Project Overview

Each student shall **incrementally design, code, and test** their software application based on the requirements stated for a project increment (aka spiral). The first three project spirals are defined by the instructor. After this, the student follows their plan (developed in the project plan assignment) to complete their project.

The student shall continue to design and implement the game selected during the first increment.

Artifacts for Spirals/Increments 4-6

For each spiral/increment described in your project plan, you shall submit ***a single zip file*** containing the following artifacts:

- Your updated project plan.
  - Update the Actual Hours to Complete for the spiral you are currently submitting.
  - Update the scope of the remaining increments/spirals, if you need to adjust your plan.
- A single design document (docx file) which consolidates ALL of the design artifacts into a single document.
  - Your single design document will eventually contain design models for all four sections of the document - architecture, data, interfaces, and components.
- Code via Java source code files.

Due Dates for Increments 4-6

See the Project Plan Instructions document for the start date and due dates for each of these increments.

Assessments for Increments 4-6

The rubrics used to assess your artifacts are**:**

- **Code**: all evaluation criteria are used.
- **Data Model**: all evaluation criteria are used, once you begin to design and implement the XML requirements.
- **Design**: all evaluation criteria are used.
- **HCI**: all evaluation criteria are used, once HCI is covered in class.
- **Plan**: recording actual hours for each spiral.

Scope of Remaining Increments (Spirals)

Mandatory requirements are stated using *shall* while optional requirements are stated using *should*. The requirements are in no particular order, but they are numbered for easy reference.

1. Your game shall be played by one human player and one computer player.
   1.1. The human player shall be allowed to enter a name.
   1.2. The computer player shall be named after the student that is developing the game.
   1.3. The two names cannot be the same.
   1.4. The game shall use a player's name when asking for input or displaying game results.

2. The design of your game shall adhere to the model-view-controller architectural pattern.

3. While the game is being played:
   3.1. The human player shall be allowed to end game play even though they have not won the game. When this happens, allow the player to either exit the software application or restart game play with a total score of zero for both players.
   3.2. The ***human player*** shall play the game as described in the game rules.

3.3. The ***computer player*** shall play the game as described in the game rules, with the following exceptions:

3.3.1. **Farkel game**:

3.3.1.1. Rules 3.d and 3.e do not apply. The computer player can only roll the six dice once and determine its turn score. It cannot reroll during a turn. This includes when the roll of the six dice results in "hot dice".

3.3.2. **Gin Rummy game**:

3.3.2.1. For rule 5.a, the computer player shall always pick the top card from the stock pile. It cannot pick the top card from the discard pile.

3.3.2.2. Rule 6.b does not apply to the computer player (i.e., the computer player cannot decide to knock). It can end the hand only by having all ten cards in melds (see rule 6.a) or by the stock pile being empty (see rule 6.c).

4. Once a player has won the game:

4.1. The human player shall be given a choice between exiting the software application or restarting game play with a total score of zero for both players.

5. Your game shall persistently store a game in an XML-formatted text file.

5.1. The XML file shall contain data for only one game.

5.1.1. When a new game is started, any data in the XML file for an older game shall be removed.

5.2. The XML file shall contain the following data items.

5.2.1. Each player name.

5.2.2. The turn score and total score for each player's turn.