Middle East Technical University Department of Computer Engineering

CENG546

Object Oriented Programming Languages and Systems

Spring ’2014-2015

Programming Assignment 1

By: Semih Aktaş, Yusuf Mücahit Çetinkaya, Emre Külah, Arif Görkem Özer, Yamaç Kurtuluş, İrem Tanrıseven

Due date: 19 April 2015, Sunday, 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. (We called this, **Knight Offline**) 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: **Normal attacks and Special Skills**. Using these skills, two teams having 8 **Knights** will combat. Each kill will net 75 points to the killer, and its team. When knights are dead, new knights are spawned from spawn points.

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 a chance to change strategy randomly. A knight can move at any time, but can only attack every 1 second. Please visualise normal and special attacks using graphic elements. You can use a timer to synchronize the attack behaviors and cooldowns. 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, first encountered is selected.

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

  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(100px 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).

- **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, as well as a logo on the side panel.

- **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 such 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, While moving, attacks normally to the rivals it gets close and with 20% chance each second, it will use special skill.

  Move Randomly: The knight will wander around the map, going in random directions toward random points. While moving, attacks normally to the rivals it gets close, and each second it will have %20 chance to 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. A dot in a different color over the tokens is sufficient.

0-75: Noob (Basic Character)

76-150: Grade 3

151-300: Grade 2

301+ : Grade 1

2.3 Design

In addition to the code, you will also submit your UML documents. You need to generate classes from this document, as a stub, then fill them in with the logic. Please submit this document as well. You can also submit other design documents, state diagrams, sequence diagrams etc. for bonus.

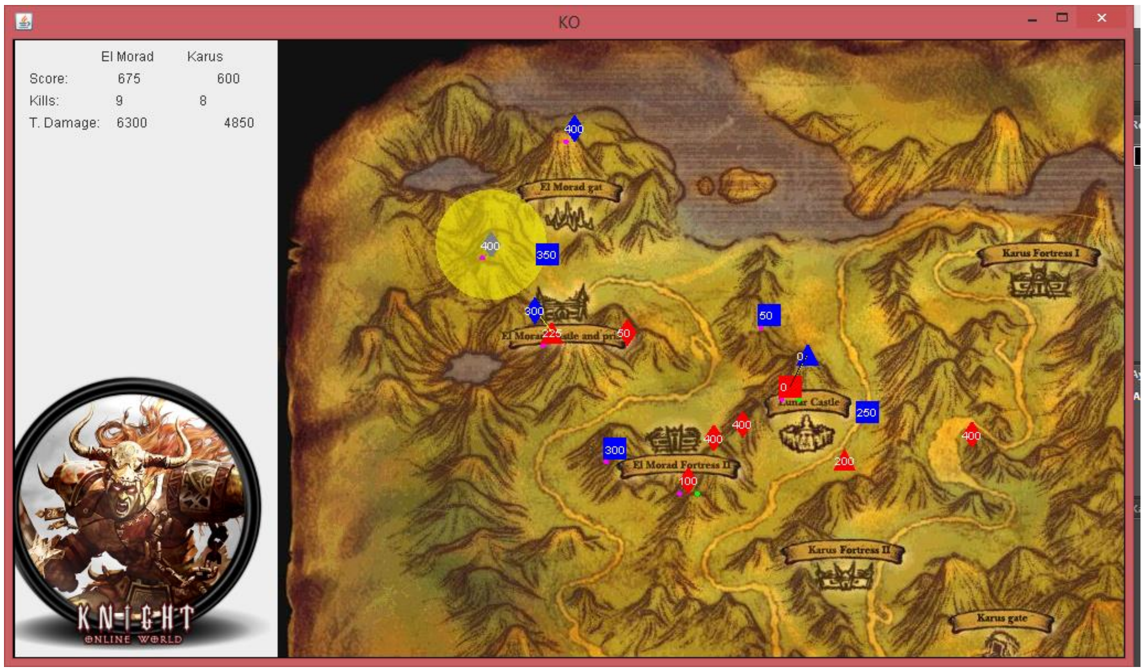
3 Regulations

1. **Programming Language:** Java
2. **Late Submission will not be accepted.** Deadline is 19.04.2015.
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. Additionally, we are going to have a tutorial session for questions about homework in 2 weeks.
5. **Submission:** Submission will be done via COW.
   Create a zip archive file named "<ID>_<FullName>.zip" that contains all your source code files. (e.g. e123456_<YamacKurtulus>) Please do not use Turkish characters.
   Create 2 folders for source and docs, and put your source files under "source" folder, and your UML design document and additional information under "docs" folder.
6. The following command sequence is expected to run your program on a Linux system:

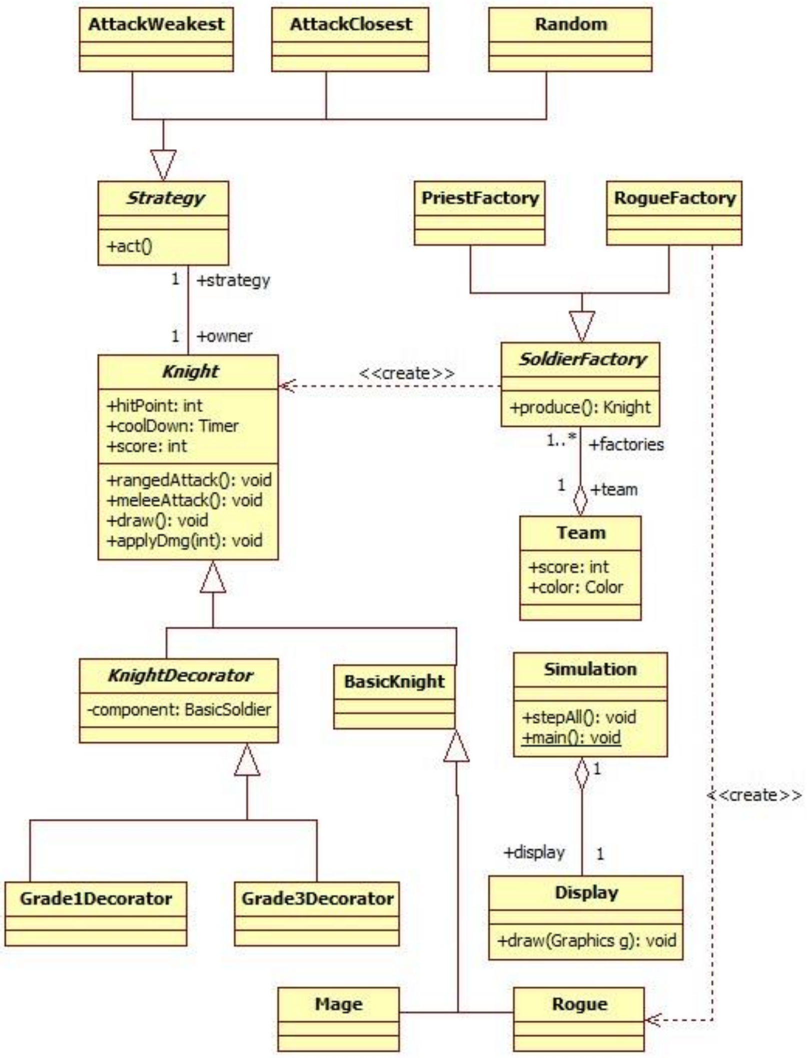
```
 $ tar -xf e1234567_hw1.tar.gz
$ javac *.java -o e1234567_hw1.jar
$ java -jar S
```

7. **Bonus:** There are various design decisions such as additional strategies, additional design patterns, better graphics, visual improvements, detailed UML designs etc. that are deliberately left openended for this project. There will be bonus points for such extra e_ort. However, late submissions will not be accepted, therefore try to implement minimum requirements by the deadline.
8. A screenshot of the simulation and a sample **non-complete** UML class diagram is given on next pages.
9. You can ask questions about assignment on CoW or to Yamaç Kurtuluş at A-410.

Good luck.



	El Morad	Karus
Score:	675	600
Kills:	9	8
T. Damage:	6300	4850


AttackWeakest
AttackClosest
Random
Strategy
+act()
1 +strategy
1 +owner
PriestFactory
RogueFactory
SoldierFactory
+produce(): Knight
<<create>>
Knight
+hitPoint: int
+coolDown: Timer
+score: int
+rangedAttack(): void
+meleeAttack(): void
+draw(): void
+applyDmg(int): void
1..* +factories
1 +team
Team
+score: int
+color: Color
KnightDecorator
-component: BasicSoldier
BasicKnight
Simulation
+stepAll(): void
+main(): void
1
<<create>>
+display 1
Grade1Decorator
Grade3Decorator
Display
+draw(Graphics g): void
Mage
Rogue