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 increments are defined by the instructor. After this, the student follows their plan (developed after the third increment) to complete their project.

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

Artifacts for Increment #3

- The total number of hours it took the student to complete the increment.
  - Hint: It will benefit you if you take the time to keep track of the hours you spent designing, coding, and testing each increment. Splitting up your total hours into these three buckets will help you produce a better project plan that you develop after the third increment.
- In one zip file:
  - A dia file that contains a class diagram.
  - A dia file that contains a state machine diagram that describes the human-computer interactions.
  - Java source code files.

Grading for Increment #3

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

- **Code**: all evaluation criteria are used.
- **Design**: **all evaluation criteria except *Design Elements* are used**.

Scope of Increment #3

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. Items below that represent a change in the project scope, when compared to the previous increment, ***are identified in bold italics***.

1. Your game shall be played by one human player.

2. While the game is being played:
   2.1. The software application shall exit only after the human player has won the game.
   2.2. The human player shall be given the option of ending game play after each of their turns, even though they have not won the game. When this happens, the software application will exit.

3. For each human player's turn, the:
   3.1. **Cosmic Wipeout game** shall:
      3.1.1. Allow the player to roll all five dice.
      3.1.2. When the roll results in a score using only the rules listed below, add the appropriate turn score to the human players total score.
         3.1.2.1. Produce a turn score only for the scoring described in rules requirements 2)a), 2)b), and 2)c).
      3.1.3. When the roll results in no score, the player's turn has ended and their total score is not changed.
      3.1.4. Play continues for this one human player until they have won the game or they have elected to end the game.

3.2. **Tunk game** shall:
- 3.2.1. Allow the player to deal seven cards after shuffling the deck.
- 3.2.2. The player will immediately say "tunk" indicating the end of this hand. (For this increment, you do not implement any portion of requirement 6.)
- 3.2.3. The player will add up all of their unmatched cards, ignoring only those cards that are in a matched set as described in rules requirement 4.a. This hand total is added to the human player's total score. (Note: in this increment we are ignoring matched sets described by 4.b and are not dealing with the multi-player aspects of how *other* players may play on a tunker's matched set.)
- 3.2.4. Play continues for this one human player until they have lost the game or they have elected to end the game.

4. ***The design of your solution shall adhere to the model-view-controller architectural pattern.***