RPG Design

1 Overview

Inheritance and Interfaces 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 and/or interfaces. 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 area should contain two of each of the classes described below. We should be able to get a description of the arena with everyone in it and the beginning and between each round. 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. When anyone takes damage, it should be announced who took how much, of what kind of damage, from who. 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 from the last to die. Below is a list of the different types of contestants.

Random values for damage are determined every attack. Random values for health are determined when the character is created.

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 of 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.) A mage 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:

- design object oriented systems using interfaces and inheritance;
- utilize polymorphism;
- use interfaces and inheritance to reduce coupling in a system;
- 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 and turn in your answer on BBLearn before you come to lab:

This project will require you to make use of inheritance and interfaces.

- Read Big Java chapters 9 and 10 including special topic 9.3 on page 441.
- For each of the following classes, is this class an abstract class or interface? Why was it designed this way? Write down your answers and turn in on BBLearn.
  - InputStream
  - JComponent
  - Iterator
  - AbstractList
  - NavigableSet

4 Lab Instructions

Do this part in lab:

Step 1

Our first step is to determine what classes will be necessary and how those classes should be related. Consider the problem description in the overview of this document. Create a UML diagram for your proposed system. Your UML diagram must include all dependencies, interface implementation and subclass arrows that will be necessary to meet the goals of the assignment. In addition, the UML diagram should contain the method signature for every public method of the class (excluding those implied by implemented interfaces or super classes.) You must create a design that has the fewest connections reasonable to accomplish the task. Assume that the main method that will instantiate the game is in a separate class that is not listed in your UML diagram.

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.

Once you have your UML 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.

Step 2

You will now implement the system based on your design. To earn full points, your code must follow your design. You may not utilize one class in another class unless that dependance is represented in your UML design. Start with a part of your design with low coupling, represented by the least number of arrows exiting the element. What are the minimum number of classes you could implement to test your logic? Implement this section first. Once you are satisfied it is working, implement more classes until your program is fully featured. Your main method should populate your scenario with at least two of each contestnat type. An example output is distributed with the lab. Your program should also print the description of the battle every round and declare a winner, though yours may look different.

5 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. In order to recieve

points, you MUST make a BBLearn submission.

Your lab report should begin with a preamble that contains:

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

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 funtions, classes or numeric requirements given to you, they should be represented in this bulleted list. It is recomended that you complete this section before you begin coding the problem, as gathering these requirements may help you organize your thoughts before you begin your soulation.

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. How do you plan on breaking up the problems into functions, classes and methods?

3. Implementation and Testing

In the third section you should demonstrate your implementation. As directed by the lab instructor you should (as appropriate) include:

- a copy of your source code (Submitted in BBLearn as .java files, should not be in your report document)
- a screen shot of your running application / solution. This should include a screenshot of the output, be it a terminal window or graphical interface
- results from testing. This is specific to the lab, and not every lab will include this element.

4. Reflection

In the last section you should reflect on the project. What part of the projec was the most difficult? Where there other solutions to this problem that you considered? Where there any new solutions you can think of now that you couldn't then? Do you think the way you chose was the best option, or would you go about the problem differently if you had to start over? How might the problem had been broken up into easier problems? If you had one more day to work this problem, what improvements might you make? Every problem has alternative solutions and solution has tradeoffs or improvements, you are required to identify some of these elements.

5. Partner Rating

Every assignment you are required to rate your partner with a rating between 1 and 10. For each patener a name and rating should be submitted in the comment section of the BBLearn submission, and not in the report document. You do not have to tell your partner the rating you assign them. A rating 8 or more indicates that your partner was particularly helpful or contributed exceptional effort. A rating of 5 indcates that your partner met the class expectations of them. Rating your partner 3 or less means that they refused contribute to the project, failed to put in a resonable effort or actively blocked you from participating. In the case where you give a very low rating, please explain why in the comment section of the submission. Those who recieve low scores may be asked to explain their actions to the lab staff, and additional low ratings after a warning could lead to losing a letter grade, or even failing the course. Consistant high ratings from partners are noted durring the grading process, and may be taken into account when rounding your final grade.

Colophon

This project was developed by Richard Lester and Dr. James Dean Palmer of Northern Arizona University. Except as otherwise noted, the content of this document is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.