
CSC 211 – Introductory Programming and Design

Laboratory Assignment 11: *Track and Smite*

Monday, April 14, 2008

Due date — Section 1: Wednesday, April 23, at 1 PM
Section 2: Tuesday, April 22, at 1 PM.

1 About this Assignment

1.1 Objectives of the assignment

This last lab assignment features again our bottle-picking barbarian. Today we will

- use inheritance and polymorphism to improve the design of our application;
- add fighting between our Barbarian and Monsters.

Read very carefully the text of this assignment before rushing to type code. Read it completely through once. Do not stop at the first unclear sentence you encounter; sometimes things are explained and detailed a bit later. Then you should start all over again. This time stop to ask questions when a point remains fuzzy, code your solution, and move on to the next section.

1.2 Handouts

This week's sole handout is the text of this assignment as a `.pdf` file.

2 What to Do, Part I: Revise your Design

2.1 The `GameCharacter` class

We spent some time in class discussing how to develop a parent `GameObject` class. Whatever we said about bottles and pizzas applies just as well to monsters and barbarians. This means that we can create a class `GameCharacter` and define the `Monster` and `Barbarian` classes as subclasses of `GameCharacter`.

2.2 `GameObject` implementation

While you are at it, complete the `GameObject` and inventory makeover as well.

2.3 Weapon subclasses

`Weapon` an abstract subclass of `GameObject`. for this application we only need one non-abstract subclass of `Weapon`: the `Sword` class. A `Weapon` object is characterized (in part) by the maximum amount of damage that it can inflict. A `GameCharacter` can take and carry multiple weapons but it can only wield (and use) one. You can define for your `GameCharacter` class a method `equip` that makes the character wield the first weapon available in his/her/its inventory.

Note, if you prefer non-violent action, you can have characters carry a camera and take picture of other `GameCharacter` objects. In that case, define a `Camera` parent class and, say a `CellPhoneCamera` or `VideoCamera` subclass (I guess in that case the monsters are paparazzi).

3 What to Do, Part II: Combat System

3.1 Hit points

Your `GameCharacter` class should have a variable that stores the hit points (or life points) of the character, and another to store the maximum hit point value for the character. Hit points can be regained simply by moving around (say, 1 point every 0 moves) or by eating food or drinking healthy liquids.

3.2 The `hit` method

Your `GameCharacter` class should have a method named `hit` that takes as parameter a reference to another `GameCharacter` object. The method uses a random generator to determine whether the target is hit and, possible the number of damage points inflicted. You can imagine here that in a full-fledged game we could take into account the armor class of the target and the dexterity, strength, and skill level of the `GameCharacter`.

3.3 Combat in the game

Monsters that are within one square of the barbarian when it is their turn to move can try to hit him/her. The player can get the barbarian to hit a monster standing next to him/her by clicking on the monster's icon (note that if there is a bottle/pizza/sword in that square, your application should not bring up the pickup dialog in the middle of a fight).

4 What to Hand in

4.1 End-of-session evaluation

You are not expected to complete the assignment by the end of the lab session, but you are definitely expected to have done *some* work during that session. Try to use the lab session to make sure you understand everything about the assignment. Ask questions; try things; ask more questions.

You should *not* leave the lab before your work has been *evaluated*. This first evaluation is worth 10 pts out of 100 for the complete assignment. If you leave before you have been evaluated, these points are lost with no chance of a later evaluation.

4.2 Your project

A cleaned-up and properly named project folder containing all your source files and your report should be uploaded to your CSC211 submit folder. Do *not* include the documentation produced by javadoc to this folder.

5 How You Will Be Evaluated

5.1 Point distribution

The maximum number of points is 100, but extra points could be awarded for excellent aspects of the project or report. The point distribution for this assignment is as follows:

Execution evaluation

End-of-session evaluation	10 pts
Execution of the project handed in	30 pts

Source Code

Identifier names	5 pts
Good indentation and general readability	5 pts
Judicious comments within the code	10 pts
Javadoc comments	10 pts
Follows lab specifications	10 pts

Report

Discussion	10 pts
General quality of the writing and presentation	10 pts

If you submit a project late, then it is your responsibility to notify the TA (with CC. to the instructor) that the project is finally available in your submit folder. If you fail to do so, then the “late penalty clock” will keep ticking until the TA gets around to checking your folder and notices your project. Unless specifically asked to do so, do *not* mail your project folder as an attachment.