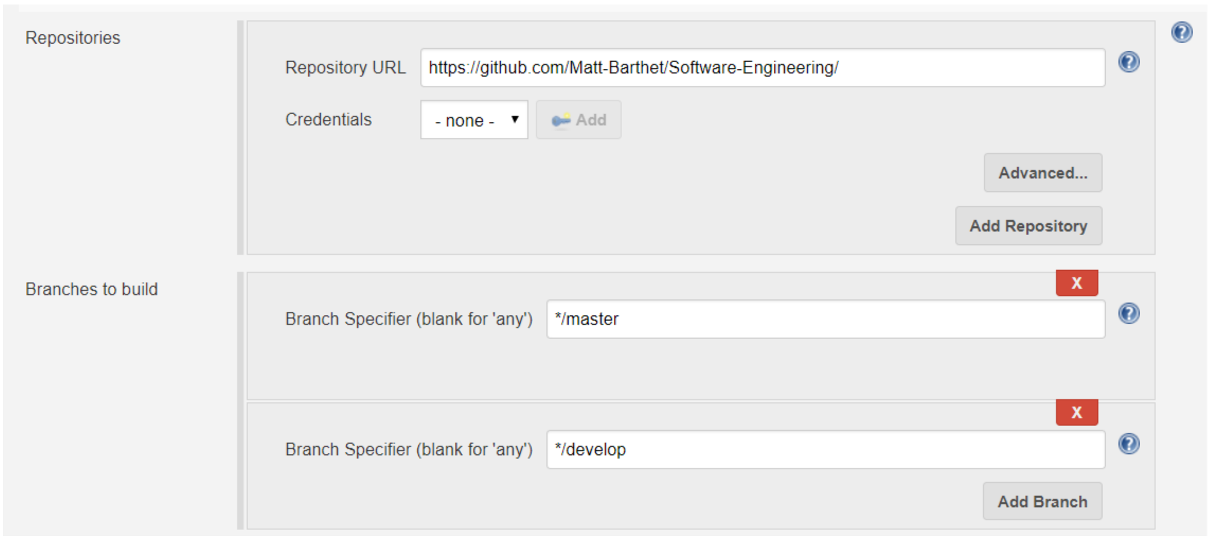
Final Unit Assignment

Collaborators : Matthew Barthet and Vitaly Volozhinov
Course : Bsc(Hons) Computer Science 2nd Year
Unit : CPS2002-SEM2-A-1718: Software Engineering

Contents:

Assignment Part 1 :

This part of the assignment was the initial set up of the Github repository and it's integration with the University Jenkins server .


Repositories

Repository URL: https://github.com/Matt-Barthet/Software-Engineering/

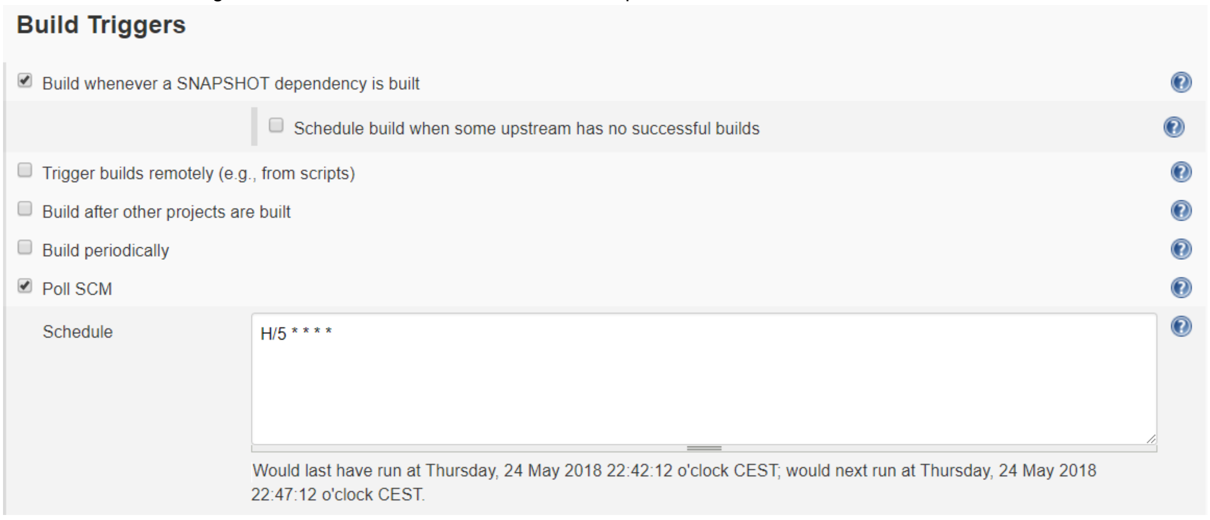
Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */master

Branch Specifier (blank for 'any'): */develop

Fig 1.0 Github linked to Jenkins Server with Specific branches to build .


Build Triggers

☑ Build whenever a SNAPSHOT dependency is built

☐ Schedule build when some upstream has no successful builds

☐ Trigger builds remotely (e.g., from scripts)

☐ Build after other projects are built

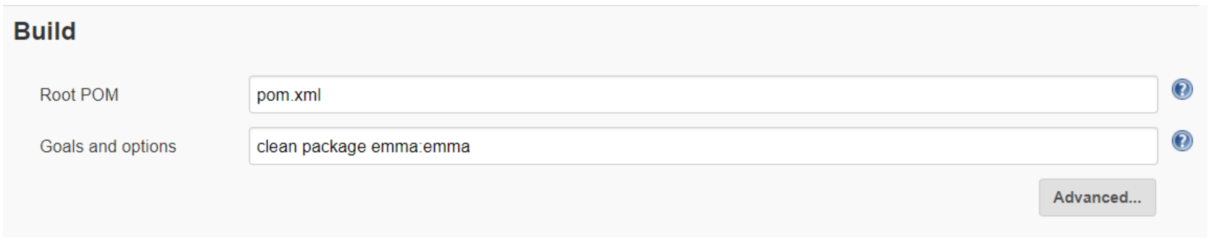
☐ Build periodically

☑ Poll SCM

Schedule: H/5 * * * *

Would last have run at Thursday, 24 May 2018 22:42:12 o'clock CEST; would next run at Thursday, 24 May 2018 22:47:12 o'clock CEST.

Fig 1.1 Build Triggers set up to build either on schedule or when a new update is pushed to Github


Build

Root POM: pom.xml

Goals and options: clean package emma:emma

Fig 1.2 Build set up to read the Pom.xml file for specifications on and how it's run using emma code coverage .

The Github Repo link : https://github.com/Matt-Barthet/Software-Engineering
The Jenkins link :https://jenkins-ict.research.um.edu.mt/job/Software-Engineering%20CPS2002%20-VM/
The Github was set up together with Git Flow which created extra branches for development purposes where the features have been updated to . Once a task was started a new feature was started and once ended it was merged to the develop branch . After a final product was

finished the develop branch created a release of the projects current state and added it to the master branch.

Assignment Part 2 :

In this part of the assignment the first iteration of the treasure map game was created , the main game loop consisted of the game asking the user how many people are playing and the amount of tiles are going to be used in the game . Restrictions had been put in as to have the map a certain amount of tiles for the amount of players in the game :

i. 2-4 players = 5 x 5
ii. 5-8 players = 8 x 8

The maximum map size was set to 50 and min player amount was set to 2 players as a person can’t play this game solo .

The map was made up out of 3 type of tiles : grass , water and one winning tile . An abstract tile class was created and those three tile types had inherited it . The map was generated according to the user input for the n amount of tiles and generated the map randomly with 10% water tiles . After it’s generated the winning tile overwrites any random tile in the game .
The player is spawned on a random tile which is made sure to be green . A player explores the map with commands : (Up, (Down, (Left, (Right there have been means to stop the player going out of the map , by setting the limits of the players movement not to go out of the map .When the player is first created a starting position is stored so that when he moves onto a water tile his current position is rolled back to it . The game waits for the all the players to enter their inputs before their positions are updated and outputted. Once a player moves and the tile they are currently on is uncovered to be a treasure they win the game and the game ends for all players.

As was just mentioned, the game waits for all the players to enter their inputs before positions are updated. Once the game verifies the input and updates the positions, an html file is generated for each player, which contains the treasure map for each player. This works by using J2HTML, an external java module which allows for the creation of HTML code using java syntax. The HTML is coded and generated by writing to an html file. Each file differs in which tiles are being displayed, since for each player only the tiles they have visited are set to render on the web page. The rendering is done by casting a background image on the tiles that have been uncovered by the player. A player icon is also rendered in the foreground of the tile the player is currently standing on.

Test driven development was used , by creating tests for each object and class after they have been completed to test the code’s functionality and that all their functions are working accordingly. The coverage metrics are shown below :

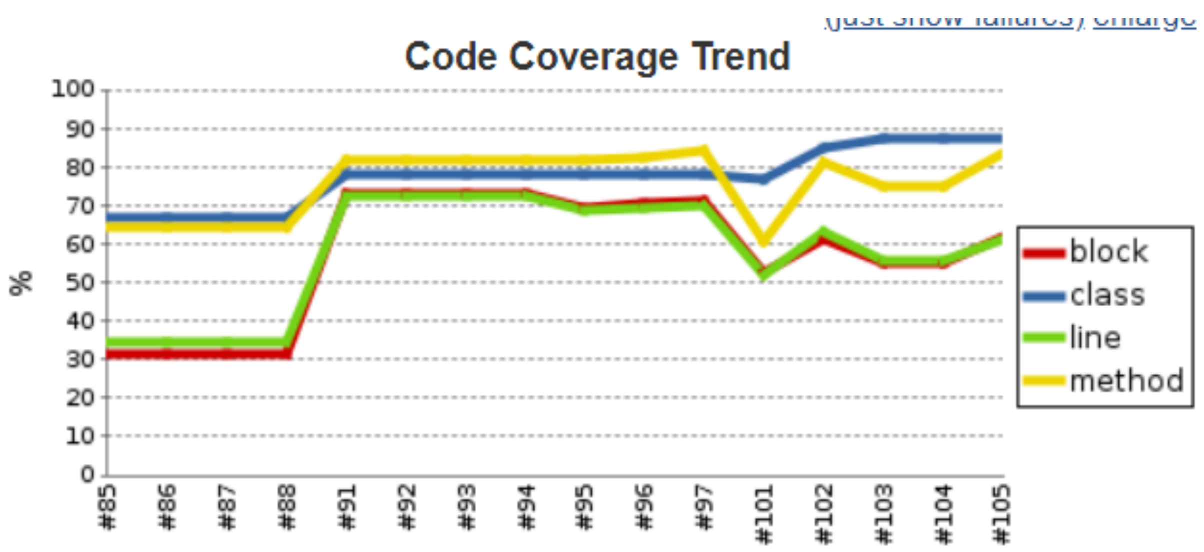


Fig 2.1: Code Coverage trend started late due to having different Java versions in POM.xml file and problem wasn't found.

The code coverage was up to 80% as most of the code was tested but except for the different tile classes which where just tested on the map so there wasn't a need .

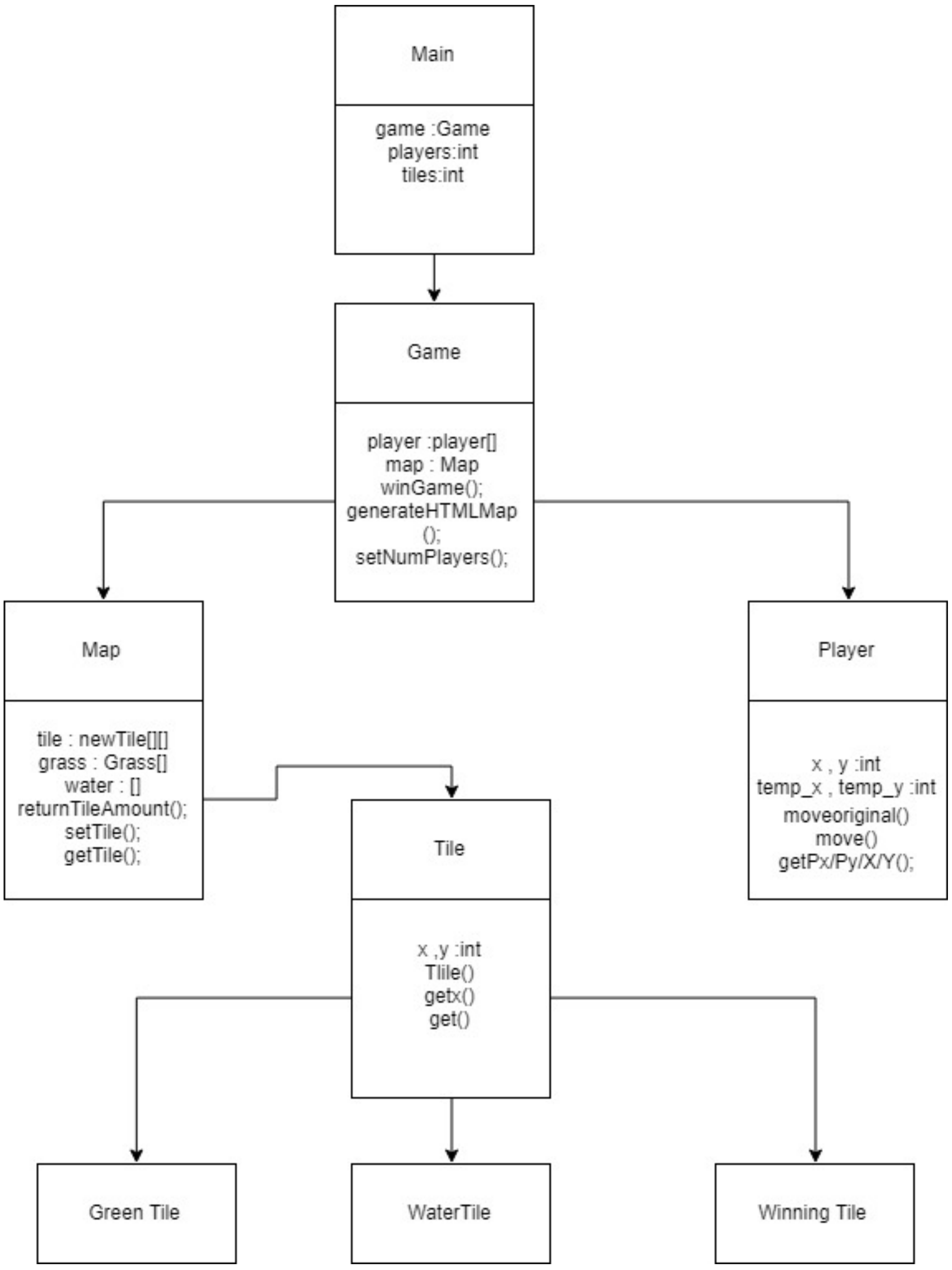


Fig 2.2 Class Diagram of above game implementation .

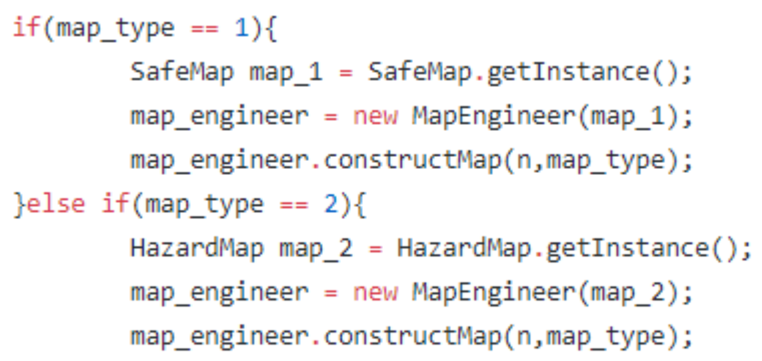
The code was then merged into the master branch and tagged accordingly .

Assignment Part 3 :

The requirements of the game have been changed as the customer wanted new things for the game . The customer wanted 3 new features using different design patterns to implement them .

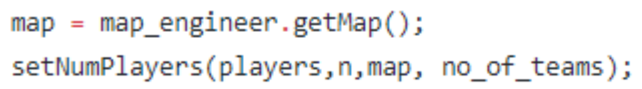
The customer wanted different map types , which would be a safe map with 10% water tiles and a hazard map which would have between 25% and 35% water tiles . The choice was given at the beginning of the game together with player amount and number of tiles . The map creation was done in the Game Class where the choices would create a safe map object which is used to create a specific map engineer , this map engineer would then construct the map with the tiles and the correct percentage of water tiles in the map according to either safe or hazardous. The design pattern used for this was the Builder design pattern where the Map engineer would construct a new map type (safe/hazard) using a map builder which would then create a general map that follows a map plan .The map engineer then returns the general map that was created (either safe or hazard and this general map is then used as the object around the game .

The customer wanted a single instance of the map to be used without creating other maps . so the singleton design pattern had been implemented on the safe and hazardous map , by calling instances of those maps once created and setting the constructor to private making it impossible to create more instances , so only one map is used throughout the entire game.



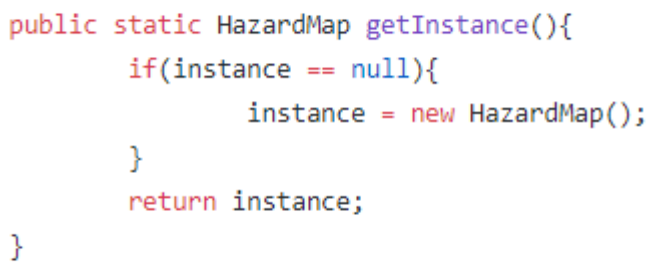
```
if(map_type == 1){
        SafeMap map_1 = SafeMap.getInstance();
        map_engineer = new MapEngineer(map_1);
        map_engineer.constructMap(n,map_type);
}else if(map_type == 2){
        HazardMap map_2 = HazardMap.getInstance();
        map_engineer = new MapEngineer(map_2);
        map_engineer.constructMap(n,map_type);
```

Fig 3.0 Shows how the map is created using the builder and singleton design pattern



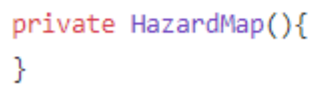
```
map = map_engineer.getMap();
setNumPlayers(players,n,map, no_of_teams);
```

Fig 3.1 Shows how the map is retrieved and is used in parameters of functions like the previous version .



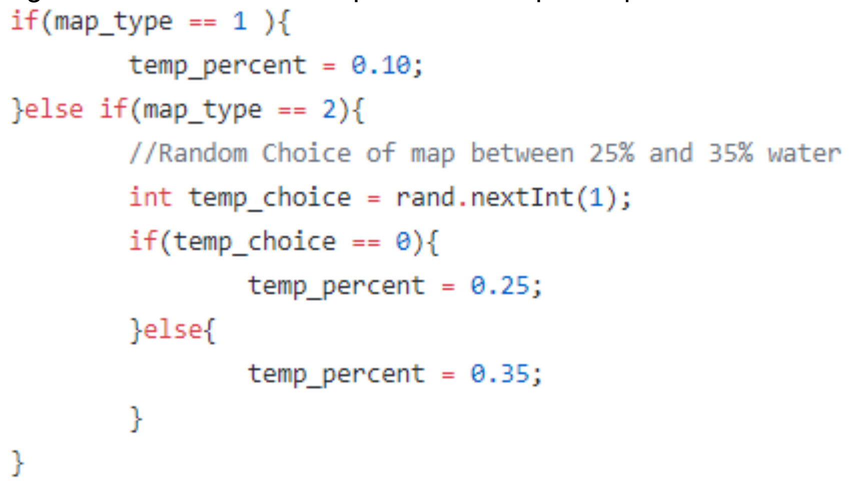
```
public static HazardMap getInstance(){
        if(instance == null){
                instance = new HazardMap();
        }
        return instance;
}
```

Fig 3.2 Showing how the calling of the map instance is done .



```java
private HazardMap(){
}
```

Fig 3.3 Private constructor prevents multiple maps created .



```java
if(map_type == 1 ){
        temp_percent = 0.10;
}else if(map_type == 2){
        //Random Choice of map between 25% and 35% water
        int temp_choice = rand.nextInt(1);
        if(temp_choice == 0){
                temp_percent = 0.25;
        }else{
                temp_percent = 0.35;
        }
}
```

Fig 3.4 Percentage of Water Tiles being chosen either the type is one (10%) or two where then a random number is generated to make the choice .

It was also required to implement a collaborative mode into the game. To implement this, every player is randomly assigned to a team and should be able to see all the tiles discovered by his/her teammates. The observer design pattern was chosen to abstract updating the team's uncovered tiles and notifying each player. An abstract observer class was created which was inherited by each player object, and the Score_Subject class was created which inherits from the abstract subject class. When a player was assigned to a team they were also attached to the team's score_subject object. Whenever a player moves from their position, they set a new state in the score_subject which notifies all the players attached to it. This in turn allows the other player's to update their uncovered lists so that they can see what their other team mates have uncovered.

The following class diagram reflects the changes made to the requirements:

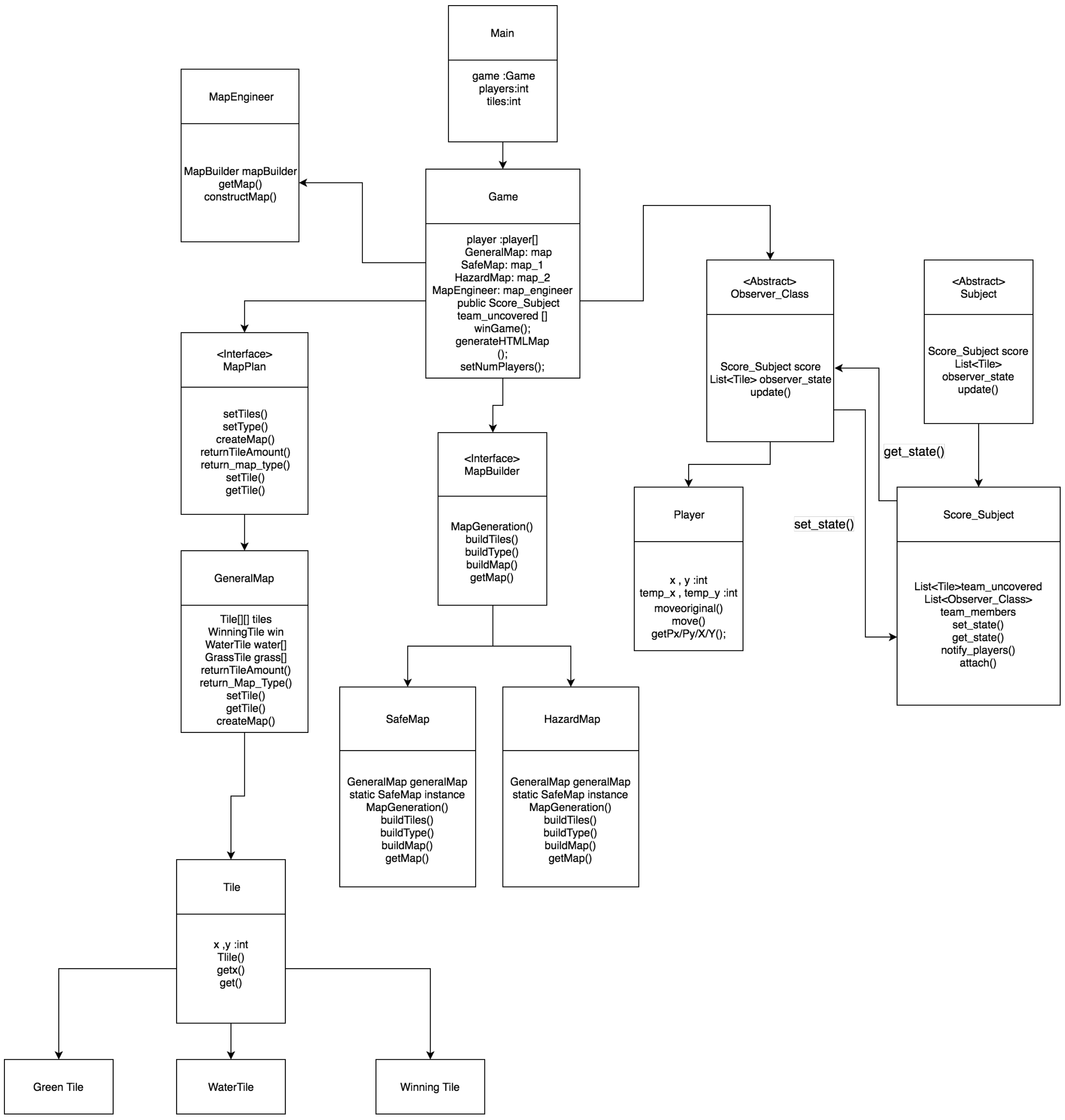


Fig 3.5 New Class Diagram for the updated map.

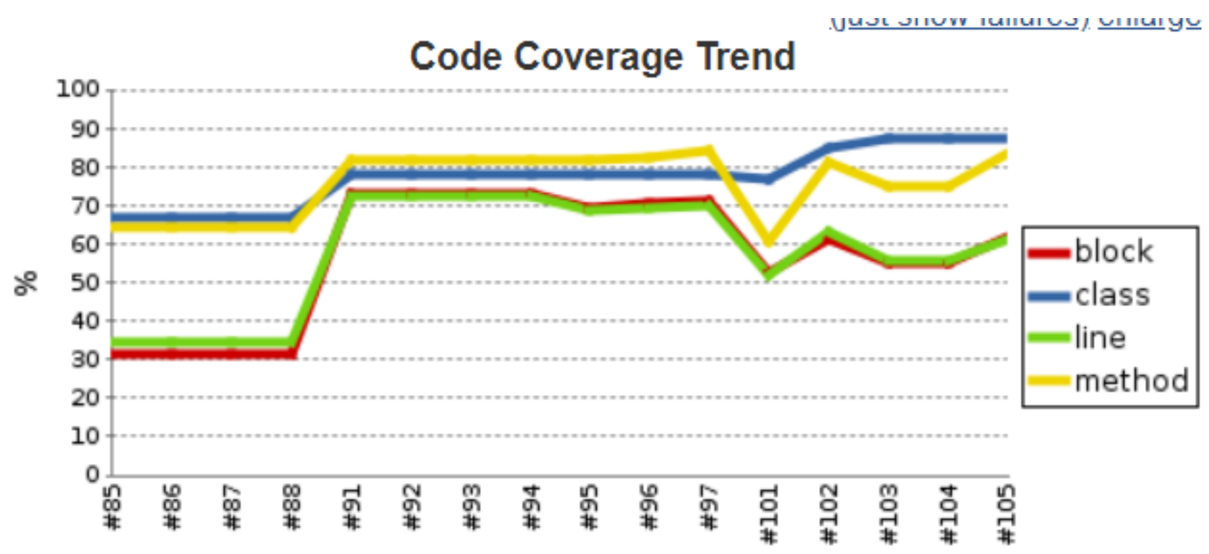


Fig 3.6 Final Coverage Trend of Part Assignment .

From 94 to 105 builds we can see that the code coverage has gone down up and down due to new code being added and not tested .The Methods and classes are mostly covered while the line and blocks aren't all covered . The code that wasn't fully covered was the abstract classes , the interface classes and the different tiles . These weren't covered as there wasn't a need to as their use cases was tested by other classes that have been using them . Such as the map builder would use multiple functions and having them fail would make the map builder fail , therefore the tests didn't need to be run on those functions .

Running :

```
mvn clean package emma:emma
java -cp target/game-1.0-SNAPSHOT-jar-with-dependencies.jar com.uom.game.Main
```

References :
Builder : https://www.geeksforgeeks.org/builder-design-pattern/amp/
Singleton: https://www.geeksforgeeks.org/singleton-design-pattern/

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as "the unacknowledged use, as one's own, of work of another person, whether or not such work has been published, and as may be further elaborated in Faculty or University guidelines" (University Assessment Regulations, 2009, Regulation 39 (b)(i), University of Malta).

I / We*, the undersigned, declare that the [assignment / Assigned Practical Task report / Final Year Project report] submitted is my / our* work, except where acknowledged and referenced.

I / We* understand that the penalties for committing a breach of the regulations include loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected, and will be given zero marks.

* Delete as appropriate.

(N. B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

Matthew Barthet	[illegible]
Student Name	Signature
Vitaly Volozhinov	[illegible]
Student Name	Signature
Student Name	Signature
Student Name	Signature

CPS2002	Software Engineering Assignment
Course Code	Title of work submitted
25/5/18	
Date	