**Software Development 2 Coursework** (SET11103)

**“Sky Wars”**



You should write a Java program to meet the specification shown below.

This assignment constitutes 100% of the module assessment.

**Problem Statement**

You can design your own game or use the sample game given below.

If you design your own game, your game should be a move-based game with a GUI.

Read through the sample game to get an idea about the type of game. Double check with the module leader that the game you are developing is OK.

**Sky Wars (Sample Game)**

Sky Wars consists of a Sky, a Master Space Ship and a number of Enemy Space Ships

**The Sky**

The sky can be thought of as a four by four grid. (The code for a grid and the basic functionality of a grid will be supplied on Moodle, you can use that code or use your own if you prefer).

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

When the game starts the Master Space Ship is placed in a random square.

**Master Space Ship**

**Moving**

When the game starts the master space ship is randomly allocated to a square in the sky. When the Master Space Ship moves, it can move to any of the neighbouring squares as shown below. The space ship cannot move out of the sky.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | Master Ship |  |  |
|  |  |  |  |
|  |  |  |  |

**The Master Space Ship can move to any of its neighbouring squares**

The square the Master Space Ship moves to is chosen at random from all the possible squares that it can move to.

**Enemy Space Ships**

**Moving**

There are three types of enemy space ship that can enter into the sky; a BattleStar, a BattleCruiser and a BattleShooter. The top left corner of the sky is a one directional intergalactic hole and it is through this hole that all enemy ships enter the sky (because the intergalactic hole is one directional nobody leaves the sky through the top left hand corner). Once in the sky the enemy ships move in exactly the same way as the Master Space Ship; i.e. they move randomly to any of the neighbouring squares.

Every time the Master Space Ship moves there is a one in three chance of an enemy space ship entering the sky, the type of enemy ship is completely random.

|  |  |  |  |
| --- | --- | --- | --- |
| Battle  Cruiser |  |  |  |
|  |  |  |  |
|  | Master Ship |  |  |
|  |  |  |  |

**Enemy ships always enter the sky through the one-directional intergalactic hole in the top left corner of the sky**

Although there are only three types of enemy ships, new types of enemy ship may be available in the future. You should take this into account when coding your solution.

**Conflict**

If the Master Space Ship moves into the same square as an enemy ship, the Master Space Ship destroys the enemy ship and the enemy ship is removed from the sky.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Battle  Star |  |  |
|  |  |  |  |
|  |  | Master  Ship |  |
|  |  |  |  |

**The Master Space Ship destroys a lone enemy ship**

If the Master Space Ship moves into the same square as two or more enemy ships, the enemy ships destroy the Master Space Ship and the game is over.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Battle  Cruiser |  |  |
|  |  |  |  |
| Battle  Star |  | Master  Ship |  |
|  |  |  |  |

**Two or more enemy ships destroy the Master Ship and the game is over**

**GUI**

Your game should be controlled from a Graphical User Interface (GUI). From the GUI you should be able to start the game, and make moves. Your GUI should also have a visual representation of the game which should be updated as the game status changes. **Note that your GUI must be connected to your application by one association. There should be no game logic in your GUI.**

**Testing**

You should provide a comprehensive JUnit test suite for the main classes in your program. If you are unsure about which classes to provide testing for you should speak with the module leader for clarification.

**Programming Technique not Taught During the Module**

Ten percent of the marks will be allocated to the correct use of any programming technique which was not taught during the module. This is your opportunity to demonstrate that you can learn and apply software development techniques on your own (This is what all employers look for when asking for a reference).

**Advanced Features**

Marks will be given for attempting any of the advanced features listed below

**Save Game Status (5%)**

From the GUI you should be able to save the current game. The GUI should allow the user to reload the saved game.

**Design Patterns (5%)**

You are required to use the observer pattern at an appropriate place in your program. It is up to you to decide where is appropriate.

**Master ship mode (10%)**

The master ship has two modes; offensive and defensive. In defensive mode the master ship behaves as described above. In offensive mode it takes three enemy ships on the same square to destroy the master ship (as opposed to two enemy ships in defensive mode). You should be able to change the mode of the master ship **from your GUI while the game is running**.

**Report**

You should write a report explaining your design (including your class diagram). You should also explain where and why you have used polymorphic programming and interfaces. Your report should also discuss the main features of your GUI, and include screen shots.

Your report should also discuss the advanced features that you attempted to implement.

If you have used any programming techniques not taught on the module you should discuss what techniques have been used, where they were used and why.

**Marks**

Marks will be allocated as follows

Overall design and implementation 20%

Use of polymorphic programming 10%

JUnit Testing 10%

GUI 20%

* Creativeness (10%)
* Technical (10%)

Programming technique not taught 10%

Report 10%

**Advanced Features**

Save and reload game status 5%

Use of Observer pattern 5%

Master ship mode 10%

**Collaboration and Plagiarism**

This is an individual piece of assessment and the work submitted must be entirely your own. You are not allowed to collaborate with other people or to copy the work of other people.

In the event of any doubt about authorship you will be interviewed by the School of Computing Academic Conduct Officer and may be asked questions about any aspect of the work.

**Submission**

You should demonstrate your program working during the demos **on Tuesday 17th April 2018** (details of the demo order, times etc. will be sent out by e-mail one week before the demos)**.**

After the demonstration you should submit your assignment via Moodle.

Your submission is entirely electronic via Moodle, you should submit your

* Java project, exported as an archive file
* JAR File
* report

by **23.59 on Tuesday 17th April 2018**