CIS*4150 F13 Project 1 — Part 1: Checkers Software and Ad-hoc Test Suite Creation

Part 1 is graded out of 20 points (20% of Project 1)

Part 1a: Software Design Document [8 points]

- Deliverables:
  - [1 point] One to two page summary of the requirements/design
    - Similar in style to the Program Overview document
  - [4 points] Simple Class Diagrams
    - Class/Interface Name
    - Instance variables (if a concrete class)
    - Method signatures
    - Public/Private denotation
    - Inheritance Structure
  - [3 points] Use Cases
    - Written from a "bird's eye" view
    - Need not be strictly requirements based: can include "design" elements
    - You may use any use-case format style you are comfortable with. It does not have to match the use-case style provided in the use-cases handed out in lab.
- Comments
  - This is to be a "lightweight" design. The purpose of the course is to test for faults in the code, not teaching object-oriented design.
    - Your goal should be getting the program working, even in a very simplified manner, in order to be able to apply the tests.
    - Consequently, I will be very lax in following "proper" requirements and design formalisms. You do not need to dot your 'i's and cross your 't's.
  - That said, please try to make your design as understandable as possible.
    - Write clearly and concisely (not verbose … less pages is better)
    - Point form is acceptable
  - You can include UML diagrams if you feel it makes your design more intelligible, however, only the deliverables listed above are required.
  - Concentrate your design efforts on the client side of the program. The server side, while not complete, has been mostly provided in the document I gave out in lab (which is also provided on Moodle in a wiki).
  - For class diagrams, include only the instance variable and method details for your large and/or important classes. Small classes should be named, but the details can be elided.
- Due Date:
  - Wed Sept 25th, 8:30am

Part 1b Software Implementation [8 points] [+ up to 8 bonus points extra]

- Deliverables
  - Code
    - [4 points] Basic Code working – no extra features
      - Basic code: 2 players playing British rules checkers with an RMI server
    - [1 point per extra feature] Example extra features
      - Deciding on a game
      - Observers
      - Implementing different rules (1 point per rule set)
      - GUI
      - Etc.
  - User's Manual
    - [1 point] A lightweight users manual describing how to run your software.
  - Walk Through
    - During lab time
    - Where the grading of the code will take place
    - Each group has 15 minutes to show the software working and give an overview of their testing (see P1c). This is just a spot check. No extra documentation needs to be developed for this deliverable.

- Hints:
  - Get the RMI version of "Hello World" up and running first. Expand that code slowly, testing frequently. Always make sure that communications between client and server remains intact.
  - Implement the vital classes and methods necessary to get a bare bones version program working.
    - Add features, even those on the "must have" list, afterwards if they are not necessary to get the basic program running.
    - Remember, the software is not where the marks are coming from – the testing of the software is the important part
  - Use versioning software, such as GitHub.

- Due Dates:
  - [Code] Fri Oct 11th, 11:55 pm
  - [Walk Through] Wed Oct 16th, 8:30 am

Part 1 c Ad-hoc Test Suite creation [4 points]

- Procedure:
  - First make sure your code compiles without errors or warnings
  - Then debug your code; but while debugging, record every debugging test you use: i.e. <program-input | expected-output>
  - Debug until code is error free (to the best of your knowledge)
  - Keep track of which test cases in the test suite discovered bugs and which did not.
  - Note: These debugging tests becomes the base test suite upon which future test suites will be built
- Deliverables
  - [3 points] Test cases in test suite
  - [1 point] List of whether each test case found at least one bug
- Due Date: (same as P1b)
  - Fri Oct 11th, 11:55 pm

Formatting and Upload instructions

- All code and documents are to be uploaded to Moodle as a single archived and compressed file (such as .zip, .tgz etc.)
- Include all .java source files, as well as any makefile or instructions on how to obtain any third party software needed to compile an run your code.
- Include a readme file describing how to compile and run your code
- All documentation, including UML diagrams if any, should be in .pdf format. For the writing, please use 12 pt font, with at least 0.7" margins around.