Introduction to Programming

CMPT 120 and CMSC 120 • Fall 2012

Project 5 - game v0.8

Goals

To continue development of your semester-long project: a text adventure game in the spirit of The Hitchhikers Guide to the Galaxy, Planetfall, and Zork. Also, to show off your expertise in using **S**oftware **D**evelopment **B**est **P**ractices, as well as Git.

Instructions

Fix anything that was incorrect or incomplete with your prior project. (Commit. Push.) Then, beginning with a perfect implementation of the prior version of your game, implement the following new features:

- □ Develop a Location prototype {id, name, description, item, toString()} [10 points]
- □ Instantiate an instance of Location() for each location in your game. (If you have 10 locations in your game, you'll have 10 different instances of the same prototype.) [5 points]
- □ Store those location instances in a global array. [5 points]
- □ Refactor your game so that the location functions use this global array instead of the hard-coded text you had before. [5 points]
- □ Develop an Item prototype {id, name, description, toString()} [5 points]
- □ Instantiate an instance of Item() for each item that can be found in your game (at least four of them) and store zero or one item at each location. [5 points]
- □ Declare a global array to hold the player's inventory. [5 points]
- □ Modify your `take` command to use the inventory array. [5 points]
- □ Update your `i` or `inventory` command to make use of this array when listing the items the player is carrying. [5 points]

Advice

Test, test, and test again. Then test some more. When you think you've tested enough, go back and test again. Then get someone else to test for you while you test theirs. Rinse and repeat.

Push your work to your Git repository early and often. While you're in there . . .

- Be sure to write meaningful commit messages.
- Practice using *diff* to see the differences between successive versions of your code.
- Practice reverting to an earlier version so that you'll have that option in the future.

Don't forget to test. A lot. Really. (Rilly.)

Submitting

1. Push your work to your Git repository **before** the class in which it is due.
2. **Print** and staple your source code **before class** and hand it in at the **start** of the class in which it is due. Remember to include your name, the date, and the assignment in the (copious, meaningful, and accurate) comments in your code.