CS136L: Computer Science II Lab – Fall 2018

Assignment	Points	Announced	Due
#7	40	Nov-05	Nov-23

RPG

1 Overview

Inheritance and polymorphism allow us to create a range of objects with wildly different behaviors, yet still can be interacted with in the same way. Consider a Role-Playing Game (RPG) video game in which you may be fighting anything from mutant frogs to wizards to dragons. Different enemies may carry different code to behave, attack and move in different ways, but they all can be targeted, animated, and looted in the same way. In this assignment you will be challenged to not only write classes representing elements in an RPG, but to design a class structure using inheritance. Below is a scenario, explaining what your system must be able to accomplish. It will be up to you to decide how to do it.

In our RPG, a set of contestants are to be placed into an arena to battle one another. The arena should contain two of each of the classes described below. We should be able to get a description of the arena with everything in it at the beginning and after each round of fighting. Contestants that are dead should be described as such. In each round, all contestants in the arena are given a turn to act, in which they will be able to attack if possible. After a round, if there is only one contestant left standing, they are to be named the victor. If no one is alive, pick a winner at random. Random values for damage are determined every attack. Random values for health are determined when the character is created. Below is a list of the different types of contestants.

Berserker

Berserker contestants attack a random contestant in the arena and deal 20 smashing damage. They may hit themselves. They can take 35 points of damage before dying. They take half damage from fire, and double from slashing damage.

Warrior

The warrior looks at all his potential targets and attacks the one he finds most threatening, which is determined by how much damage it would take to kill a target. Warriors never attack themselves. Their attacks deal between 10 and 16 slashing damage, and they take between 20 and 40 points of damage before dying. A warrior has a 25% chance to dodge smashing or slashing damage.

Wild Mage

Wild mages deal fire damage to everyone in the arena, themselves included. Each target takes a random amount of damage between 0 and 6 fire damage (calculated per target). It takes between 10 and 60 points of damage to kill a mage. When they are killed they deal 5 fire damage back to whoever killed them.

2 Learning Outcomes

By the end of this project students should be able to:

- write algorithms based on a technical problem description;
- utilize inheritance and polymorphism;
- work effectively with a partner using pair-programming;
- write an effective report that describes the students' problem-solving process.

3 Pre-Lab Instructions

Do this part before you come to lab:

- Review chapter 9 on inheritance and polymorphism from the textbook.

4 Submission Instructions

1. You should use NetBeans for this lab assignment.
2. In addition to the lab report, submit a ZIP archive that is exported from NetBeans by using the NetBeans menu (File → Export Project → To ZIP).
3. Add Javadoc comments for all classes, fields and methods that you implement

5 Lab Instructions

Do this part in lab:

1. Our first step is to determine what classes will be necessary and how those classes should interact with each other. Consider the problem description in the overview of this document. Create a class diagram for your proposed system. Your class diagram should include a list of all potential classes and inheritance relationships between them. Also, add public methods to your diagram. Make sure that your design is compatible with the provided main method and the Arena class skeleton.

2. Once you have your diagram, get it checked off by the lab staff, who will verify that it is a reasonable design, and that you may proceed. Once your design is checked off, include it in the planning section of your report, either by creating it with software or taking a picture.

3. You will now implement the system based on your design. It is a requirement that for both attacking and taking damage polymorphism be utilized. You will lose points if logic for how a character attacks or take damage is located outside of their own class file. The provided main method populates your scenario with two of each contestant type. Below is an example of the output of a working program. Your program should also print the description of the battle every round and declare a winner, though yours may look different.

In the arena...
A raging Berserker (35)
A raging Berserker (35)
A disciplined Warrior (33)
A disciplined Warrior (39)
A wild Mage (29)
A wild Mage (11)

A wild Mage (11) takes 20 Smashing damage from A raging Berserker (35)
A raging Berserker (35) takes 2 Fire damage from A dead Mage
A disciplined Warrior (39) takes 13 Slashing damage from A disciplined Warrior (33)
A raging Berserker (35) takes 26 Slashing damage from A disciplined Warrior (26)
A raging Berserker (33) takes 1 Fire damage from A wild Mage (29)
A raging Berserker (9) takes 0 Fire damage from A wild Mage (29)
A disciplined Warrior (33) takes 2 Fire damage from A wild Mage (29)
A disciplined Warrior (26) takes 5 Fire damage from A wild Mage (29)
A wild Mage (29) takes 1 Fire damage from A wild Mage (29)

In the arena...
A raging Berserker (32)
A raging Berserker (9)
A disciplined Warrior (31)
A disciplined Warrior (21)
A wild Mage (28)
A dead Mage

A disciplined Warrior (31) takes 20 Smashing damage from A raging Berserker (32)
A wild Mage (28) takes 20 Smashing damage from A raging Berserker (9)
A raging Berserker (32) takes 28 Slashing damage from A disciplined Warrior (11)
A disciplined Warrior (11) takes 15 Slashing damage from A disciplined Warrior (21)
A raging Berserker (4) takes 0 Fire damage from A wild Mage (8)
A raging Berserker (9) takes 1 Fire damage from A wild Mage (8)
A disciplined Warrior (21) takes 3 Fire damage from A wild Mage (8)
A wild Mage (8) takes 1 Fire damage from A wild Mage (8)

In the arena...
A raging Berserker (4)
A raging Berserker (8)
A slain Warrior
A disciplined Warrior (18)
A wild Mage (7)
A dead Mage

A raging Berserker (4) takes 20 Smashing damage from A raging Berserker (4)
A raging Berserker (8) takes 26 Slashing damage from A disciplined Warrior (18)
A disciplined Warrior (18) takes 4 Fire damage from A wild Mage (7)
A wild Mage (7) takes 0 Fire damage from A wild Mage (7)

In the arena...
A slain Berserker

```
A slain Berserker
A slain Warrior
A disciplined Warrior (14)
A wild Mage (7)
A dead Mage

A wild Mage (7) takes 14 Slashing damage from A disciplined Warrior (14)
A disciplined Warrior (14) takes 5 Fire damage from A dead Mage

In the arena...
A slain Berserker
A slain Berserker
A slain Warrior
A disciplined Warrior (9)
A dead Mage
A dead Mage

Winner: A disciplined Warrior (9)
```

6 Lab Report

Each pair of students will write a single lab report together and each student will turn in that same lab report on BBLearn. Submissions from each student on a pair should be identical.

Your lab report should begin with a preamble that contains:

- The lab assignment number and name
- Your name(s)
- The date
- The lab section number

It should then be followed by four numbered sections:

1. Problem Statement

In this section you should describe the problem in *your* own words. The problem statement should answer questions like:

- What are the important features of the problem?
- What are the problem requirements?

This section should also include a reasonably complete list of requirements in the assignment. Following your description of the problem, include a bulleted list of specific features to implement. If there are any specific functions, classes or numeric requirements given to you, they should be represented in this bulleted list.

2. Planning

In the second section you should describe what planning you did in order to solve the problem. You should include planning artifacts like sketches, diagrams, or pseudocode you may have used. You should also describe your planning process. List the specific data structures or techniques you plan on using, and why.

3. Implementation and Testing

In the third section you should describe how you implemented your plan. As directed by the lab instructor you should (as appropriate) include:

- a copy of your source code
- a screen shot of your running application / solution
- results from testing

4. Reflection

In the last section you should reflect on the project. Consider different things you could have done to make your solution better. This might include code organization improvements, design improvements, etc.

You should also ask yourself what were the key insights or features of your solution? Were there alternative approaches or techniques you could have employed? How would these alternatives have impacted a different solution?

5. Partner Rating

Every assignment you are required to rate your partner with a score -1, 0 or +1. This should be submitted in the comment section of the BBLearn submission, and not in the report document. If you don't want to give your partner a negative rating making sure not to use a dash before listing the number! You do not have to tell your partner the rating you assign them. A rating of 1 indicates that your partner was particularly helpful or contributed exceptional effort. A rating of 0 indicates that your partner met the class expectations of them. Rating your partner at -1 means that they refused to contribute to the project, failed to put in a reasonable effort or actively blocked you from participating. If a student receives three ratings of -1 they must attend a mandatory meeting with the instructor to discuss the situation, and receiving additional -1 ratings beyond that, the student risks losing a letter grade, or even failing the course.

6. Contribution

Every assignment you are required to describe your contribution to coding of the solution and writing of the report. You must include the percentage of your contribution. This should be submitted in the comment section of the BBLearn submission, and not in the report document.

7 Grading Rubric

This lab assignment will be graded according to this rubric:

Criteria / Component	Points
Report	10 pts
Correct Beserker class implementation	7.5 pts
Correct Warrior implementation	7.5 pts
Correct WildMage implementation	7.5 pts
Correct Arena implementation	7.5 pts

Lab assignment penalties:

Item	Points
Compilation errors	-40
Missing program	-40
Missing report	-10
Missing names	-4

Missing partner rating	-2
Missing screenshots	-2
Missing contribution description and/or percentage	-5
Too late to pair (if attended)	-4
Absent	-10
Insufficient / No commenting (if required)	-5
Improper format (e.g. NetBeans ZIP archive instead of a Java file)	-5

Note:

If your partner is not responding to your emails and/or not collaborating, don't hesitate to reach out to the lab TA aide and/or the primary instructor.

Colophon

This project was developed by Richard Lester and Dr. James Dean Palmer and improved by Dr. Mohamed M. Elwakil of Northern Arizona University.