Homework 4

CMPS251
Fall 2015

Updates

Any updates to this handout after release will be described here:

- November 16: Update the Error Handling section to make it explicit that you should print an error message if the save game file being loaded is corrupt.

Overview

In this homework you will extend the Legend of Blatt game to include the ability to save and load an player's status from a file.

You should start this homework early. If you wait until the last minute, you may very well run out of time to finish it.

Academic Honesty

This is an individual assignment, meaning that you should work on it alone. All code turned in by a student must be written by that student. Copying code found on the internet or using code from other students is strictly prohibited. Students are permitted to discuss the assignment together, but *must not look at each other's code under any circumstances*. If you need someone to look at your code and help you, you should meet with one of the course staff. A sophisticated plagiarism-checking tool will be used to compare your solutions against all other student submissions as well as code from the Internet. If code is found to have been shared between students, all students involved will be penalized, including the student who actually wrote the code.

Students found to have violated this honesty policy will receive a 0 on the project and *may* be reported to the Vice President for Student Affairs. Repeat offenders will receive a 0 in the course and *will* be reported to the Vice President for Student Affairs.

Remember:

- If you are copying code from the Internet into your solution, you are committing academic dishonesty.
- If you look at another student's code, you are committing academic dishonesty.
- If you allow another student to look at your code, you are committing academic dishonesty.

Things You'll Need

There is Blatt code provided for this project. You should load the Blatt code into Eclipse and then write your solution. You may modify and/or add any files you need.

My solution to this homework is available for you to run on the homework helper service. You should make use of it to help you figure out any ambiguities that may exist in this handout. *Your solution should function exactly like mine* except for one difference: My solution includes the Blacksmith to help in your testing, but your solution is *not* required to include a Blacksmith.

Blatt Overview

The Legend of Blatt is a role-playing adventure game built around killing monsters and earning gold. In the version provided to you, after the user creates their character they are taken to a menu that allows them to go to the forest or quit.

If a player goes to the forest, then they can fight monsters for a chance to win gold pieces. A player has a limited number of hit points. During fights with monsters, the player and monster will exchange attacks. Each attack will remove hit points. If the monster runs out of hit points first, then it dies and the player receives gold. If the player runs out of hit points first, then the player dies and the game ends. If a player needs to recover HPs, they can take a nap in the forest and recover 5 HP per nap.

When the game is started, it checks to see if the file `player_save.txt` exists and contains a previously saved player. If so, it loads the player and prompts the user regarding whether or not they want to use it.

When quitting the game, the user is prompted to see if they want to save their game. If they do, then the player's information is stored into a file named `player_save.txt`.

You should play the game for a while in order to make sure you are familiar with how it works.

Extensions

You will need to extend the given Blatt code to include new functionality as described below.

Save Game

In this version of the game, a new concept has been added: Saving your game. When the user quits the game, they are prompted to save their player if they desire. If they choose to save their player, then all of the player's information is stored in text format in `player_save.txt`. The file contains information about the player one line at a time. The following lines are part of the files:

Line	Description
`name: <String>`	The name of the player.
`type: <String>`	The type of character the player is. Possible values are Thief, Warrior, and Wizard.
`currhp: <Integer>`	The current hit points of the player.
`gold: <Integer>`	The gold the player has.
`items:`	Line to indicate that the lines following contain items currently in the player's inventory.
`\t<Name><Type><Power><Sell Price><Buy Price>`	The details of one item in the player's inventory.

A sample file is as follows:

```
name: John Smith
type: Thief
currhp: 10
gold: 14
items:
	Battle Axe,1,5,100,50
	Gauntlets,0,1,50,25
```

Loading Game

When the game is started, it should check to see if the `player_save.txt` file exists. If it does, then it should load the player from that file and prompt the user to see if they want to use the loaded player. If so, then start the game with the player loaded from the file. If not, then create a new player as usual by prompting the user to enter the details of the new player.

Error Handling

Your program needs to be able to handle any and all errors without crashing. (Although in the case of some errors it may need to print an error message and quit.) This includes handling exceptions from the file reading/writing code as well as syntax errors in the saved game file itself. If there are syntax errors in the save game file (meaning it is invalid in any way) then you should print an error message and start the game as if there was no save game file.

Hints on Testing

Due to the limitations of the homework helper, you will not be able to upload your own save game files and see how my solution responds to them. This limits your ability to test thoroughly. As such, here are some friendly hints regarding things you might want to test and make sure that your program responds appropriately:

1. There could be unexpected lines in the input. (For example, “Hi Mom”).
2. There could be lines that are almost correct, but not quite. For example, “currhp: Lots”.
3. There could be missing lines. (For example, a save game file without a name line.)
4. The lines in the file could be in a different order than presented above. (This is not an error, you should be able to load the player regardless of the order of the lines.)

Grading

Your homework will be graded on functionality as well as code formatting.

90% of the points will be for functionality. Projects that do not compile will receive an automatic 0 for functionality. The grading of functionality will be based on us running your code with a series of test cases where we supply a variety of inputs and verify correct outputs. Each test case is graded as either correct or incorrect; there is no partial credit for test cases.

10% of the points will be based on code formatting and comments. This includes proper using of indentation, following the Java naming conventions described in the Unit01 slides, and adding good comments.

Submission

You must submit a zip archive of your working Eclipse project to Blackboard. You should produce this archive from inside Eclipse. In the left sidebar, right-click on your project (probably called hw4) and choose "Export". In the menu that appears, click on General, then Archive File, then Next. On the next screen, choose "Browse" to choose where to output your .zip file, then click "Finish".

Prior to submission, manually check the contents of your zip file to ensure it contains the .java files you expect.

Due Date: Thursday, November 19 at 11:59pm on Blackboard. After this deadline the submission link will be disabled and you will not be able to submit the project. Students wishing to make use of their late days will then need to submit using the late submission link.

Submissions by email *will not be accepted*. You must submit on Blackboard.