Lab 4: Anti Tower Defence

Oskar Mothander dit06omr

Alan Mendez Larsson dit06mln

Lärare: Johan Eliasson

Handledare: Johan Granberg
Tor Sterner-Johansson
Thomas Johansson
Daniel Henriksson

Innehåll

1. Problemspecifikation

I den här laborationen ska vi skapa ett så kallat *Tower Defence* spel där du som spelare skickar ut trupper som ska nå ett mål utsatt på kartan. Kartan kan ha flera startpunkter och ett mål. Det ska finnas minst två olika trupper du kan skicka ut. Datorn ska i sin tur sätt upp torn som skjuter på dina trupper. Om du blir av med alla dina trupper och inte har några pengar kvar har du förlorat, om du istället får igenom ett visst antal trupper har du vunnit. Olika trupper kan ha olika egenskaper som hjälper eller stjälper dom att komma i mål.

Krav som ställs på spelet är att koden är skrivet enligt Suns java kod konvention, i objektorienterad stil och allmänt bra programmeringsstil utan onödiga kompliceringar.

Orginallaborationsspecifikationen återfinns på:

http://www.cs.umu.se/kurser/5DV085/HT08/labbar/lab4.html

2. Åtkomst & användarhandledning

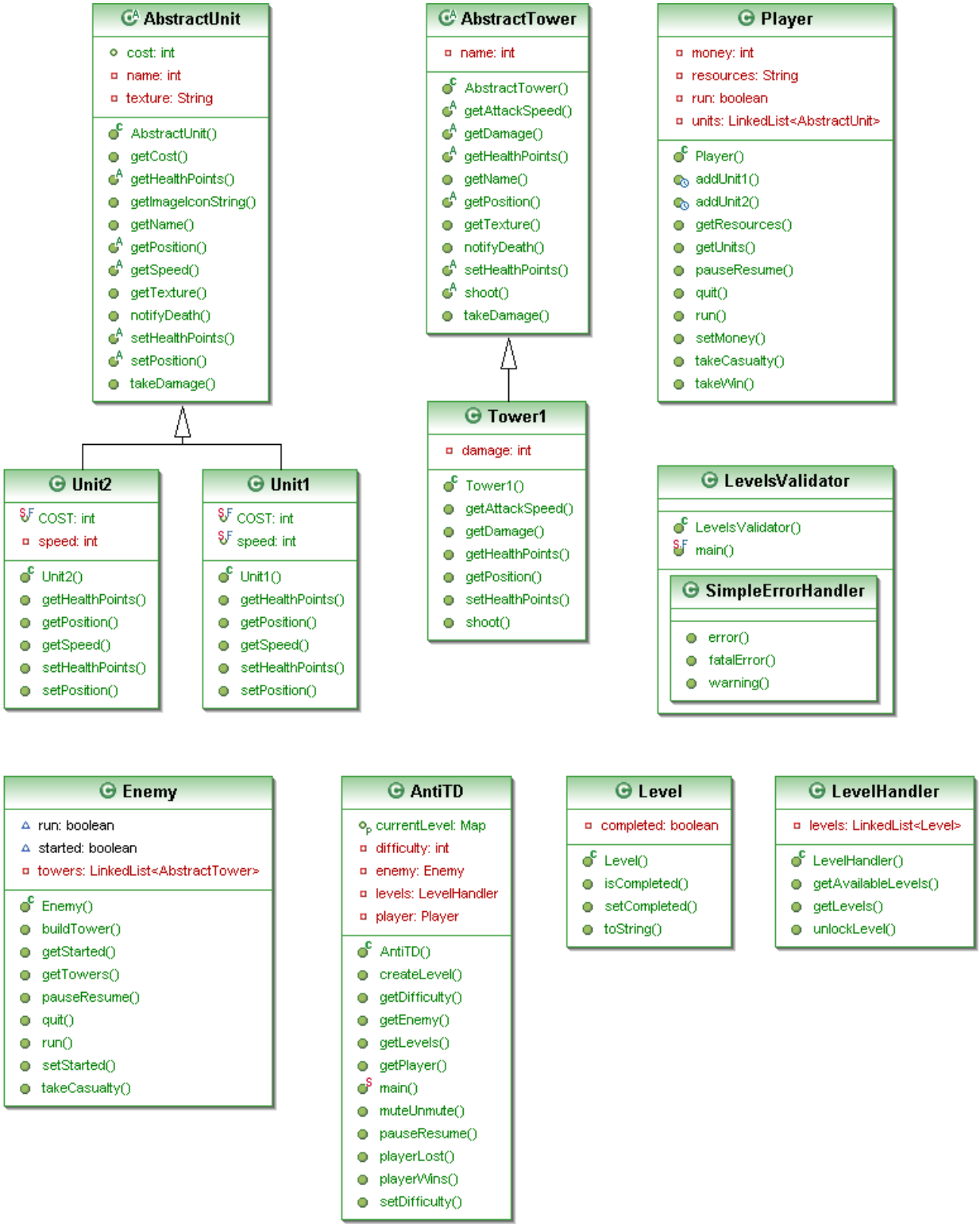
Källkoden till programmet ligger i katalogen:	`dit06mln/edu/apjava/lab4/src/`
En komplierad version finns i katalogen:	`dit06mln/edu/apjava/lab4/bin/`
Denna rapport hittas i katalogen:	`dit06mln/edu/apjava/lab4/report/`
JavaDoc-sidorna för programmet ligger i katalogen:	`dit06mln/edu/apjava/lab4/doc/`

3. Systembeskrivning

Spelet är uppbyggt i ett rutnät där trupper förflyttar sig en ruta i taget som i schack. Runt om alla vägar finns platser för torn. Spelet kan spelasi olika svårighetsgrader: *Easy*, *Medium* och *Hard*. På enklaste nivån (Easy) placerar datorn ut torn på strategiskt dåliga positioner där den endast kan skjuta på t.ex. två rutor. Svårighetsgraden på rutorna bestäms av banan du kör. På Hard placerar datorn ut tornen på de mest strategiska positionerna såsom vid korsningar där de kan skjuta på upp till fem rutor. När datorn har placerat ut torn på alla de bästa positionerna fortsätter den att placera ut på resterande i ordningen Hard -> Medium -> Easy och tvärt om för Easy.

3.1 Paket: Engine

Engine-paketet är grunden till spelet och det paketet som står för de viktigaste klasserna för spelandet. I Engine finns hanteringen av vilka kartor som finns, spelaren, fienden och dess AI, samt alla enheter och torn. Se figur 1.



Figur 1: Package Engine

3.1.1 AntiTD

Huvudklassen till spelet. Startar upp alla trådar utom till Enemy som startas när du börjar skicka ut trupper.

3.1.2 Level

Enkel klass som beskriver en bana, om den är klarad eller ej och filnamn.

3.1.3 LevelHandler

Läser in alla banor och sparar dom som Level.

3.1.4 LevelValidator

LevelValidator är den klass som validerar levels.xml med hjälp av levels.xsd som är ett Schema. Den huvudsakliga uppgiften är att kolla om sökvägen till de olika kartorna är giltiga. En annan av dess uppgifter är att skapa Level-objekt.

3.1.5 Player

Avspeglar dig som spelare. Håller reda på alla units, dina pengar om spelet är pausat. Klassen flyttar även dina unit frammåt kontinuerligt på kartan. Kör en egen tråd.

3.1.6 AbstractUnit

Abstrakt klass som beskriver units. Varje unit sparas som denna typ.

3.1.7 Unit1

Även av AbstractUnit. Kostar 20 och har 100 HP (Health Points).

3.1.8 Unit2

Även av AbstractUnit. Kostar 60 och har 250 HP

3.1.9 Enemy

Avspeglar fienden. Lagrar och bygger alla torn och ser till så de skjuter. Kör en egen tråd.

3.1.10 AbstractTower

Abstrakt klass som beskriver torn. Varje torn lagras av denna typ. Finns utrymme för implementering av attackering av torn.

3.1.11 Tower1

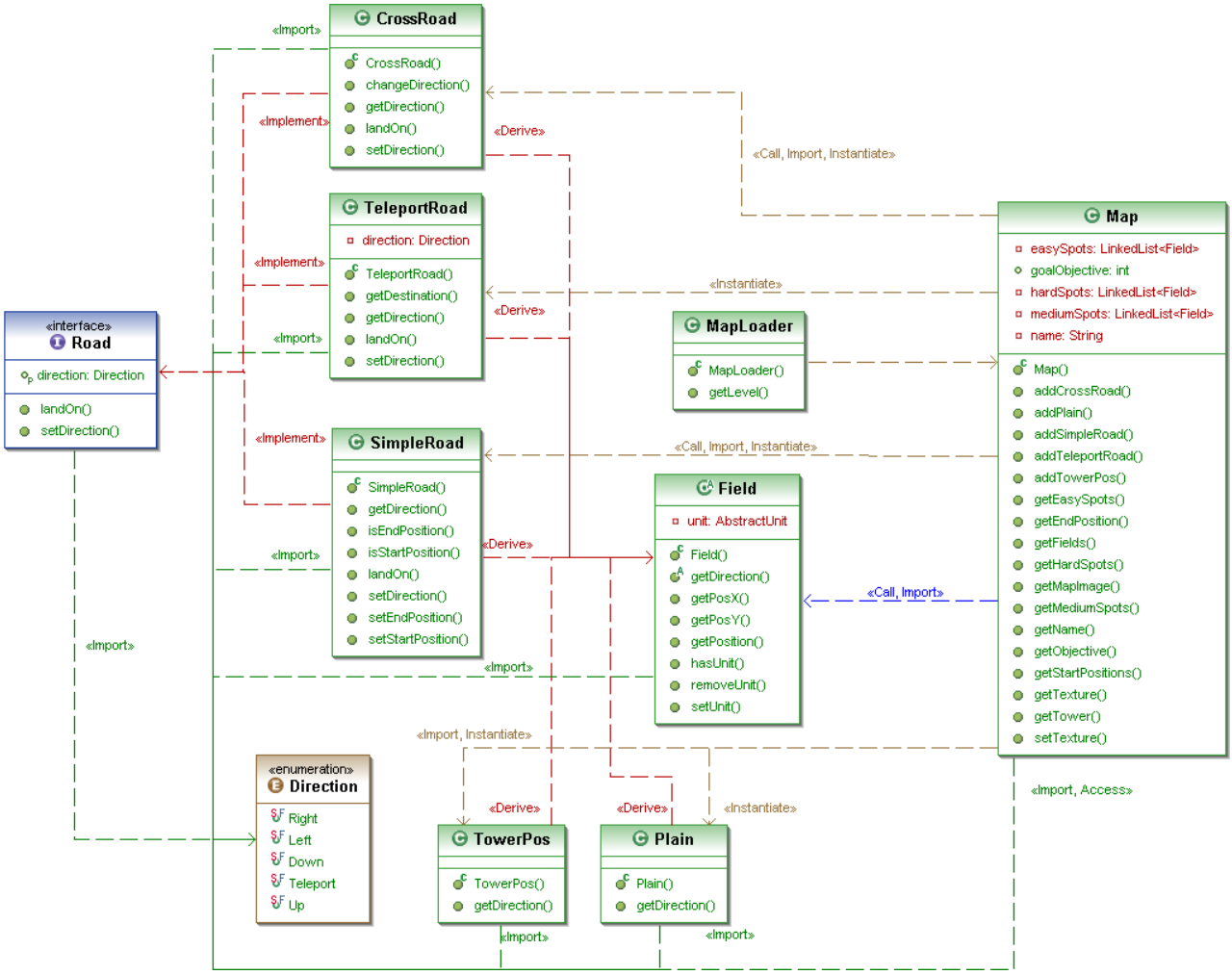
Även av AbstractTower. Har en attack hastighet på 500 och slår 20 HP i skada.

3.1.12 Arrow

Klassen är en superklass tänkt att hålla olika texturer till pilar beroende på vilken riktning pilen ska åka i, t.ex. snett uppåt vänster etc.

3.2 Paket: Map

Map-paketet innehåller alla de nödvändiga klasser för att skapa en karta och alla dess fält. Här finns den karta som sedan skickas till GUI:et. Se figur 2.



Figur 2: Package Map

3.2.1 MapLoader

Läser banfiler och skapar Map objekt av dom.

3.2.2 MapValidator

MapValidator är den klass som kollar om en karta är rätt skriven i det sättet att den har rätt syntax. MapValidator kollar inte att alla vägar är rätt eller om alla rutor är placerade korrekt, utan den kollar enbart om de olika fälten är rätt skrivna och vilken textur kartan använder sig utav. Likt LevelValidator är den bifogade Schema:t till klassen (level.xsd) ej korrekt skrivet och därför implementeras inte MapValidator i MapLoader där den ska utföras innan skapandet av kartan.

3.2.3 Map

En representation av en karta med alla dess fält och egenskaper. Består av Field-objekt och BufferedImage för alla typer av olika fälttexturer. Map bestämmer även om en Plain ska få default-texturen, "plain.gif", eller den dekorerade texturen, "decoration.gif".

3.2.4 Direction

En Enum som beskriver riktning på vägar, d.v.s. vilken riktning units ska gå i. Möjliga vägar är: Up, Down, Left, Right och Teleport där Teleport är för positioner på vägen man ska kunna teleportera på.

3.2.5 Field

Abstrakt klass där varje ruta på kartan representeras av ett Field.

3.2.6 Plain

Dekorativa rutor som gräs eller träd representeras som denna typ.

3.2.7 TowerPos

På dessa rutor kan torn placeras. Varje TowerPos har en tilldelad svårighetsgrad också.

3.2.8 Road

Ett interface som beskriver en väg. Har abstrakta metoder getDirection för att hämta rikning på väg och landOn för teleportvägar.

3.2.9 SimpleRoad

Vägen på kartan representeras av denna typ. Har en riktning (Direction), kan vara start eller slutpunkt. Eventuell teleporter kan implementeras.

3.2.10 CrossRoad

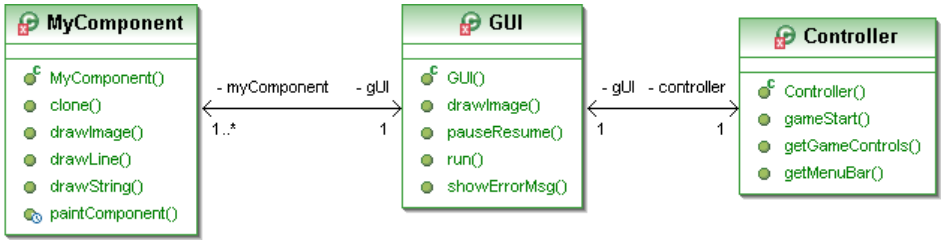
En vägpunkt representerar en korsning där du kan ta olika vägar. Har därför flera riktningar (Direction) och vilken som är för närvarande vald.

3.2.11 TeleportRoad

En väg vartifrån du kan teleporteras ifrån eller till. Har en destinationsposition.

3.3 Paket: Graphics

Graphics-paketet består av alla de klasser som använder sig utav Swing-paketet. Dessa klasser används för visandet av all grafik och kontroller. Se figur 3.



Figur 3: Package Graphics

3.3.1 GUI

Denna klass sköter all utritning av grafik. Uppdaterar kartan, torn, units och torn som skjuter 24 gånger per sekund. Kör en egen tråd. Använder sig utav MyComponent för att rita ut torn och units. Ritar ut BufferedImage's med storleken 40x40 pixels. Kan vara pausad.

3.3.2 MyComponent

Given modifierad klass som sköter inläsning och har metoder för utritning av BufferedImages (bl.a.).

3.3.3 Controller

Denna klass sköter alla Events i programmet som knapptryckningar och menymanövrering. Håller även menyer och visar popups.

5. Begränsningar

Units kan dela samma ruta. Priset som visas för varje Unit är målat på ikonen för knapparna, detta betyder att om man vill ändra priset på en Unit-typ så måste alla de tillhörande "unit*button.gif" ändras.

LevelValidator är den klass som ska validera levels.xml med hjälp av levels.xsd som är ett Schema. Vid skrivande stund (2009-01-12) är levels.xsd ej korrekt skriven och därför är inte LevelValidator implementerad i AntiTD. Dock är koden för klassen rätt skriven och om ett korrekt skrivet levels.xsd skapas kan den implementeras.

6. Problem & reflektioner

Mycket tid har ägnats åt efterforskning på hur man bäst gör olika saker som t.ex. ritar grafik, rörlig grafik. Hur man undviker trådproblem; *ConcurrentModificationException* har jag löst genom att klona listor innan de ritas ut samt döda units tas bort efter alla har gåtts igenom.

Oskar Mothander

Till en början las mkt tid på xml-valideringen som tyvärr prioriterades bort när tiden för redovisningen började närma sig. Klassen finns kvar än men implementationen i programmet togs bort. Tydligare visning av vad som skiljer sig mellan de olika Units hade varit fördelaktigt för användarvänlighet, såsom hälsa.

Alan Larsson