poder tenir els valors cert o fals. Es considerarà cert sí i només sí el valor val *true* (sense tenir en compte majúscules i minúscules).

5.7.3. Llista d'associacions

Una llista d'associacions està formada per la repetició del següent tros un per a cada associació. Els atributs nom i id ja estan explicats a abans. El mínim és no definir cap associació.

```
<associacio nom="nom_associacio" id="numero">
  <tipus>tipus</tipus>
  <direccio>navegacio</direccio>
  <connexio_1>
    <classe>identificador_classe</classe>
    <connexio>numero</connexio>
    <multiplicitat>multiplicitat</multiplicitat>
  </connexio_1>
  <connexio_2>
    <classe>identificador_classe</classe>
    <connexio>numero</connexio>
    <multiplicitat>multiplicitat</multiplicitat>
  </connexio_2>
  <atributs>
    <atribut>nom_atribut</atribut>
  </atributs>
</associacio>
```

- **tipus:** tipus de l'associació. Ha de ser un dels identificadors de tres lletres definits en el fitxer de configuració de les associacions.
- **direccio:** indica la navegació de l'associació. Els seus valor són: "-", "<" i ">" (els dos últims són els símbols < i > codificats per no interferir en l'estructura del document). "-" indica que l'associació és bidireccional o que no es pot definir cap altre navegació, "<" (<) indica que la navegació està en la classe de la connexió 1 i ">" (>) que la navegació està en la classe de la connexió 2. Pot ser que "<" i/o ">" no siguin vàlids segons el tipus (mirar la definició d'una associació).

Nota aclaridora:

En el següent dibuix es pot veure la diferència entre la primera i segona connexió d'una associació.

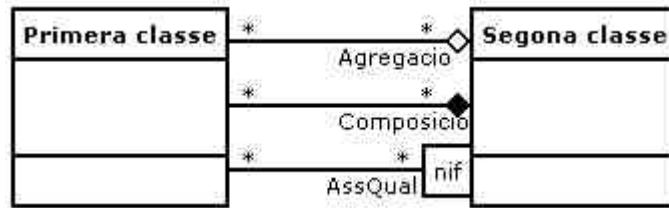


Figura 5.41: Exemple de les connexions d'una associació: "Primera classe" és la classe de la primera connexió i "Segona classe" és la de la segona.

- **connexio_1**: conté les dades de la primera connexió d'una associació.
 - **classe**: valor de l'atribut id de la classe la qual està connecta l'associació en la primera connexió.
 - **connexio**: número de connexió de la classe de la primera connexió usada per l'associació en la primera connexió.
 - **multiplicitat**: multiplicitat de la connexió, obligatori definir-la només si en la configuració del tipus d'associació s'indica que s'ha de permetre modificar la multiplicitat de la connexió o té un valor de multiplicitat per defecte. Els possibles valors són: "0..1", "1", "*", "1..*", "n..m".
- **connexio_2**: conté les dades de la segona connexió d'una associació. Els seus elements són iguals als de connexio_1 però aplicats a la segona connexió.
- **atributs**: llista d'atributs. Només a d'aparèixer en cas que l'associació sigui qualificable i tingui un mínim d'un atribut, si no passa cap d'aquestes condicions no s'ha de posar aquest element. Cada atribut només guarda el nom, tots han de ser atributs definits en el problema.

5.7.4. Llista de classes associació

Una llista de classes associació està formada per la repetició del següent tros un per a cada classe associació. Els atributs nom i id ja estan explicats abans.

```
<classeAssociacio nom="nom_classe_associacio" id="numero">
  <associacio>identificador_associacio</associacio>
```



```
<classe>identificador_classe</classe>
<connexio>numero</connexio>
</classeAssociacio>
```

- **associacio:** valor de l'atribut id de l'associació de la qual depèn la classe associació.
- **classe:** valor de l'atribut id de la classe de la qual depèn la classe associació.
- **connexio:** número de connexió de la classe usada per la classe associació.

5.8. L'editor

Una vegada implementades les classes i el codi corresponent s'ha anat provant i millorant les funcionalitats de l'editor. El resultat final de la interfície dissenyada ha estat:

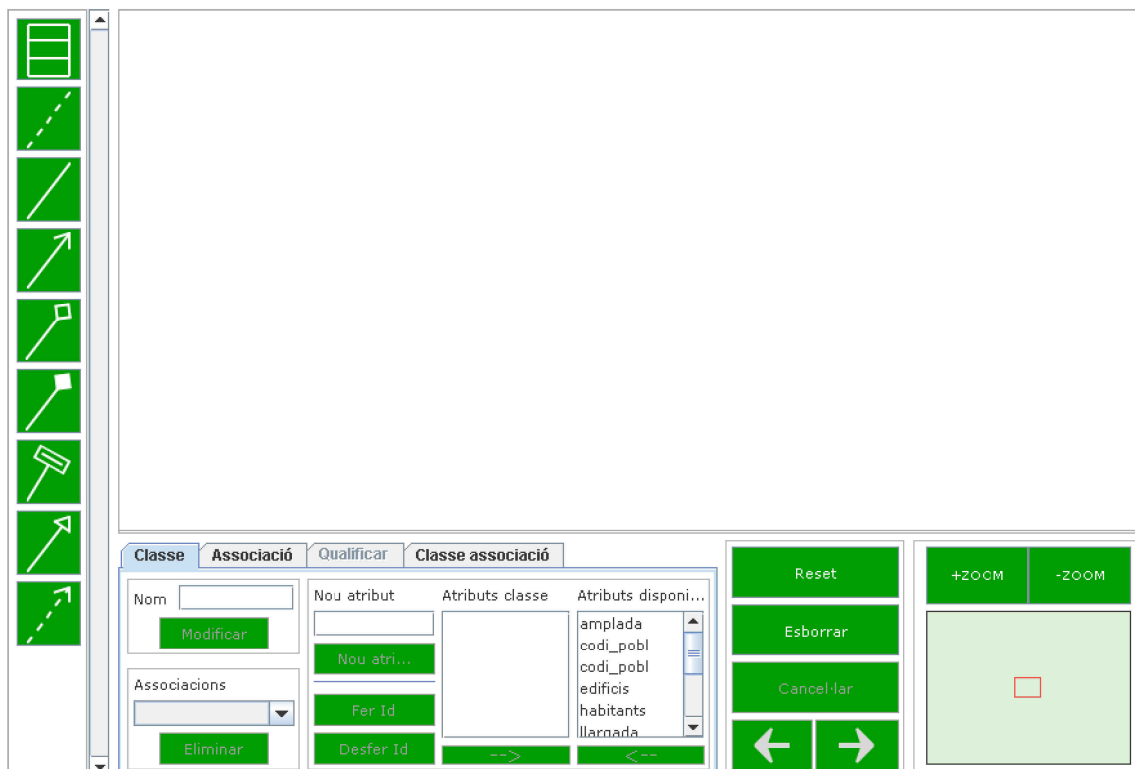


Figura 5.42: Interfície gràfica de l'editor.

A continuació hi ha l'explicació del funcionament de l'editor.



Figura 5.43: Gràfic d'explicació del funcionament de l'editor.

1. **Botons de control.** Aquest botons serveixen:

- **Reset:** elimina tot el dibuix. Abans d'eliminar-lo pregunta si estàs segur de fer-ho. Sempre es pot desfer l'acció amb el botó desfer (←).
- **Esborrar:** si hi ha un element seleccionat l'esborra del dibuix. Si no hi ha un element seleccionat, s'esborrarà el següent element que es polsi dos cops seguits.
- **Cancel·lar:** cancel·la l'acció de l'últim botó de dibuix (sempre que no s'hagi dibuixat) o del botó esborrar si es dona el segon cas. Només es pot usar si hi ha alguna acció a cancel·lar.
- ← (desfer): desfà l'últim canvi en el dibuix. Si es polsa múltiples cops va desfent els canvis.
- → (refer): si no s'ha canviat el dibuix des de l'últim desfer torna a refer el canvi desfet.

2. **Àrea de dibuix.** Aquesta és la zona on es dibuixa. La rata canvia al cursor que és

una mà quan es pot seleccionar un element. També es pinta diferent l'element seleccionat. Es pot usar la roda de la rata per a fer zoom. En fer zoom s'intentarà mantenir el punt sota el cursor en el mateix lloc.

3. **Botons de creació.** Si es posa la rata sobre un boto mostra l'element que dibuixa. Aquests botons en ordre descendent serveixen:

- **Classe:** permet dibuixar una classe. Per a dibuixar-la cal polsar sobre una àrea que no tingui cap classe de la zona de dibuix (2).

- **Classe associació:** permet dibuixar una classe associació. Per a dibuixar-la cal seleccionar la classe i la relació (sense importar l'ordre) entre les quals es vol dibuixar. Només es pot formar una classe associació entre una classe que no tingui cap relació ni classe associació i una associació que no tingui cap classe associació (hi ha algun tipus de relació com la dependència que no permeten classes associació).

- **Altres botons:** permeten dibuixar el tipus de relació que tenen dibuixada. Per tal de dibuixar-les cal seleccionar les dues classes entre les que s'ha de crear la relació. Cap de les dues classes pot formar part d'una classe associació. Aquests botons per ordre serveixen per a crear una associació, una associació direccional, una agregació, una composició, una associació qualificada, una generalització i una dependència.

4. **Miniatura:** aquesta zona mostra una miniatura del diagrama que s'està fent. Es pot canviar l'àrea del dibuix que es veu a la zona principal polsant sobre un punt de la miniatura. Els dos botons que hi ha a sobre la miniatura serveixen per a canviar el zoom de la imatge. També es pot canviar el zoom i moure l'àrea de dibuix amb la roda de la rata.

5. **Pestanyes** de modificació de les propietats. Aquestes pestanyes serveixen per a modificar les propietats de les classes, associacions, atributs qualificadors d'una associació qualificada (aquesta pestanya només es podrà usar si hi ha una associació qualificada seleccionada) i classes associació respectivament. Totes

les pestanyes tenen en comú que els seus botons només estan actius si es poden usar.

- **Classe:** la primera pestanya serveix per a editar una classe seleccionada.

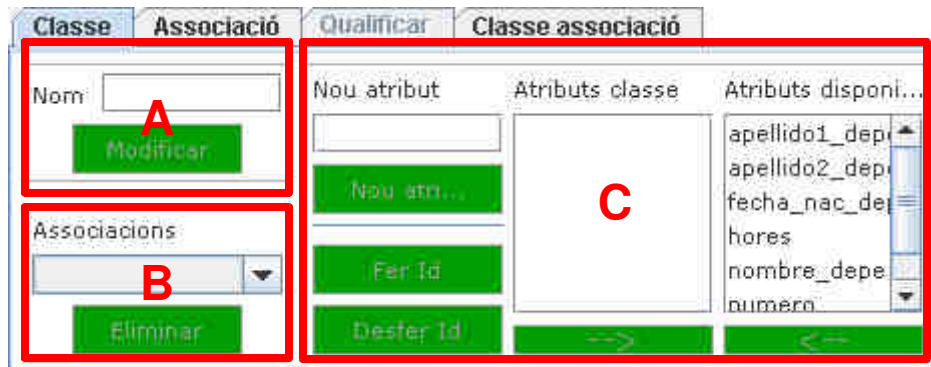


Figura 5.44: Pestanya de modificació de classes.

A: edita el nom de la classe. El nom no pot ser buit ni està repetit, ni es permeten aquests caràcters: “K_”, “\$”, “&”, “:”, “,””, “(”, “)”, “<” i “>”.

B: elimina l'associació seleccionada de la classe.

C: Edita els atributs de la classe. La primera llista són els atributs actuals de la classe i la segona els disponibles. Per canviar atributs entre llistes es pot pulsar dos cops sobre un atribut o seleccionar-lo (o seleccionar-ne varis amb l'ajuda de les tecles majúscules i control) i pulsar el botó de sota de les llistes. Per a marcar un atribut (o varis a la vegada) com a identificadors cal seleccionar-los i pulsar en el botó “Fer id”, per a desfer identificadors s'ha d'usar el botó “Desfer id”. Els atributs identificadors es marquen amb “K_” davant del nom. L'apartat “Nou atribut” permet afegir un atribut personalitzat a la classe (té les mateixes restriccions que el nom d'una classe).

- **Associació:** la segona pestanya serveix per a crear una nova associació, o editar o eliminar una associació seleccionada.

Figura 5.45: Pestanya d'edició d'associacions.

Tipus: es canvia el tipus de l'associació. La possibilitat de modificar la navegació i multiplicitats depenen del tipus escollit.

Navegació: canvia la navegació de l'associació. Els seus valors disponibles depenen del tipus.

Primera i segona classe: classes dels extrems entre les que s'ha de dibuixar l'associació. Només es llisten les classes que no formen part d'una classe associació. A continuació un exemple per entendre la diferència entre primera i segona classe.

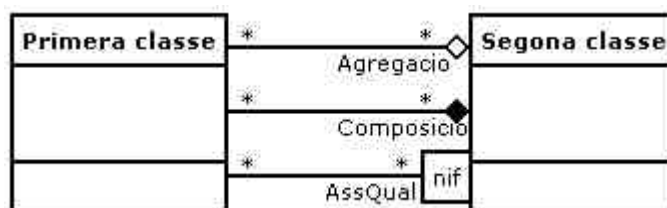


Figura 5.46: Exemple de les connexions d'una associació: "Primera classe" és la classe de la primera classe i "Segona classe" és la de la classe.

Multiplicitats: es pot escollir les multiplicitats de cada classe.

Nom: nom de l'associació. El nom no pot ser buit ni està repetit, ni es permeten aquests caràcters: "K_", "\$", "&", ":", ",", "(", ")", "<" i ">".

Nou: crea una nova associació amb la configuració actual.

Modificar: modifica l'associació seleccionada.

Eliminar: elimina l'associació seleccionada.

Girar dades: gira la informació de les classes i multiplicitats de l'associació seleccionada.

Qualificar: si l'associació és qualificable porta a la pestanya de qualificació.

- **Qualificar:** la tercera pestanya serveix per a modificar els atributs qualificadors d'una associació qualificable seleccionada.

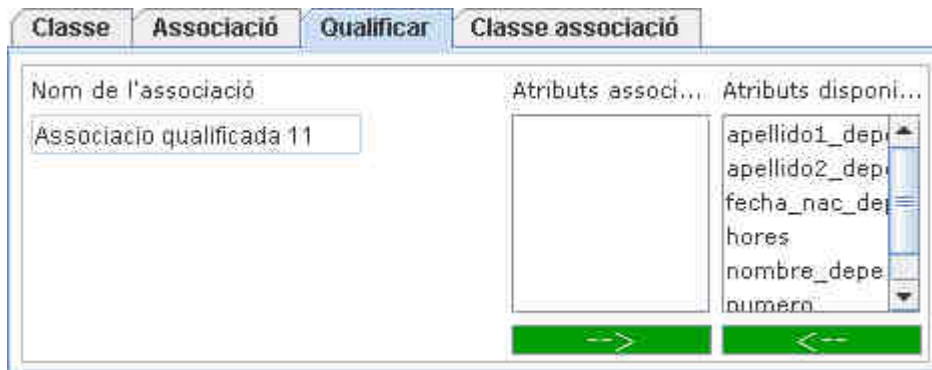


Figura 5.47: Pestanya de qualificació.

Nom: només es mostra el nom, s'ha d'editar des de la pestanya Associació.

Llistes: la primera llista són els atributs actuals de la classe i la segona els disponibles. Per canviar atributs entre llistes es pot polsar dos cops sobre un atribut o seleccionar-lo (o seleccionar-ne varis amb l'ajuda de les tecles majúscules i control) i polsar el botó de sota de la llista.

- **Classe associació:** la quarta pestanya serveix per a a crear una nova classe associació, o editar o eliminar una classe associació seleccionada.



Figura 5.48: Pestanya d'edició de les pestanyes.

Nom: nom de la classe associació. El nom no pot ser buit ni està repetit, ni

es permeten aquests caràcters: “K_”, “\$”, “&”, “:” “”, “(”, “)”, “<” i “>”.

Associació: associació de la qual depèn la classe associació. Només es llisten les associacions possibles.

Classe: classe de la qual depèn la classe associació. Només es llisten les classes possibles.

Nou: crea una nova classe associació amb la configuració actual.

Modificar: modifica la classe associació seleccionada.

Eliminar: elimina la classe associació seleccionada.

L'editor també es pot usar per mostrar una solució (i no permetre editar-la). Quan s'està en aquest mode, no es permet editar i s'oculten la majoria de botons.

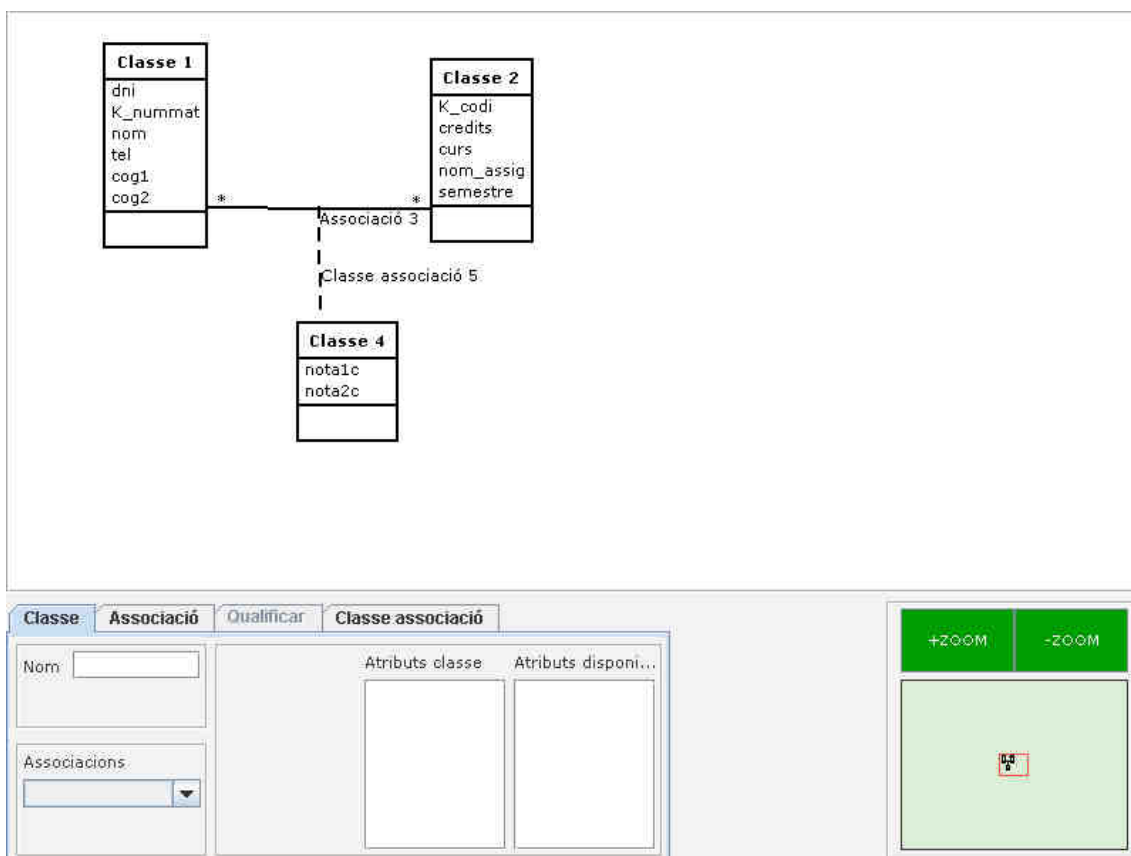


Figura 5.49: Editor en mode de visualització.

5.9. Exemples d'us de l'editor

A continuació hi ha el mateix enunciat de l'exemple de la introducció i a sota l'editor amb el diagrama de classes que correspon a l'enunciat. Recordo que l'enunciat és en castellà.

- *La empresa está organizada en departamentos. De cada departamento nos interesa saber su número, el nombre y el teléfono. Cada departamento tiene asignados empleados de los que nos interesa saber su nombre completo, NIF, dirección completa, sueldo, sexo y fecha nacimiento. El trabajo de cada empleado está supervisado por otro empleado. Uno de los empleados del departamento desempeña la función de director del departamento.*
- *Cada departamento controla un cierto número de proyectos. De cada proyecto nos interesa saber su código, su nombre y la ciudad donde se realiza. Cada proyecto es controlado por un único departamento.*
- *Un empleado puede trabajar en varios proyectos. Nos interesa saber el número de horas que cada empleado dedica a cada proyecto.*
- *Un empleado puede tener varias personas dependientes que figuran en su cartilla de la seguridad social. De cada una de estas personas nos interesa saber su número de orden dentro de la cartilla, su parentesco con el empleado, su nombre y su fecha de nacimiento.*

El sistema diseñado nos debe permitir:

- *A partir de un departamento saber todos los empleados que tiene asignados, saber quien es su director y los proyectos que controla.*
- *A partir de un empleado saber sus datos personales, los datos de su supervisor, en que proyectos ha trabajado y cuantas horas ha dedicado a cada proyecto. También nos interesa saber las personas dependiente que tiene a su cargo.*
- *A partir de un proyecto saber los datos de este, que departamento lo controla, los empleados que están trabajando en él y las horas dedicadas.*

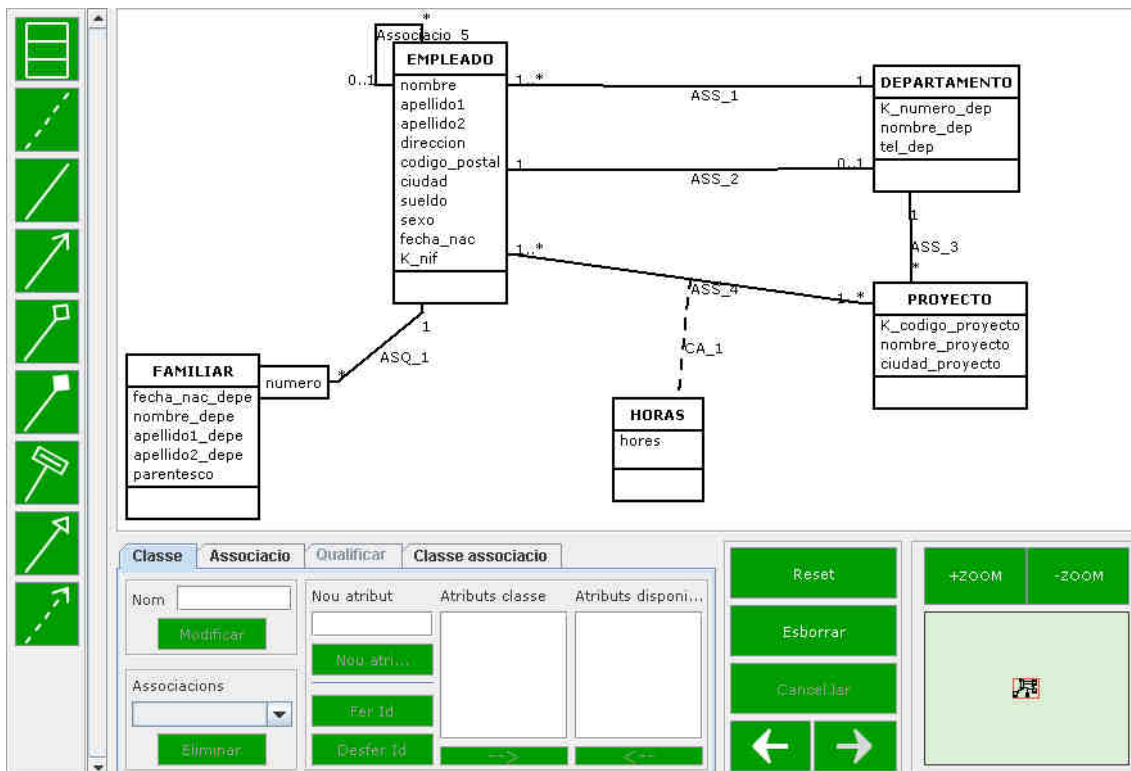
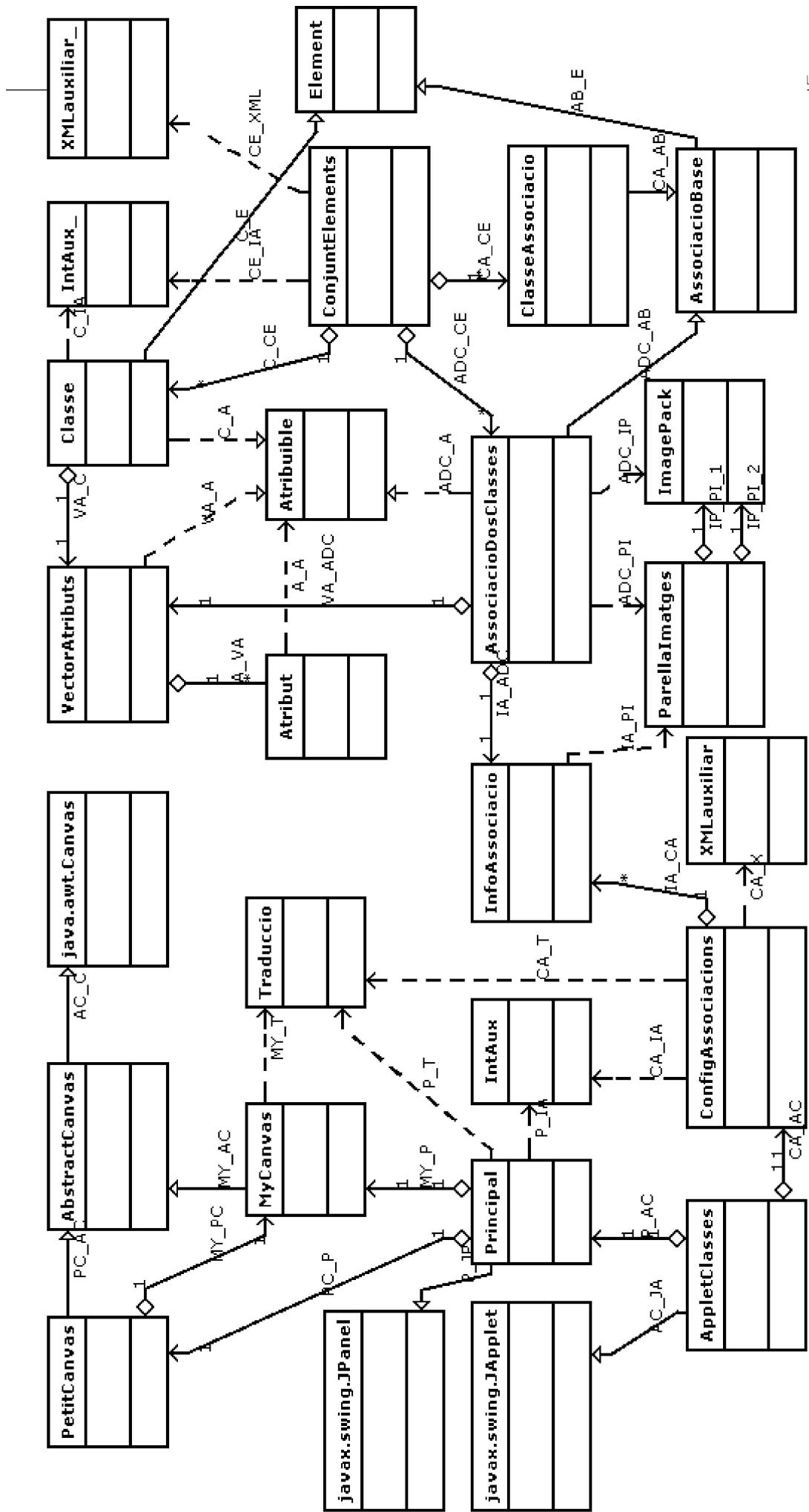


Figura 5.50: Exemple de l'editor.

A la següent pàgina hi ha el diagrama de classes de l'editor (veure pàg. 30) creat amb el propi editor.



6. DEFINICIÓ D'UN PROBLEMA

Quan un professor vol escriure un problema de diagrames de classes, ha d'escriure amb qualsevol editor de text un fitxer de text amb l'enunciat i les solucions que considera correctes. Aquest fitxers han de seguir l'estructura explicada a continuació.

6.1. Estructura d'un problema

Per definir un problema s'utilitza un fitxer de text on hi ha definits els enunciats i les solucions correctes del problema. Té la següent estructura:

```
39
<E>
Enunciat 1.
</E>
<E>
Enunciat 2.
</E>
<SOLUCIO1>
Solució 1.
</SOLUCIO1>
<SOLUCIO2>
Solució 2.
</SOLUCIO2>
```

- El número 39 és obligatori. És un número que necessita l'ACME per a saber de quin tipus de problema es tracta.
- Seguidament venen els enunciats (només un és obligatori). Els enunciats comencen per <E> i acaben per </E> i tot el que hi hagi entremig es considera l'enunciat. A dintre un enunciat no es pot usar cap de les etiquetes usades per definir un problema: <E>, </E>, <SOLUCIO_> (on _ és un número positiu) i </SOLUCIO_> (on _ és un número positiu).
- Després dels enunciats venen les solucions (només una és obligatòria). Una solució s'escriu entre <SOLUCIO_> i </SOLUCIO_> (on _ és un número positiu que ha de coincidir en les dues etiquetes). El número de solució ha de començar a 1 i anar incrementant-se de un en un. La solució n ha d'aparèixer després de la solució $n - 1$ i no es poden saltar números de solucions: per exemple, si es

defineix la solució 1 i després la solució 3 (sense definir la solució 2), la solució 3 i posteriors seran ignorades. A dintre una solució no es pot usar cap de les etiquetes usades a definir un problema: `<E>`, `</E>`, `<SOLUCIO_>` i `</SOLUCIO_>`.

- Tant en el cas d'un enunciat com en una solució, qualsevol altre contingut de les línies d'inici o final d'aquests serà ignorat:

```
Text ignorat<E>Text ignorat
Aquest text es tindrà en compte.
Text ignorat</E>Text ignorat
Text ignorat<SOLUCIO1>Text ignorat
El text que hi hagi aquí es tindrà en compte.
Text ignorat</SOLUCIO1>Text ignorat
```

6.2. Estructura d'una solució

Una solució està formada per diverses línies de text separades per salts de línia. Cada línia està formada per una paraula o paraules clau, un separador (dos punts) i el contingut de la línia. El contingut d'una línia pot ser un sol element o una llista separada per comes.

En tots els exemples d'aquesta secció, s'utilitza el color verd per marcar les paraules clau i el color vermell per marcar el símbol de coma.

Exemple d'una línia:

```
Classe: ALUMNE
```

En l'exemple anterior, "Classe" és la paraula clau de la línia i "EMPLEAT" el contingut.

Una línia correcta, ha de tenir una paraula clau, un sol signe de dos punts, excepte si la línia està buida o només conté espais i/o tabulacions (en aquests casos la línia serà ignorada).

Una solució és insensible a majúscules i minúscules, però si s'ha de mostrar algun missatge d'error es farà servir el text original. Per exemple la següent línia es considera igual a l'anterior (excepte que si s'ha de mostrar un missatge d'error l'anterior mostrarà *ALUMNE* i aquest alumne):

```
classe: alumne
```

Es poden posar espais, tabulacions o combinacions d'aquests caràcters al principi de línia, després del símbol dos punts (entre la clau i els dos punts no), a davant o darrera d'una coma que separa els elements d'una llista, al final de la línia o estar una línia composta només d'aquests caràcters. En els casos que els espais i/o tabulacions es tinguin en compte, si n'apareixen més d'un de seguits s'usarà un sol espai, és a dir, "bon dia" es considerarà el mateix que "bon dia".

Una classe, una associació i un classe associació estan compostes per diverses línies que apareixen seguides.

La cadena de caràcters "##%##" no pot ser usada en una solució ja que té un altre ús en guardar les solucions de l'alumne a la base de dades de l'ACME.

6.2.1. Classe

La definició d'una classe està formada per tres línies que han d'aparèixer seguides. Un exemple d'una classe:

```
Classe: ALUMNE
Id: nummat
Atributs: dni, nom, cog1, cog2, tel
```

- La primera de les línies d'una classe (amb paraula clau *Classe*)

```
Classe: ALUMNE
```

conté el nom de la classe. Aquesta línia no pot contenir cap coma (,).

- La segona de les línies d'una classe (paraula clau *Id*)

```
Id: nummat
```

conté l'atribut o atributs identificadors. Si hi ha més d'un atribut cal separar-los amb comes. Aquesta línia pot no tenir contingut (però la paraula clau i els dos punts han d'aparèixer).

- La tercera i última de les línies d'una classe (paraula clau *Atributs*)

```
Atributs: dni, nom, cog1, cog2, tel
```

conté la resta d'atributs de la classe. Si n'hi ha més d'un cal separar-los amb comes. Aquesta línia pot no tenir contingut (però la paraula clau i els dos punts

han d'aparèixer).

6.2.2. Associació

Una associació està composta per cinc línies que han d'aparèixer seguides. Un exemple d'una associació:

```

Associació: ASS_ALU_ASS
Classes: ALUMNE, ASSIGNATURA
Multiplicitat: *, *
Navegació: -
Tipus: ASS

```

- La primera de les línies d'una associació (paraula clau *Associació*)

```
Associació: ASS_ALU_ASS
```

conté el nom de l'associació. Aquesta línia no pot contenir cap coma (,).

- La segona línia (paraula clau *Classes*)

```
Classes: ALUMNE, ASSIGNATURA
```

conté les dues classes dels extrems de l'associació. El contingut ha de tenir dos noms de classes separats per una coma. Les dues classes han d'estar definides abans de l'associació.

L'ordre de les classes és important, la segona classe és la que en el dibuix conté el dibuix que representa l'associació. Per exemple:

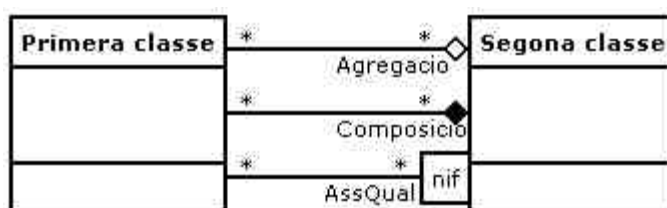


Figura 6.1: Exemple de l'ordre de les classes en la definició d'una solució

En tots tres cassos, *Primera classe* seria la que s'hauria d'escriure primer i *Segona classe* la que s'hauria d'escriure segona. En cas d'una associació normal, no importa l'ordre (sempre i quan es posin correctament les multiplicitats i la navegació).

- La tercera línia (paraula clau *Multiplicitat*)

Multiplicitat: *, *

conté les multiplicitats de l'associació. El contingut ha de contenir les dues multiplicitats separades per una coma (la coma és obligatòria si es defineix alguna de les multiplicitats). La primera multiplicitat correspon a la multiplicitat del costat de la primera classe i la segona multiplicitat a la del costat de la segona classe. Els possibles valors de les multiplicitats han de ser una de les següents:

- 0..1
- 1
- *
- 1..*
- n..m
- multiplicitat buida: en aquest cas significa que l'associació no té, ni pot tenir multiplicitat en el costat indicat.

Aquesta línia pot no tenir contingut (però la paraula clau i els dos punts han d'aparèixer). En aquest cas es considera que les dues multiplicitats són buides.

- La quarta línia (clau *Navegacio*)

Navegacio: -

defineix la navegació de l'associació. Els valors possibles de la navegació són:

- >: indica que la navegació va de la primera classe indicada a la segona.
- <: indica que la navegació va de la segona classe indicada a la primera.
- Qualsevol altre valor (inclòs que estigui buit), tot i que es recomana – (un guió): la navegació és bidireccional.

- La última de les línies (clau *Tipus*)

Tipus: ASS

indica el tipus de l'associació. I si l'associació pot ser qualificada, també conté els atributs de l'associació a darrera del tipus. Exemple de una associació qualificable:

Tipus: ASQ, numero

Els tipus possibles són els definits en l'arxiu de configuració de l'editor:

- ASS: associació normal. Necessita multiplicitat definida per les dues classes.
- AGR: agregació. Necessita multiplicitat definida per les dues classes.
- COM: composició. Necessita multiplicitat definida per les dues classes.
- ASQ: associació qualificada. Necessita multiplicitat definida per les dues classes.
- GEN: generalització. No ha de tenir multiplicitat.
- DEP: dependència. No ha de tenir multiplicitat.

El corrector no comprova si es compleixen les condicions de multiplicitat segons el tipus, només estan aquí a mode d'informació del que fa l'editor.

6.2.3. Classe associació

Una classe associació està composta per tres línies que han d'aparèixer seguides. Un exemple d'una classe associació:

```
Classe_associacio: NOTES
Atributs: nota1c, nota2c
Associacio: ASS_ALU_ASS
```

- La primera de les línies d'una classe associació (paraula clau *Classe associació*)

```
Classe_associacio: NOTES
```

conté el nom de l'associació. Aquesta línia no pot contenir cap coma (,). El separador entre les dues paraules que formen la clau de la línia pot ser espai en blanc, tabulacions, símbols de subratllat (_) o qualsevol combinació d'aquests caràcters.

- La segona línia (clau *Atributs*)

```
Atributs: nota1c, nota2c
```

conté un atribut o una llista d'atributs separats per comes. Aquesta línia pot no tenir contingut (però la paraula clau i els dos punts han d'aparèixer).

- La última de les línies (clau *Associacio*)

Associacio: ASS_ALU_ASS

té el nom de l'associació de la qual depèn. La línia no pot contenir comes i l'associació ha d'estar definida abans de la classe associació.

6.3. Exemple d'un problema

Les parts en color només serveixen per a distingir el contingut.

```

39
<E>
Volem guardar la informació de les assignatures que s'han matriculat
els alumnes en un curs acadèmic. Per això disposem de la següent
informació:
- Per cada alumne ens cal guardar informació referent al seu número de
matrícula (nummat), el seu dni (dni), el nom complert(nom)(cog1)(cog2)
i el seu telèfon(tel).
- De cada assignatura ens cal guardar el seu codi (codi), el seu nom
(nom_assig), els crèdits (credits), el curs (curs) i semestre
(semestre).
- També i per cada assignatura que un alumne es matricula ens cal
guardar la nota de la primera i segona convocatòria (notalc)(nota2c)
que han obtingut els alumnes que hi estan matriculats.

El sistema ha de permetre saber la informació dels alumnes, de les
assignatures i les notes que ha obtingut cada alumne en les
assignatures en que està matriculat.
</E>

<SOLUCIO1>
Classe: ALUMNE
Id: nummat
Atributs: dni, nom, cog1, cog2, tel

Classe: ASSIGNATURA
id: codi
Atributs: nom_assig, credits, curs, semestre

Associacio: ASS_ALU_ASS
Classes: ALUMNE, ASSIGNATURA
Multiplicitat: *, *
Navegacio: -
Tipus: ASS

Classe_associacio: NOTES
Atributs: notalc, nota2c
Associacio: ASS_ALU_ASS
</SOLUCIO1>

```

L'enunciat és el text que hi ha entre <E> i </E> i la única solució és el que està entre <SOLUCIO1> i </SOLUCIO1>. El diagrama de classes que correspon a aquesta solució és:

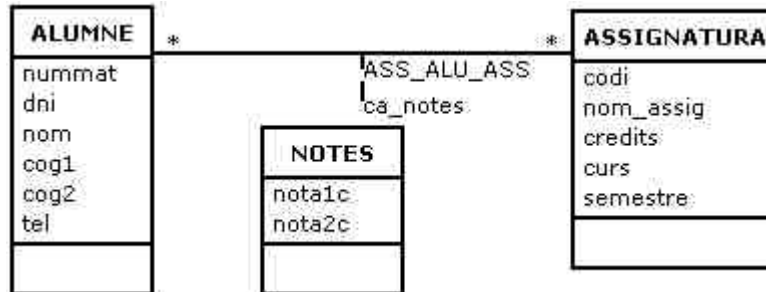


Figura 6.2: Diagrama de classes corresponent a l'exemple d'una solució

6.4. Interpretació d'una solució

Una vegada el professor ha escrit un problema cal que el pugi a l'ACME. Com que és possible que hi hagi algun error, abans de ser guardat al repositori es passa per un corrector que verifica que sigui correcte. Per a fer això s'utilitza un parser propi que a partir de l'entrada com a una cadena de caràcters crea una estructura de dades que el corrector pot usar. Per als errors, usa la tècnica pànic (si hi ha un error no continua amb el que segueix) i indica l'error trobat, i no es pot usar el retorn per a corregir. El seu funcionament bàsic és:

Eliminació de caràcters ignorats i línies sense contingut.
Divisió de la cadena de caràcters del pas anterior per els caràcter de salt de línia.
Mentre hi hagi línies i no hi hagi error
 Dividir la línia
 Si hi ha errors, plegar.
 Altrament, si la línia correspon a l'inici d'una classe.
 Llegir la classe.
 Si té errors, plegar.
 Altrament, afegir-la a l'estructura.
 Altrament si la línia correspon a l'inici d'una associació
 Llegir l'associació
 Si té errors, plegar.
 Altrament, afegir-la a l'estructura.
 Altrament si la línia correspon a l'inicia d'una classe associació
 Llegir la classe associació.
 Si té errors, plegar.

```
Altrament, afegir-la a l'estructura.  
FiMentre  
Si hi ha error retornar-lo.  
Altrament retornar l'estructura creada.
```

A continuació un diagrama de col·laboració del parser implementat per a interpretar una solució. En la numeració vermella, està indicat en lletres les funcions que es poden cridar en qualsevol ordre, els guions senyalen crides consecutives a la mateixa funció.

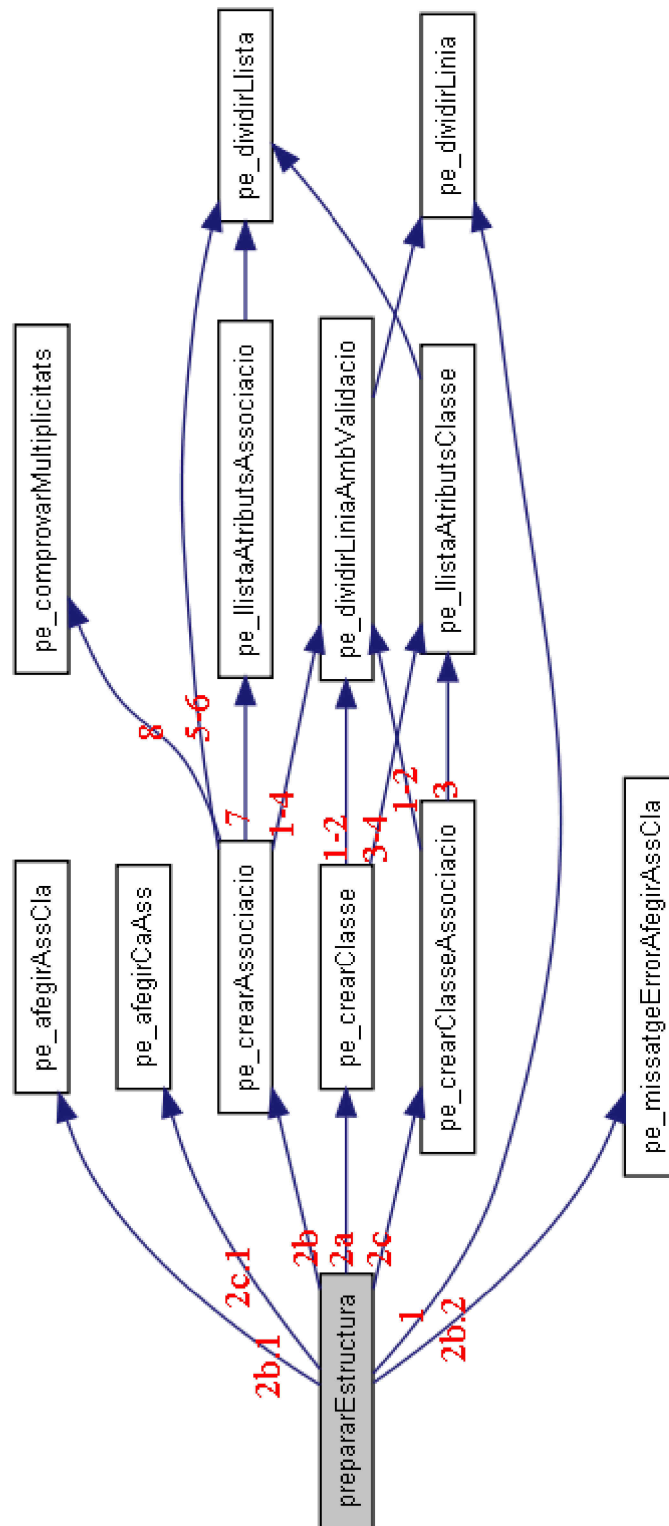


Figura 6.3: Diagrama de col·laboració del parser d'una solució.

Al final d'aquest procés si les solucions entrades pel professor són sintàcticament correctes es puja el problema al repositori de l'ACME. Si es detecten errors s'informa al

professor d'aquests errors i no es puja al repositori. En el següent apartat es mostren tots els possibles errors que es poden donar.

6.5. Errors en llegir un fitxer d'un problema

En interpretar un problema es poden donar el següents errors:

- **“Un dels enunciats comença abans de la fi d'un enunciat anterior”**: Com diu l'error, a dintre del text d'un enunciat, hi ha el començament d'un altre enunciat (<E>).
- **“No he trobat la marca de fi d'enunciat </E>”**: falta alguna etiqueta de tancar enunciat.
- **“No he trobat cap enunciat en aquest problema”**: no s'ha definit cap enunciat en el problema.
- **“La solució *número_solució* comença abans de la fi de la solució anterior.”**: la solució amb el número indicat comença a dintre del bloc de text de la solució anterior a aquesta.
- **“No he trobat la marca de fi de solució </SOLUCIO_>”** (_ és un número): la solució indicada, té l'etiqueta d'obertura però no la de tancament.
- **“No he trobat cap solució en aquest problema”**: no s'ha definit cap solució en el problema.
- **“La solució *número_solució* té atributs diferents a la primera solució correcte.”**: la solució amb el número indicat té un número d'atributs diferent a la primera solució correcte, també pot ser que sigui la primera solució correcte la que tingui els atributs mal definits.
- **“Està prohibit l'ús de la cadena de caràcters “###%##” en una solució.”**: una de les solucions conté la cadena de caràcters “###%##”. Aquesta cadena no es pot usar perquè té una funció fonamental en guardar la solució d'un alumne a la base de dades de l'ACME.

A part d'aquests errors, també pot ser que hi hagi altres errors en el text d'una solució.

6.5.1. Errors en interpretar una solució

Per corregir, primer es crea una estructura a partir d'una solució. Durant aquest procés, es poden donar alguns errors deguts a que alguna solució no està correctament estructurada. Els errors són:

- **“Una línia ha de tenir un sol signe ":" (dos punts).”**: una línia que hauria de ser la primera d'una definició d'una classe, associació o classe associació té més d'un signe de dos punts.
- **“El nom d'un element no pot ser buit.”**: com diu el text de l'error, el nom d'una classe, associació o classe associació no pot ser buit.
- **“No s'esperava una llista en el contingut. S'ha obtingut "text_obtingut"”**: en la línia d'inici d'una definició d'una classe, associació o classe associació, no hi pot haver cap coma.
- **“S'ha repetit el nom "nom_element". Els noms han de ser únics.”**: el nom d'una classe, una associació o una classe associació ha de ser únic dintre una solució. Es dona aquest error quan es repeteix un nom.
- **“S'esperava un inici de classe, associació o classe associació. S'ha trobat la paraula clau *paraula_clau*.”**: com diu l'error, s'esperava un inici de la definició d'una classe, una associació o una classe associació, però la paraula clau trobada no és correcta. Aquest error també pot aparèixer, si en la definició d'un problema, s'escriu el principi d'una solució en la mateixa línia que l'etiqueta d'obertura de la solució.
- **“Error en la classe *nom_classe*.”**: s'ha trobat una línia d'inici de classe però en voler obtenir les dues línies següents amb la informació dels identificadors i els atributs, la paraula clau d'alguna d'aquestes línies és incorrecta. També es pot donar si alguna d'aquestes línies conté més d'un signe de dos punts. També es pot donar aquest error si falta alguna línia de la definició de la classe o la última

línia de la classe està, en el problema, en la mateixa línia que l'etiqueta de tancar una solució.

• **“Error en l'associació *nom_associació*.”**: s'ha trobat una línia d'inici d'associació, però alguna de les següents línies té algun error:

- Alguna paraula clau de les línies de les classes, multiplicitats, navegació o tipus no és correcta.
- Alguna de les línies que contén les classes, multiplicitats, navegació o tipus, té més d'un signe de dos punts.
- No hi ha exactament dues classes definides.
- Hi ha valors de multiplicitat no vàlids, o n'hi ha un de definit però no hi ha la coma.
- No té tipus definit.
- Falta alguna línia de la definició de l'associació.
- La última línia de l'associació està, en el problema, en la mateixa línia que l'etiqueta de tancar una solució.

• **“Error en afegir l'associació "*nom_associació*" com a associació depenent de la classe "*nom_classe*". Assegurat que la classe "*nom_classe*" estigui definida abans que l'associació "*nom_associació*".”**: una de les classes d'una associació no existeix, pot ser degut a que tingui el nom mal escrit, o que la classe no estigui definida abans que l'associació.

• **“Error en la classe associació *nom_classe_associació*.”**: s'ha trobat una línia d'inici de classe associació però alguna de les línies següents no és correcte:

- La paraula clau de la llista d'atributs o de l'associació no és correcta.
- Alguna de les línies que contén els atributs o l'associació, té més d'un signe de dos punts.
- L'associació definida és buida.

- Falta alguna línia de la definició de la classe associació.
- La última línia de la classe associació està, en el problema, en la mateixa línia que l'etiqueta de tancar una solució.
- **“Error en afegir la classe associació "*nom_classe_associació*" com a classe associació dependent de l'associació "*nom_associació*". Assegurat que l'associació "*nom_associació*" estigui definida abans de la classe associació "*nom_classe_associació*" i que sigui l'única classe associació que depèn d'aquesta associació.”**: l'associació de la qual depèn una classe associació no existeix. Això pot ser degut a que el nom de l'associació estigui mal escrit o que l'associació no estigui definida abans que la classe associació. També pot ser que hi hagi alguna altra classe associació que depèn d'aquesta associació.

A continuació un exemple del que es mostra en cas que hi hagi un error en el problema.

```
ERROR: No he trobat la marca de fi de solució </SOLUCIO1>
```

```
Aquest fitxer no conté un exercici d'avaluació continuada!!!
```

Figura 6.4: Exemple d'un error en pujar un problema a l'ACME.

7. CORRECTOR DE DIAGRAMES DE CLASSES

El corrector és una eina que permet comparar diagrames de classe i indicar el semblants que són. També indica on hi ha els errors.

7.1. Entrada i sortida del corrector

El corrector d'entrada rep com entrada una solució correcte definida en el problema i la solució de l'alumne. Retorna un número entre 0 i 10 que indica el semblant que és la solució de l'alumne a la del professor i si no és equivalent, també retorna els errors que té l'alumne.

A la base de dades de l'ACME es guarda la solució enviada i si és correcte. Si no és correcte, també guarda el missatge amb els errors trobats.

7.2. Correcció

Per a comparar dues solucions s'ha fet un corrector que les compara indicat les classes, relacions i classes associació que tenen errors. Aquest corrector s'utilitzarà per comparar la/les solució/ons correctes del professor i amb la solució enviada per l'alumne.

Aquest corrector es pot configurar per tal que en comparar relacions tingui en compte la multiplicitat, la navegació o faci una comparació laxa en el tipus de relació. És a dir, es pot indicar si s'han de tenir en compte que les multiplicitats de les relacions sigui iguals (per defecte es té en compte), si la navegació sigui exacte (per defecte no es té en compte) i que la comparació de tipus d'associació sigui laxa (és a dir que en cas que ho sigui, si s'esperava una agregació o composició i l'alumne a posat una associació normal es consideri correcte, per defecte s'utilitza correcció laxa).

Tot i que encara no s'utilitzi, el corrector està pensat per a donar una nota: indicant el correcta que és una solució mitjançant una nota entre 0 i 10. És a dir com més alta és la nota més igual a la solució del professor és la solució de l'alumne.

7.2.1. Càlcul de la puntuació

La puntuació d'un problema la conformen dos apartats: classes i classes associacions (C) i associacions (A) de forma que $T = C + A$ sent C i A valors configurables pel professor. Un alumne tindrà el total del valor C o A si totes les classes i classes associació i associacions respectivament de la seva solució són correctes. En funció del número total de classes, classes associació i associacions de la solució del professor que més s'assembla a la del alumne es calcula el valor de cada classe, classe associació i associació. El valor a restar per cada error comés ve multiplicat per un coeficient penalitzador que pot ser diferent per a classes i classes associació i associacions. Els valors d'aquests coeficients penalitzadors poden ser configurables i es representen per cp_c i cp_a .

Així doncs, en cas d'errors la puntuació que es tindrà en compte serà l'obtinguda a partir de la fórmula:

$$\text{NOTA} = 10 - (v_c * ne_c * cp_c) - (v_a * ne_a * cp_a) \text{ i } \text{NOTA} \geq 0$$

Amb:

$v_c = C /$ suma de les classes i classes associació de la solució del professor (valor d'una classe o classe associació correcta)

$v_a = A /$ quantitat d'associacions de la solució del professor (valor d'una associació correcta)

$ne_c =$ número de classes i classes associació errònies de la solució de l'alumne

$ne_a =$ número d'associacions errònies de la solució de l'alumne

La interfície per entrar C, A, cp_c i cp_a s'està desenvolupant en un altre Projecte Final de Carrera, de manera que en aquest Projecte Final de Carrera aquests valors es troben en la codificació del corrector. Actualment s'utilitzen els valor per defecte 6, 4 ,1 i 1 respectivament.

S'ha fet el corrector que indiqui una nota ja que en un futur es pugui usar els

problemes de diagrames de classes per a un examen i també per a poder comparar diferents correccions d'una mateixa solució de l'alumne (el professor pot definir varies solucions).

7.2.2. Estratègia de correcció

Per a corregir, es comparen totes les combinacions de unir cada classe d'una solució correcte amb una classe de la solució enviada per l'alumne i el mateix amb les relacions cercant la combinació que té una major semblança o la combinació exacte. Per tal d'anar més ràpid i descartar solucions pitjors a la millor trobada s'utilitza la nota parcial calculada fins al moment i en el cas de trobar una combinació que és exacte no es calcula cap altre. També, només es fan les combinacions de les associacions sobre les associacions d'una parella de classes combinades en lloc de sobre totes les associacions.

A nivell d'implementació està fet amb dues funcions: una recursiva "indirecta" (la recursió ve a través de la segona funció) i l'altre recursiva. Aquestes funcions s'anomenen `cor_classeAjuntar` i `cor_associacioAjuntar` respectivament. La primera calcula les combinacions de les classes i la segona les de les associacions. Totes dues funcions treballen sobre el grup de classes o associacions que té menys elements i combinen aquests amb les de l'altre solució.

El funcionament de `cor_classeAjuntar` és:

```
Si no hi ha mes classes a combinar
  Comptar les penalitzacions dels elements no combinats
  Calcular la nota i guardar la millor de l'anterior i actual
Altrament
  Obtenir la classe que toca de la solució que en té menys a combinar
  Marcar la classe de la solució que en té menys com a combinada
  Mentre hi hagi classes en la solució que en té més i no s'hagi
trobat una combinació exacta fer
  Obtenir la següent classe de la solució que en té més
  Si no està combinada
    Comparar les dues classes i afegir els errors
    Marcar la classe de la solució que en té més com a combinada
    Si la nota és millor que la millor nota
      Combinar les associacions de les dues classes
(cor_associacionsAjuntar)
```

```

Fsi
  Desfer els errors afegits
  Desmarcar la classe de la solució que en té més com a
combinada.
  Fsi
  Fmentre
    Desmarcar la classe de la solució que en té menys com a combinada
Fsi

```

El funcionament de la funció `cor_associacionsAjuntar` és:

```

Si no hi ha mes associacions a combinar
  Combinar la següent classe de la solució que en té menys
(cor_classeAjuntar)
Altrament
  Obtenir l'associació que toca de la classe ajuntada en l'ultim
cor_classeAjuntar que té menys associacions
  Si l'associació té les classes dels dos extrems combinades
    Marcar l'associació de la classe ajuntada en l'ultim
cor_classeAjuntar que té menys associacions com a combinada
    Mentre hi hagi associacions en el grup de la classe ajuntada en
l'ultim cor_classeAjuntar que té més associacions i no s'hagi trobat
uns combinació exacta fer
      Si l'associació no està combinada i té les classes dels dos
extrems combinades
        Comparar les associacions i possibles classes associació i
afegir els errors
        Marcar l'associació de la classe ajuntada en l'ultim
cor_classeAjuntar que té més associacions com a combinada
        Si la nota és millor que la millor nota
          Combinar la següent associació de la classe ajuntada en
l'ultim cor_classeAjuntar que té menys associacions
(cor_associacioAjuntar)
        Fsi
        Desfer els errors afegits
        Desmarcar la classe de la solució que en té més com a
combinada.
        Fsi
        Fmentre
          Desmarcar l'associació de la classe ajuntada en l'ultim
cor_classeAjuntar que té menys associacions com a combinada
          Si no s'ha pogut fer cap combinació
            Combinar la següent classe de la solució que en té menys
(cor_classeAjuntar)
            Fsi
            Altrament
              Combinar la següent associació de la classe ajuntada en l'ultim
cor_classeAjuntar que té menys associacions (cor_associacioAjuntar)
              Fsi
            Fsi
          Fsi
        Fsi

```

7.2.3. Missatges d'error indicats pel corrector

Els missatges que donà el corrector per a facilitar a l'alumne que sàpiga els llocs on té errors són:

- **“A la solució enviada li falten nom_tipus.”**: nom_tipus pot ser “Classes”, “Associacions”, “Classes associació” o “atributs”. Indica que a la solució de l'alumne respecte la del professor que s'assembla més li falten classes, associacions, classes associació o atributs respectivament.
- **“A la solució enviada li sobren nom_tipus.”**: nom_tipus pot ser “Classes”, “Associacions” o “Classes associació”. Indica que a la solució de l'alumne respecte la del professor que s'assembla més li sobren classes, associacions o classes associació respectivament.
- **“La classe "nom_classe" es incorrecte; repassa-la.”**: la classe amb nom nom_classe de l'alumne té algun error en els atributs, és a dir, li falten o sobren atributs o té atributs identificadors incorrectes.
- **“L'associació "nom_associació" es incorrecte; repassa-la.”**: l'associació amb nom nom_associació té algun error, és a dir, és d'un tipus incorrecte, té la multiplicitat incorrecta, té la navegació malament, alguna de les classe entre les que està és incorrecta o té atributs qualificadors incorrectes (o no en té definits i s'esperaven o en té i no s'esperaven).
- **“La classe associació "nom_classe_associació" es incorrecte; repassa-la.”**: la classe associació amb nom nom_classe_associació té algun error, és a dir que té atributs incorrectes (ja sigui que li falten o li sobren) o està en l'associació equivocada.

En la següent captura s'il·lustra com veu un l'alumne els errors comesos.

El resultat de la correcció és:

A la solució enviada li falten Classes.
A la solució enviada li falten Associacions.
A la solució enviada li falten Classes associació.
A la solució enviada li falten atributs.

Continuar

Figura 7.1: Exemple de la informació que dona en detectar que una solució és incorrecta.

8. INTEGRACIÓ AMB LA PLATAFORMA ACME

Una vegada desenvolupat l'editor i el corrector calia integrar tota la feina feta a l'ACME, per tal que aquest reconegués aquest tipus de problema. Per aixó ha calgut definir un nou tipus d'exercici. La plataforma ACME necessita que es defineixi una classe en php que s'anomeni "tipus_xx" on xx és el número de tipus que toca al tipus d'exercici (en aquest cas 39, és a dir la classe s'anomena "tipus_39". En aquesta classe es defineixen una serie de mètodes:

- Mètodes per a visualitzar una solució enviada. Aquest mètode serveix per a poder mostrar una solució i per visualitzar els errors comesos. L'editor que es carrega es configura per tal que no es pugui editar. En la següent imatge es pot veure el seu en mostrar els errors comesos en el llistat de solucions enviades.

Data	Solució Enviada	Resultat
28/5/2009 19:36:05	1 Veure la solució parcial 1.1 Incorrecte 2 Veure la solució parcial 1.2 Incorrecte	Incorrecte
28/5/2009 19:37:04	1 Veure la solució parcial 2.1 Incorrecte A la solució enviada li falten atributs. 2 La classe "Classe 1" es incorrecte; repassa-la. Incorrecte La classe "Classe 2" es incorrecte; repassa-la. Incorrecte	Incorrecte
28/5/2009 20:07:50	1 Veure la solució parcial 3.1 Incorrecte 2 Veure la solució parcial 3.2 Incorrecte	Incorrecte
28/5/2009 20:11:41	1 Veure la solució parcial 4.1 Incorrecte A la solució enviada li falten atributs. 2 La classe "Classe 1" es incorrecte; repassa-la. Incorrecte La classe associació "Classe 4" es incorrecte; repassa-la. Incorrecte	Incorrecte
28/5/2009 20:12:03	1 Veure la solució parcial 5.1 Incorrecte A la solució enviada li falten atributs. 2 La classe associació "Classe 4" es incorrecte; repassa-la. Incorrecte	Incorrecte
28/5/2009 20:12:29	Veure la solució 6	Correcte

Figura 8.1: Llista amb les solucions enviades d'un problema.

La següent imatge es pot veure el l'editor que mostra el mètode amb una solució mostrada.

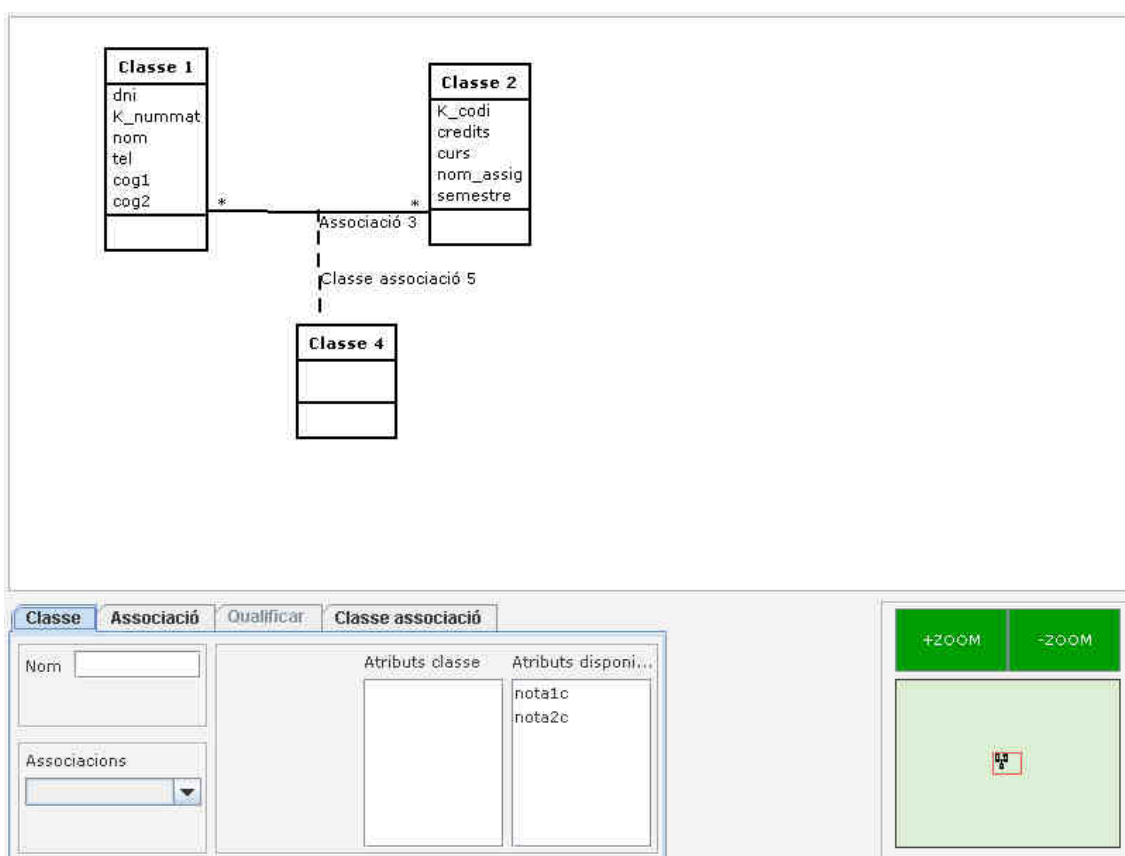


Figura 8.2: Editor en mode visualització amb la solució parcial 5.1 de la imatge anterior.

8. Integració amb la plataforma ACME

- Crear el contingut de les pàgines on l'alumne o el professor introduiran la solució (usant l'editor de diagrames de classes). Aquests mètodes creen la part de la pàgina web on és mostra l'editor per tal que s'introdueixi una solució. També s'encarreguen de passar a l'editor l'última solució enviada i de carregar els atributs que ha de tenir l'editor.

Volem guardar la informació referent als carrers de les diferents poblacions de Catalunya per això disposem de la següent informació :

- Catalunya està formada per un conjunt de Comarques. De cada comarca ens interessa saber les seves sigles identificatives (sigles_id), el seu nom (nom_comarca) i la seva superfície (superficie).
- A cada Comarca hi ha moltes poblacions . De cada població ens interessa saber el seu codi identificatiu (codi_pobl), el seu nom (nom_pobl) i el nombre d'habitants que té (habitants).
- De cada població volem tenir catalogats els diferents carrers. Cada carrer l'identifiquem pel seu nom (nom_carrer) ... Tingueu en compte que un mateix nom de carrer pot estar en poblacions diferents, ara bé a cada població no hi ha dos carrers que es diguin igual. També ens interessa saber la llargada del carrer (llargada), la seva amplada (amplada) i el número de edificis/cases (edificis)

El sistema que dissenyem us ha de permetre saber :

- Totes les comarques de Catalunya i les seves corresponents poblacions
- Tots els carrers de cada ciutat i donat un nom de carrer saber totes les poblacions que el tenen

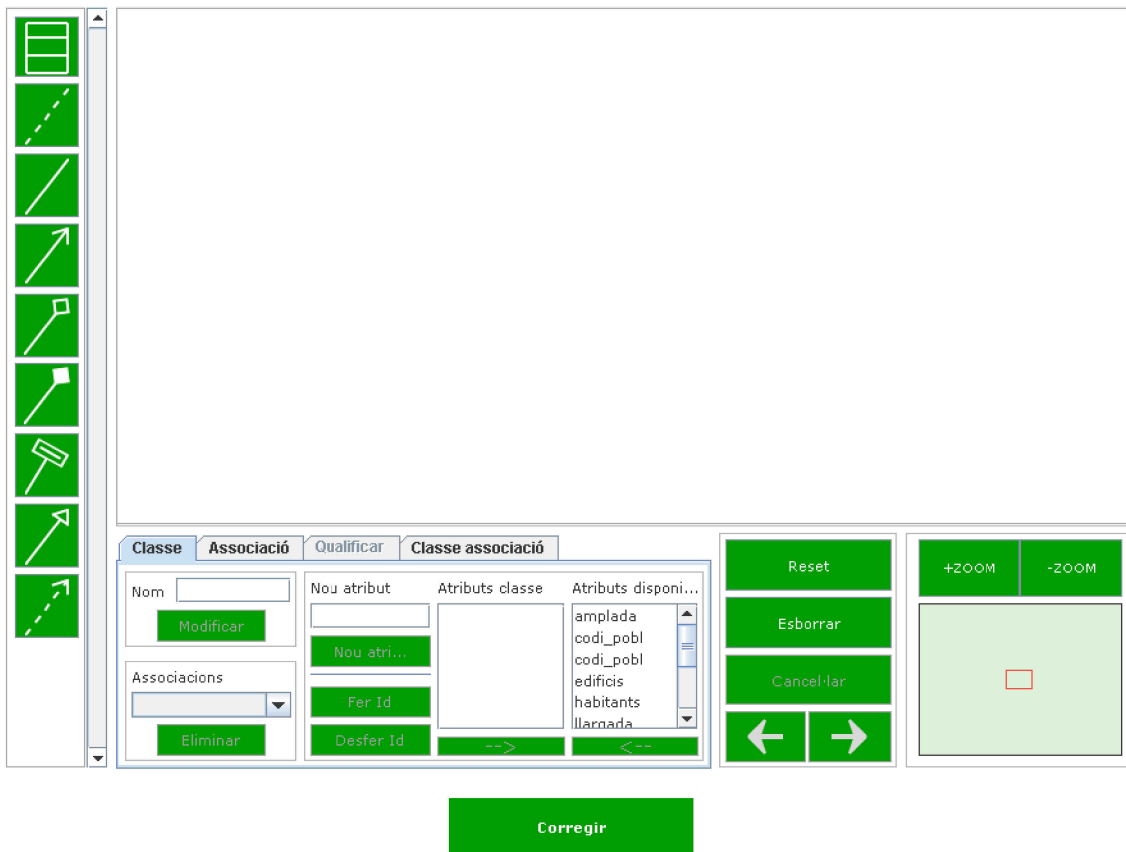


Figura 8.3: Editor preparat per tal que l'alumne dibuixi la solució de l'enunciat adjunt.

- Un mètode per a validar que un fitxer d'un problema sigui vàlid. Aquest mètode s'utilitza quan el professor vol afegir un problema a l'ACME. Comprova que l'estructura del problema i l'estructura de les solucions que conté siguin correctes.

ERROR: No he trobat la marca de fi de solució </SOLUCIO1>

Aquest fitxer no conté un exercici d'avaluació continuada!!!

Figura 8.4: Error mostrat en validar un problema a l'hora de pujar-lo a l'ACME.

- Un mètode per a corregir les solucions enviades. Aquest mètode compara la solució que ha introduït l'alumne amb totes les solucions mostrant, si en el problema no hi ha cap solució equivalent els errors amb la comparació millor. Per a corregir, es fa de la següent forma:

Comparar la solució a corregir amb totes les solucions del problema.
Si no es troba cap solució equivalent, es mostren les diferències amb la solució més semblant.
Altrament si alguna solució del problema és igual a la solució a corregir, es considera correcte.

Aquesta classe usa funcions definides en el corrector de diagrames de classes.

8.1. Paràmetres que es poden passar a l'editor

Els paràmetres són la informació que es passa a un applet de Java des de la pàgina web que el conté. Són utilitzats per a passar informació a l'applet o configurar-lo.

L'editor de diagrames té els següents paràmetres, tots opcionals.

- **pEdicio:** valor booleà (cert o fals) que indica si s'ha d'usar l'editor per a modificar o crear un diagrama o per a visualitzar-lo. Només es considerarà que es pot editar un diagrama en cas que el seu valor sigui *true* (sense tenir en compte majúscules o minúscules). Si no està definit, es considera que no es pot editar.
- **pProfessor:** valor booleà (cert o fals) que indica si es poden usar certes funcions de l'editor per a modificar els atributs o canviar el diagrama sense

necessitat d'haver de demanar informació a l'ACME. Només es considerarà que es poden usar les funcions afectades en cas que el seu valor sigui *true* (sense tenir en compte majúscules o minúscules). Si no està definit, es considera que no es poden usar aquestes funcions.

- **pTraduccio**: cadena de caràcters amb la traducció del text mostrat en l'editor. Les diferents parts han d'estar separades per “&%&”. L'ordre de les traduccions ha de ser el mateix amb el que està declarat l'atribut privat anomenat català de la classe Traducció de l'editor. En cas que no estigui definit s'usaran les traduccions en català de l'esmentat atribut. En cas que aquest paràmetre li faltin elements, s'usaran els definits a l'atribut català de Traducció per els que faltin.
- **pAtribut**: cadena de caràcters amb la llista d'atributs que ha de tenir l'editor en carregar-se. Els atributs han d'estar separats per comes i els espais i/o tabulacions de davant o darrera de cada atribut seran eliminats. Si no es defineix, l'editor no tindrà atributs predefinits.
- **pDibuix**: cadena de caràcters amb la cadena necessària per a reconstruir un dibuix. Es recomana substituir les símbols & per & i “ per " (en aquest ordre) per no tenir problemes amb l'html. Si no es defineix no s'utilitzarà cap dibuix en carregar.
- **pColorFons**: defineix el color de fons usat per a canviar el color de fons de l'editor i que quedi més integrat en la pàgina que el conté. Per definir el color, cal usar el format demanat per el mètode decode de la classe java.awt.Color de Java (en definitiva, usar el prefix 0x, 0X o # i el color en hexadecimal de 6 caràcters). Si el paràmetre no es defineix o es produeix un error en interpretar el color s'utilitzarà el color per defecte de Java.

8.2. Funcions per a interactuar amb l'editor des d'una pàgina web

Per a obtenir informació de l'editor, s'han creat unes funcions que es poden cridar des de Javascript i poder enviar aquesta informació al servidor ACME.

- **getCadenaCorrecio:** retorna la cadena de correcció del dibuix que hi ha actualment a l'editor.
- **getCadenaDibuix:** retorna la cadena de caràcters que serveix per a poder tornar a restaurar el diagrama.
- **setCadenaDibuix:** permet canviar des de Javascript la cadena de dibuix (canvia el diagrama actual). Té un paràmetre que és la cadena de dibuix. Només pot ser usat si els paràmetres pEdicio i pProfessor estan definits a cert.
- **getAtributs:** torna una llista amb els atributs actuals de l'editor (inclosos els definits per l'usuari) separats per comes. Només retornarà valor si l'atribut pProfessor està definit a cert.
- **setAtributs:** canvia els atributs predefinits de l'editor per els passats en el paràmetre de la funció. El paràmetre de la funció ha de tenir la mateixa estructura que el valor de l'atribut pAtribut de l'editor. L'ús d'aquesta funció elimina tots els atributs que en el moment d'usar-la estiguin se'n utilitzats en el diagrama. Només pot ser usat si els paràmetres pEdicio i pProfessor estan definits a cert.

9. CONCLUSIONS

En el inici del projecte es van proposar una sèrie d'objectius, els quals s'han assolit satisfactòriament.

S'ha desenvolupat un editor molt potent que et permet totes les funcionalitats d'un necessàries per a crear un diagrama de classes per al disseny conceptual de bases de dades.

Com a experiència personal, m'agradaria comentar que s'han complert tots els objectius proposats inicialment, amb el qual em sento satisfet dels resultats obtinguts. Portant a terme aquest projecte, he hagut d'utilitzar diferents eines i treballar amb llenguatges de programació, aspecte que m'ha permès ampliar els meus coneixements. En el transcurs del projecte he hagut de posar en pràctica molts dels coneixements adquirits durant la carrera, com per exemple: realitzar un anàlisi i disseny, especificació de requeriments...etc.

Finalment m'agradaria agrair al dos tutors la col·laboració i ajuda que m'han prestat.

10. PROPOSTES DE FUTURES MILLORES

Tot i que l'editor té totes les funcionalitats demanades, sempre hi ha detalls que es poden millorar, entre ells:

- A partir del dibuix fet per l'usuari, poder guardar-lo de forma independent a la plataforma ACME com una imatge (format png, per exemple).
- Canviar la zona de dibuix principal més fàcilment: canviar la zona on es treballa de l'editor arrossegant-ne la imatge.
- Canviar els botons de l'editor que permeten fer accions sobre la zona de dibuix principal per botons de dos estats i així saber en tot moment què s'està fent.
- Poder passar el fitxer de configuració com un paràmetre per no haver de modificar el fitxer de configuració inclòs en l'editor.

11. BIBLIOGRAFIA / WEBGRAFIA

- CEBALLOS, FCO. JAVIER; Java 2 Curso de programación. 2a ed. Madrid: Ra-Ma Editorial, 2002
- Java 6, API (només usada la part comuna amb Java 5):
 - <http://java.sun.com/javase/6/docs/api/>
- PHP Hipertext Preprocessor, manual:
 - www.php.net
- Wikipedia (definicions)
 - <http://es.wikipedia.org/>
 - <http://en.wikipedia.org/>

12. FIGURES

Figura 1.1: Exemple d'una classe.....	7
Figura 1.2: Exemple d'una associació.....	8
Figura 1.3: Exemple d'una agregació.....	8
Figura 1.4: Exemple d'una composició.....	9
Figura 1.5: Exemple de multiplicitat i navegació.....	9
Figura 1.6: Exemple 1 d'una associació qualificada.....	10
Figura 1.7: Exemple 2 d'una associació qualificada.....	10
Figura 1.8: Exemple d'una classe associació.....	10
Figura 1.9: Exemple de generalització.....	11
Figura 1.10: Exemple disseny conceptual de bases de dades.....	12
Figura 2.1: Esquema dels objectius.....	13
Figura 3.1: Esquema de les diferents etapes.....	15
Figura 5.1: Diagrama de casos d'us de l'editor.....	24
Figura 5.2: Diagrama de classes parcial de l'editor (falten dependències). Les línia puntejades separen els diferents grups de classes.....	30
Figura 5.3: Classe IntAux.....	31
Figura 5.4: Classe XMLauxiliar.....	32
Figura 5.5: Dependències de la classe Parellalmatges.....	33
Figura 5.6: Classe ImagePack.....	34
Figura 5.7: Classe abstracte Element.....	36
Figura 5.8: Dependències de la classe Classe.....	38

Figura 5.9: Atributs de la classe Classe.....	38
Figura 5.10: Mètodes de la classe Classe.....	40
Figura 5.11: Exemple d'una classe sense seleccionar i la mateixa classe seleccionada..	41
Figura 5.12: Herència de la classe abstracte AssociacioBase.....	42
Figura 5.13: Dependències de la classe AssociacioDosClasses.....	43
Figura 5.14: Exemple per il·lustrar la diferencia entre primera i segona classe en una associació.....	43
Figura 5.15: Atributs de la classe AssociacioDosClasses.....	43
Figura 5.16: Mètodes de la classe AssociacioDosClasses.....	45
Figura 5.17: Exemple d'una agregació sense seleccionar i la mateixa agregació seleccionada.....	46
Figura 5.18: Herència de la classe ClasseAssociacio.....	47
Figura 5.19: Exemple d'una classe associació sense seleccionar i la mateixa classe associació seleccionada.....	48
Figura 5.20: Dependències, atributs i classes privades de la classe ConjuntElements..	49
Figura 5.21: Mètodes de la classe ConjuntElements.....	51
Figura 5.22: Mètodes de la interfície Atribuible.....	52
Figura 5.23: Dependències de la classe VectorAtributs.....	53
Figura 5.24: Dependències de la classe Atribut.....	54
Figura 5.25: Dependències de la classe InfoAssociacio.....	55
Figura 5.26: Atributs de la classe InfoAssociacio.....	55
Figura 5.27: Mètodes de la classe InfoAssociacio.....	56
Figura 5.28: Classe AppletClasses amb totes les dependències, atributs i mètodes.....	58

Figura 5.29: Classe Principal amb les seves dependències.....	59
Figura 5.30: Atributs de la classe Principal, part 1.....	61
Figura 5.31: Atributs de la classe Principal, part 2.....	62
Figura 5.32: Atributs de la classe Principal, part 3.....	63
Figura 5.33: Mètodes de la classe Principal, part 1.....	64
Figura 5.34: Mètodes de la classe Principal, part 2.....	65
Figura 5.35: Classes privades de la classe Principal.....	65
Figura 5.36: Classe abstracte AbstractCanvas.....	66
Figura 5.37: Dependències de la classe MyCanvas.....	68
Figura 5.38: Dependències i classes privades de PetitCanvas.....	70
Figura 5.39: Classe Traduccio.....	72
Figura 5.40: Dependències de la classe ConfigAssociacions.....	73
Figura 5.41: Exemple de les connexions d'una associació: “Primera classe” és la classe de la primera connexió i “Segona classe” és la de la segona.....	80
Figura 5.42: Interfície gràfica de l'editor.....	81
Figura 5.43: Gràfic d'explicació del funcionament de l'editor.....	82
Figura 5.44: Pestanya de modificació de classes.....	84
Figura 5.45: Pestanya d'edició d'associacions.....	85
Figura 5.46: Exemple de les connexions d'una associació: “Primera classe” és la classe de la primera classe i “Segona classe” és la de la classe.....	85
Figura 5.47: Pestanya de qualificació.....	86
Figura 5.48: Pestanya d'edició de les pestanyes.....	86
Figura 5.49: Editor en mode de visualització.....	87

Figura 5.50: Exemple de l'editor.....	89
Figura 6.1: Exemple de l'ordre de les classes en la definició d'una solució.....	94
Figura 6.2: Diagrama de classes corresponent a l'exemple d'una solució.....	98
Figura 6.3: Diagrama de col·laboració del parser d'una solució.....	100
Figura 6.4: Exemple d'un error en pujar un problema a l'ACME.....	104
Figura 7.1: Exemple de la informació que dóna en detectar que una solució és incorrecta.....	110
Figura 8.1: Llista amb les solucions enviades d'un problema.....	111
Figura 8.2: Editor en mode visualització amb la solució parcial 5.1 de la imatge anterior.	112
Figura 8.3: Editor preparat per tal que l'alumne dibuixi la solució de l'enunciat adjunt.	113
Figura 8.4: Error mostrat en validar un problema a l'hora de pujar-lo a l'ACME.....	114