**Software Development 3 Coursework** (SET09101)

**“Swamp Wars”**

“Get out of my swamp!”



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

This assignment constitutes 100% of the module assessment.

A new software company called “Swamp Software Solutions” is moving in to town and instead of interviewing for their top programmer job they are going to give the job to the best solution to the problem statement given below.

**Problem Statement**

Using design patterns, threads and a GUI, you are to write the Java code for a game involving an Ogre and some ogre enemies in a swamp. You can design your own game (some marks are awarded for creativeness) or use the sample game given below.

If you design your own game, your game should have moves, a GUI and the ability to undo moves. 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.

**“Get Out of My Swamp” (Sample Game)**

“Get Out of My Swamp” consists of a swamp, an ogre and a number of ogre enemies

**The Swamp**

The swamp can be thought of as a four by four grid.

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

Your game should be written so that the user can select any swamp size but the default is four by four. When the game starts the Ogre is placed in a random square except for the top left corner.

**The Ogre**

**Diet**

The ogre, who is called HEK, is a fussy eater and he only eats three types of food; Knights in Shining Armour, Very Big Macs and “ogre enemies”. HEK’s default diet is Knights in Shining Armour; he particularly likes them because they are crunchy on the outside and soft on the inside. We need to be able to change HEK’s diet dynamically.

**Moving**

When the game starts the ogre is randomly allocated to a square in the swamp, with the exception of the top left square. When the ogre moves, it can move to any of the neighbouring squares as shown below. The ogre cannot move out of the swamp.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | Ogre |  |  |
|  |  |  |  |
|  |  |  |  |

**The Ogre can move to any of its neighbouring squares**

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

**Ogre Enemies**

**Moving**

HEK’s swamp is not all sweetness and light. There are three types of ogre enemies that can inhabit HEk’s swamp; an ogre loathing snake, an ogre despising parrot and an ogre hating donkey. To keep HEK on his toes (well claws to be more accurate) we need to be able to generate Ogre Enemies dynamically and add them in to his swamp.

There is a hole in the fence around the swamp in the top left corner and it is through this hole that all enemies enter the swamp (enemies can enter the swamp but nobody ever leaves the swamp alive). Once in the swamp the enemies move in exactly the same way as the ogre; i.e. they move randomly to any of the neighbouring squares.

Every time the Ogre moves there is a one in three chance of an enemy entering the swamp, the type of enemy is completely random.

|  |  |  |  |
| --- | --- | --- | --- |
| Donkey |  |  |  |
|  |  |  |  |
|  | Ogre |  |  |
|  |  |  |  |

**Enemies always enter the swamp through the hole in the fence in the top left corner of the swamp**

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

**Conflict**

If the Ogre moves into the same square as an enemy, the Ogre kills the enemy and the enemy is removed from the swamp.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Donkey |  |  |
|  |  |  |  |
|  |  | Ogre |  |
|  |  |  |  |

**The Ogre destroys a lone enemy**

If the Ogre moves into the same square as two or more enemies, what happens depends on the Ogre’s diet. If his diet is “ogre enemies” he kills the enemies. If the Ogre’s diet is anything other than “ogre enemies” then the ogre enemies kill the Ogre and the game is over.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Donkey |  |  |
|  |  |  |  |
| Snake |  | Ogre |  |
|  |  |  |  |

**Two or more enemies kill the ogre when his diet is not” ogre enemies” and the game is over**

If the Ogre’s diet is “ogre enemies” it takes three or more enemies in the same square to kill the ogre. Remember that you should be able to change the diet of the ogre from the GUI while the program is running.

**GUI**

Your game should be controlled from a GUI. From the GUI you should be able to start the game, change the diet of the Ogre and make a move and undo moves. Your GUI should also have a visual representation of the game which should be updated as the game status changes.

**Report**

You should write a report explaining where in your code you have used threads and design patterns, which patterns you have used and why. You should highlight the problems you would have encountered if you had not used patterns. You should also discuss how you designed your program to work with variable swamp sizes.

Your report should also discuss the main features of your GUI, and include screen shots.

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

Creativeness of game 10%

Use of Patterns 40%

* Factory (8%)
* Observer (8%)
* Strategy (8%)
* Command (16%)

Use of Threads 10%

GUI 20%

Variable Swamp Size 10%

Report 10%

**Collaboration and Plagiarism**

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

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 1st December 2015** (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. You should export your project and submit that with your report. The easiest way to do this is to put everything into your src directory of the project and compress the directory; this means you only need to submit one file. You should also include a JAR file of your program.

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

* Exported Java project
* JAR File
* report

by **23.59 on Tuesday 1st December 2015.**