

CENG 443

Object Oriented Programming

Spring '2014-2015

Programming Assignment 1

Due date: 10 March 2015, Friday, 23:55

1 Objectives

In this assignment you are going to implement a small graphical system using Object-Oriented design principles in addition to practicing UML and design patterns.

Keywords: *Java, UML, OOP*

2 Specifications

In this homework you will implement a simulation of very famous MMORPG called **Knight Online** using Java. In Knight Online, heroes from two nations (**El Morad** (humans) and **Karus** (orcs, although the game calls them "Tuareks") fight against each other. In our simulation, the aim of the characters is to gain more **national points**. Characters will be spawned at their spawn point from two opposite corner of map. The details of this war are given below.

To make implementation easy the extra details of game are ignored. However, the essence of the scenario is maintained. In this simulation we will have three character classes: **Rogue, Mage and Priest** and each will have two basic skills: **Melee and Ranged attacks**. A new combat mode, which does not exist in original game, is also introduced to make things a little more interesting.

Additionally, you are required to design and document your system. A **class diagram** in UML is sufficient for this purpose. (bonus points will be awarded for extra diagrams such as activity diagram or statechart) You will generate the stub code from the class diagram that you design. After the stub generation, you can fill in the methods and implementation details of the classes. A class diagram is given to you as a guideline. Stick to the OO standards, use getters/setters, polymorphism, etc. (will be useful especially for the patterns)

2.1 Classes

Basically, you will need 5 main classes:

- Team: There will be 2 teams to distinguish knights of different sides of the battle: El Morad knights will be blue and Karus, red.

- **Knight:** The abstract class that represents the players. Each team will have 8 knights and will spawn new knights from the corners when dead. Knights will move according to the currently assigned strategy. Each second, they will have 20% chance to change strategy randomly. It can move at any time, but can only attack every 1 second. You can use a timer to synchronize this behavior. Knight is extended by the following classes:

Rogue: This character class is recognized with its fast attack skills and combos. It is shown as a triangle.

Health: 500 HP

Speed: 100 px/s.

Normal Attack Skill: Damages 150 if touches the enemy.

Special Skill: If there is enemy in its range(200px Radius), it charges that enemy and its speed doubled until it hits enemy. Damages 250. In case of multiple enemies, one of them is selected randomly.

Mage: This character class is recognized with its mass and remote attack skills. On the map, mage is shown as circle.

Health: 400 HP

Speed: 75 px/s.

Normal Attack Skill: Damages 100 if one enemy is inside its range(75px Radius).

Special Skill: Damages 75 all enemies inside its range(75px Radius).

Priest: This character class is recognized with its heal skill. It has a square shape.

Health: 350 HP

Speed: 50 px/s

Normal Attack Skill: Damages one enemy 150 when it touches.

Special Skill: Heals all allies by 75, including itself, inside its range(100 px Radius).

- **StrategicPoint:** Strategic points are located randomly on the map. When knights are inside the capture radius of a strategic point, each second, the knight and the team that has most knights inside the radius will gain 5 national points for each strategic points. In case of tie, no one will gain the points. There will be 5 strategic points on the map, with 150px radius.
- **Display:** An extension of JFrame where simulation state is viewed. It consists of a 2D view and text information on the side. This side panel displays the information about number of current knights, number of kills of El Morads, number of kills of Karus, gained total national points, the national point of the player who has maximum. The 2-D view will have an image of a map of Knight Online or any shot of a MMORPG game map.
- **Simulation:** This class is the one that holds all the information about the simulation and the main method necessary to run it. Also, timers or synchronisation mechanisms that one would need to run the game.

;

2.2 Patterns

In addition to the classes, you will need to implement, at minimum, the following three design patterns:

- **Factory Method / Abstract Factory:** Creations of entities (i.e. rogue, mage and priest) will be accomplished via entity factories. For a good object oriented design demonstration, make sure the object that creates the entities is unaware of the entity types.
- **Strategy:** To provide a behavior to characters, you will need to use the Strategy pattern. 4 strategies are required.

Attack Closest: The knight aims for the closest enemy and moves towards it to attack. With 20% chance each second, it will use special skill.

Attack Weakest: The knight aims for the enemy with the lowest hp and moves towards it to attack. While moving, with 20% chance each second, it will use special skill.

Capture points: The knight will aim for the most overwhelmed capture point and will try to equalize the numbers. Once it is in the range of capture point, it will move inside the radius. While moving, with 20% chance each second, it will use special skill.

More strategies can be added to this list for bonus points. Remember that to demonstrate a good design, the entities should not be aware of the strategy that they are assigned to.

Decorator: Use the decorator pattern at run-time. Make sure the decorated entities are not aware of the fact that they are being decorated. Each character will be decorated according to its NP (initially El Morads are blue and Karus are red) There are 3 Grades (Decorators) according to national points. 0-75 ıı Noob (Basic Character) 76-150 ıı Grade 3 151-300 ıı Grade 2 301+ ıı Grade 1

3 Regulations

1. **Programming Language:** Java
2. **Late Submission:** You have at most 3 days of late submission
3. **Cheating: We have zero tolerance policy for cheating.** People involved in cheating will be punished according to the university regulations.
4. **Newsgroup:** You must follow the newsgroup (news.ceng.metu.edu.tr) for discussions and possible updates on a daily basis.

4 Submission

Submission will be done via COW. Create a tar.gz file named `e1234567_hw1.tar.gz` that contains all your source code files.

The following command sequence is expected to run your program on a Linux system:

```
$ tar -xf e1234567_hw1.tar.gz
$ javac *.java -o e1234567_hw1.jar
$ java -jar e1234567_hw1.jar
```