Middle East Technical University   Department of Computer Engineering

CENG 443

Introduction to Object-Oriented Programming Languages and Systems

Spring '2020-2021
Assignment 1

Due date: April 25, 2021, Sunday, 23:59

1 Introduction

In this assignment, you will build a small graphical system using Object-Oriented design principles in addition to practicing UML and design patterns.

You will implement a simulation MMORPG called **Knight Online**, which used to be very famous, 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 corners of the map. The details of this war are given in the next section.

Keywords: *OOP, Class Diagram, UML, Simulation, Java*

2 Specifications

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 the system and to document it by using javadoc. A class diagram in UML is sufficient for the design of the system. You need to comment the code properly to generate javadoc. You will fill in the methods and implementation details of the classes. Do not rename/remove the existing ones. Basic implementation for drawing the panel and images is provided. You can simply pass Simulation and Display instances to the required classes to draw the knights. You can draw the shapes using the Polygon class as described here. You will complete the parts where **TODO** comments are. Stick to the OO standards; do not use public access level everywere, use getters/setters, polymorphism, etc. It is better to start by designing the application by drawing a UML class diagram.

Design patterns are typical solutions to common problems in software design and help us to follow the design principles. In this homework, you are expected to use four design patterns; Singleton, Factory Method, Strategy and Decorator. Singleton is already implemented in classes Display and Simulation for you. You can have detailed information in the next sections.

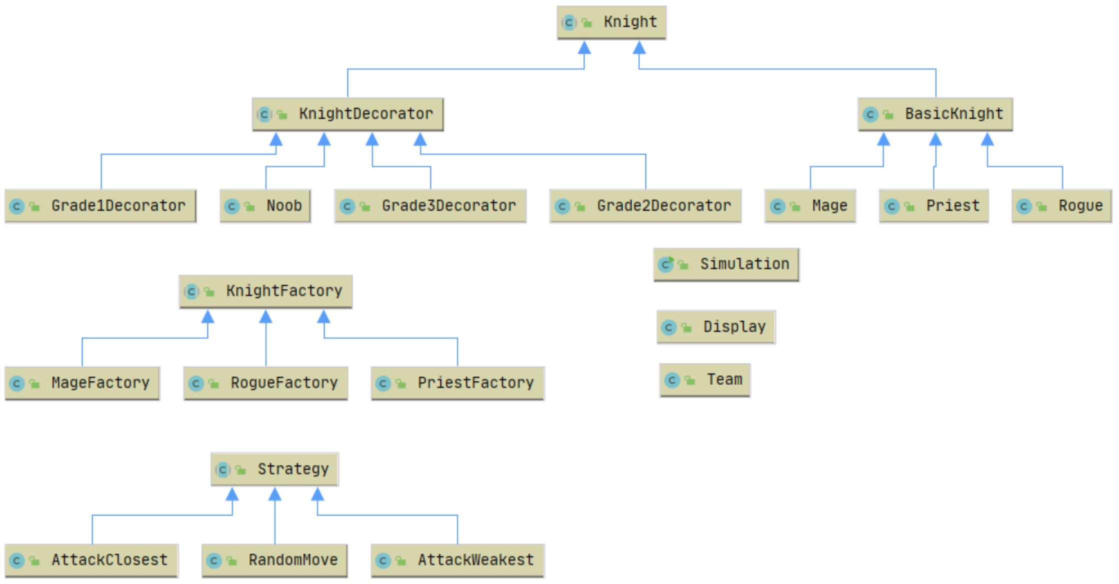


Figure 1: IDE generated Java class diagram.

2.1 Animation Entities

The classes that are provided to you are listed and explained below. In Figure 1, a simple, IDE generated Java class diagram is provided to show the big picture. Be careful, it is not the UML class diagram we expect from you.

- **Simulation:** Singleton class that contains the main method. It is the responsible class for managing the simulation and storing the required entities.
- **Display:** Singleton class that represents the display on which animation entities, i.e. national points, kills, total damage are repeatedly drawn.

- **Team:** The class for representing two sides.
- **Knight:** Represents the abstract base class for the animation entities.
- **BasicKnight:** It is the base class for other types of Knights.
- **Mage, Rogue, Priest:** The concrete classes that extend the BasicKnight class.
- **KnightDecorator:** An abstract base class for Noob, Grade1Decorator, Grade2Decorator, Grade3Decorator.
- **Noob:** The concrete decorator class that extend the KnightDecorator class that is responsible to draw the main token and Knight's HP in it.
- **Grade1Decorator, Grade2Decorator, Grade3Decorator:** The concrete decorator classes that extend the KnightDecorator class that are responsible to draw the grade with a colored dot over the tokens.
- **KnightFactory:** An abstract base class for knights.
- **MageFactory, RogueFactory, PriestFactory:** The concrete factory classes that extend the KnightFactory class.
- **Strategy:** Represents the abstract base class for AttackClosest, AttackWeakest, RandomMove strategy classes.
- **AttackClosest, AttackWeakest, RandomMove:** The concrete strategy classes that extend Strategy class which specify the moving strategy of the Knights.

2.2 Simulation Details

The simulation class holds all the information about the simulation and the main method necessary to run it—also, timers or such synchronization mechanisms that one would need to run the game. The display class will draw the required window. The side panel will display the information about national points, the number of kills, total damage of both teams. The 2-D view will have an image of a map of Knight Online, as well as a logo on the side panel.

There will be two teams to distinguish knights of different sides of the battle: El Morad knights will be blue and Karus, red. 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 visualize normal and special attacks using graphic elements. You can use a timer to synchronize the attack behaviors and cooldowns.

The details of different Knight types are as below;

- **Rogue**
  This character is recognized for its fast attack skills and combos. It is shown as a triangle.
  Health: 500 HP
  Speed: 100 px/s
  Normal attack: Damages one enemy 150 if it touches the enemy.
  Special skill: If there is an enemy in its 200px radius range, it charges the enemy, and its speed

is doubled until it hits the enemy. Damages 250 if touches the enemy. In case of multiple enemies in its range, first encountered might be selected.

- **Mage**
  This character 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
  Range: 100px radius
  Normal attack: Damages 100 if one enemy is inside its 100px radius range.
  Special skill: Damages 75 all enemies inside its 100px radius range.

- **Priest**
  This character class is recognized for its healing skill. It has a square shape.
  Health: 350 HP
  Speed: 50 px/s
  Normal attack: Damages one enemy 150 when it touches.
  Special skill: Heals all allies by 75, including itself, inside its 100px radius range.

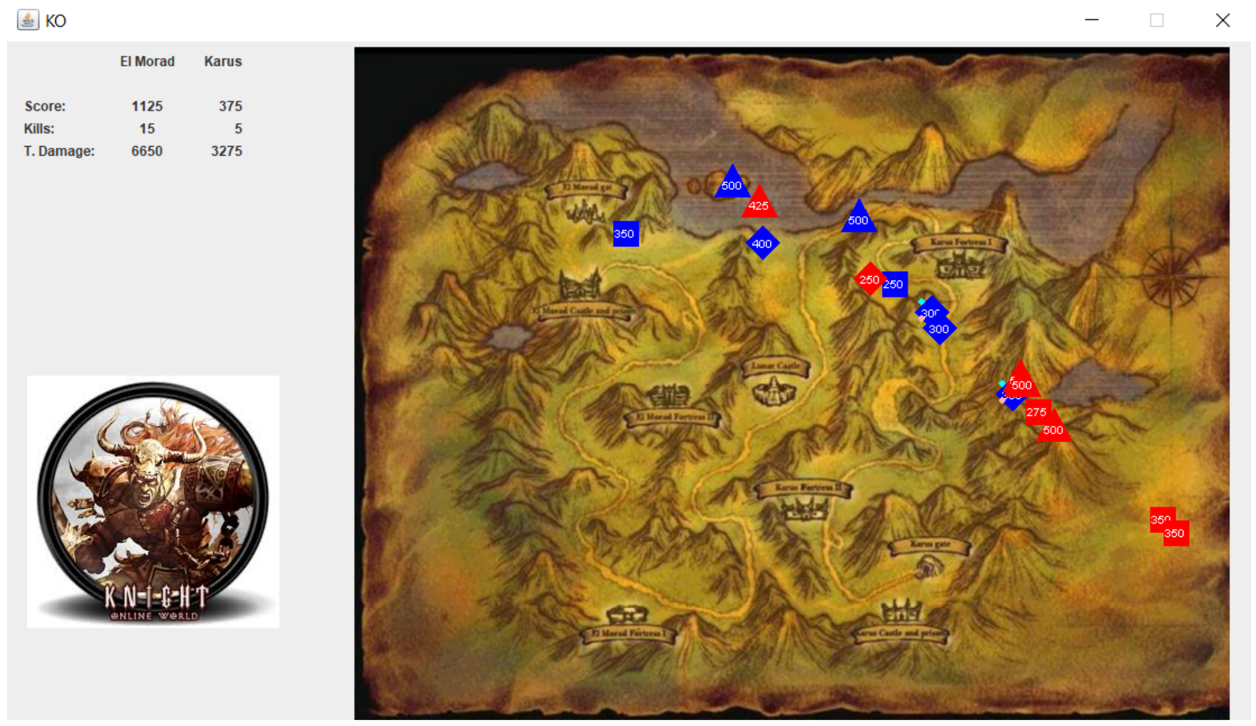
You will implement, three design patterns:

- **Factory Method / Abstract Factory:** Creations of entities (i.e., rogue, mage, and priest) will be accomplished via entity factories. You can visit **here** to get more information about Factory design pattern. For an excellent object-oriented design demonstration, make sure the object that creates the entities is unaware of the entity types.

- **Strategy:** To provide behavior to characters, you will need to use the Strategy pattern. You can visit **here** to get more information about Strategy design pattern. The details of concrete Strategy classes are listed below. Remember that to demonstrate a good design; the entities should not be aware of the strategy that they are assigned to.
  AttackClosest: The knight aims for the closest enemy and moves towards it to attack. With a 20% chance each second, it will use the special skill.
  AttackWeakest: 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.
  RandomMove: 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.

- **Decorator:** Use the decorator pattern at run-time. You can visit **here** to get more information about Decorator design pattern. Please make sure the decorated entities are not aware of the fact that they are being decorated. Moreover, decorator classes should not know each other. In other words, they should not depend on each other. Each character will be decorated according to its national points. There are 3 Grades (Decorators) according to national points. A dot in a different color over the tokens is sufficient. Noob decorator will draw the main Knight character as a circle with blue or red color.
  Noob (Basic character): 0-75 national points
  Grade3Decorator: 76-150 national points
  Grade2Decorator: 151-300 national points

Grade1Decorator: 301+ national points



	El Morad	Karus
Score:	1125	375
Kills:	15	5
T. Damage:	6650	3275

Figure 2: A screenshot of the simulation window.

3 UML Class Diagram

To properly document your code, you are required to draw a class diagram of your application exported as a PDF document. Try to keep your diagram as simple as possible by only including important fields, methods, and relations between the classes. Please, show the relations between classes such as Composition, Aggregation, Dependency, etc. Using **StarUML** might be useful for this task.

4 Sample output

Figure 2 shows the screenshot of the simulation. You can watch the sample video **here** to get an idea about the final application. If you download the video, you can watch it in better quality.

5 Regulations

1. **Programming Language:** Java

2. **Late Submission:** A penalty of $5 \times \text{Lateday}^2$ will be applied for submissions that are late at most 3 days.

3. **Cheating:** We have zero tolerance policy for cheating. Your solution must be original. People involved in cheating will be punished according to the university regulations and will get 0 from the homework. You can discuss algorithmic choices, but sharing code between groups or using third party code is strictly forbidden. In case a match is found, this will also be considered as cheating. Even if you take only a "part" of the code from somewhere or somebody else, this is also cheating. Please be aware that there are "very advanced tools" that detect if two codes are similar.

4. **Newsgroup:** You must follow the ODTUCLASS forum for discussions, possible updates, and corrections.

5. **Submission:** Submission will be made via ODTUCLASS. Create a "zip" file named "hw1.zip" that contains all your source code files inside "Source" directory and uml file, readme, javadoc in "Docs" directory. Please provide a readme which describes how to compile and run your code.

6. **Evaluation:** Your codes will be evaluated manually; small differences might be tolerated. In addition, your code will be examined to check the correctness of the implementation. You may get partial grades, so even if you are not able to implement all requirements, please submit the completed code. However, be sure that your submission can be compiled. Therefore, if you have differences when running the program, that will not be a problem.

   Important note: The visualization details are not so important. It is enough if one can distinguish the knights, decorators, strategies in the simulation visually. Do not worry about the visual details like colors, sizes, etc., and do not spend so much time on it. It would be best if you focus on object-oriented principles and design patterns. Please, do not ask for more details about the visualization; the requirements above are sufficient. You are free to demonstrate the simulation as long as it is similar to the video shared above.