CS 314 Code Inspections Pre-Class Assignment

Pre-Class Tasks:

1. Determine your role through discussion with your teammates. Your team should have one member playing each role. If you are not yet in a team, you can assume a role based on your last name:
    - Last name begins with A-F: Moderator/End User
    - Last name begins with G-N: Maintainer
    - Last name begins with O-Z: Tester
2. Complete pre-class readings. (Estimated time: 10 min)
3. Complete pre-class inspection preparation using the listed materials. (Estimated time: 20 min)
4. Take the on-line quiz before 11:55pm on Tuesday, Jan 26. You may take the quiz multiple times.

Overview

This pre-class work is meant to help you achieve the following learning objectives:

1. Understand the purpose of inspections /reviews, and what can be inspected.
2. Be able to describe a formal code review.
3. Be able to describe and compare different light-weight code review processes; state their strengths and weaknesses, and analyze a situation to determine what method would be best suited (formal or one of the light-weight methods) for code inspection.
4. Prepare for an inspection, from the perspective of a given role, and find defects in a given piece of code based on that role.

Pre-Class Readings:

- "Four ways to a Practical Code Review" from Jason Cohen at Smart Bear.
- Inspection slides, 'CS314SP2016-inspections.pdf'.

Pre-Class Inspection Preparation:

Materials you need to study:

- 1 paragraph descriptions of each inspection role, found in this document.
- Code inspection checklists for Maintainer, Tester, End User, found in 'CS314-InspChecklists.pdf'.
- Description and diagram of the Adventure Game, found in this document.
- Line numbered code listings, found in 'AdventureGameCode.pdf' or via the jar file loaded into an Eclipse project for on-line viewing. **ONLY INSPECT AdventureGame.java AND Player.java** – other files are background information

What you need to do for this task:

1. Read the description of the role you have been assigned.

2. Review the game description and diagram to get an idea of how the game is structured and how it is supposed to work.

3. Use the checklist associated with your role to find problems with the 2 code files AdventureGame.java and Player.java. Your job is only to find problems, NOT figure out how to fix them. Fixing problems is up to the owner of the code to do later.

4. Make notes of the kind of problem you found and file name and line number where you identified it. You will need these notes for the design studio. These notes are all you need to bring to the design studio, however listings of the 2 files AdventureGame.java and Player.java may also be useful so you can follow the issues that the other inspectors in your group have found.

5. If you find other problems not on the checklist for your role, but they are problems that you think a person who is really in your assigned role might run into, then note these as well.

6. If you find problems that are not associated with your role, you can note them, but only bring them up in the design studio if no one else mentions them.

Roles that will be used in the inspection – you have been assigned one of the following roles:

This design studio will use groups of 3, so whoever has the Moderator role is also assigned the End User inspection role. The other 2 people in the group each have only one inspection role.

1. Moderator/End User:
   Moderator: person who will keep the meeting focused and moving, logging defects and only allowing discussion to the point that the maintainer understands the issue – NO PROBLEM SOLVING!
   End User: person playing the game who has no idea how it is written. What kinds of input and output are expected based on the game description? How are these implemented in the code – is there sufficient information for the end user to figure out what is expected of them or to help them if something goes wrong?
2. Maintainer: person who has inherited this code and will have to add features to it. Review the code from this point of view: you have to be able to repeat and correct bug reports turned in by users, and no doubt have to add features to the game – speculate on what kind of features may be requested in the future and think about ways the existing code might make adding them easier/harder.
3. Tester: person who gets the code to test. Assume that anytime you get the code it compiles. What kinds of tests do you need to create, based on both the description of the game and the code itself. Often the testing group runs all tests and testers have to decipher all the test output to figure out what happened and which developer needs to fix things.

Role-related Checklists – use the file 'CS314-InspChecklists.pdf' to find sample items related to your role that you need to look into as you review the Adventure Game code.

Adventure Game Description –Adventure Game, code courtesy and copyright Prof. James M. Bieman

The Adventure Game is a version of an old game called *Colossal Cave Adventure*. Years ago, students spent much of their free time playing Adventure. Playing the Adventure Game, you (the player) visit a series of interconnected underground rooms in which treasure is hidden. You can move from room to room searching for treasure. The objective is to bring the treasure out of the cave.

Some rooms have doors that are locked. To open locked doors, you must find the key to locked doors, which are placed somewhere in the labyrinth. As you go from room to room, you can

- look at the room,
- go into an adjacent room or through an adjacent door,
- pick up an object in the room, or
- drop an object that you are carrying.

Like the original game, the Adventure Game uses a simple text-based command line interface. After entering a room, a textual description of the room is displayed. Then the player decides on an action. For example, to move into an adjacent room, you simply type the direction to move: "n, s, e, w, u, d'' for North, South, East, West, up, or down. (All input in the old game was text, since computers in those days did not use a mouse or widgets. Simple abbreviations were both easier to implement and use.) Of course we could build the game today with a fancier interface, one with visual displays of rooms and their contents, and with button input.

Game Code

We will inspect Adventure Game.java and Player.java

See attached pdf for the entire source code, or load the jar file into and Eclipse project to review it. DO NOT REVIEW ALL THE SOURCE CODE, JUST ADVENTUREGAME.JAVA AND PLAYER.JAVA – the extra files are only included to be used for reference if necessary.

Additional Inspection/Review References:

Smart Bear: "11 Best Practices for Peer Code Review",
http://smartbear.com/SmartBear/media/pdfs/11_Best_Practices_for_Peer_Code_Review.pdf

Game Diagram