READ THIS ENTIRE DOCUMENT CAREFULLY AND IMMEDIATELY

University of Puerto Rico
University of Puerto Rico Mayagüez
Department of Computer Science and Engineering

CIIC 4010 / ICOM 4015: Advanced Programming

Programming Assignment I: Poker Memory Game

Due Date: Friday, December 1, 2017 11:59PM

In this project, you will form a team with another student from the class. You will be given a working version of the **Poker Memory Game** with a couple of basic game levels: **Equal Pair** and **Same Rank Trio**. During each turn in the game, the player uncovers a set of cards from cards arranged in a grid with the goal of obtaining a wining hand. Each game level establishes such properties as the size of the grid, the number of cards that will be uncovered per turn, what a wining hand consists of, among others. For instance, the **Equal Pair Level** uses an 8x8 grid, allows the player to uncover two cards per turn, and a wining hand consists of a pair of cards with the same rank. If the uncovered cards conform a wining hand, then the cards are left uncovered and the turn ends. If the uncovered cards do not conform a wining hand, then the turn is unsuccessful and the cards are covered again after a short delay. The game ends when all the cards are uncovered or no other winning hands can be formed. The application allows the gamer to start a new game of a desired level by selecting this level from the "New Game" menu.

You job is to develop and integrate new game levels and improvements as described below by maintaining, as much as possible, the organization and structure of the design used by the original author of the game. That implies that you should think carefully about how to best extend the provided code base with the additional functionality with the minimal necessary changes to the code. Your code must also be efficient and avoid redundant code and unnecessary computation.

Phase I: Keeping Track of the Score

To complete this phase, you need to incorporate a notion of a numeric (long int) score on each of the existing levels. You should then modify the game display to incorporate a score display at an appropriate location on the screen. The score must be updated after each turn is completed. For the **Equal Pair Level** the score should be calculated as the sum of 50 points for each uncovered pair minus a 5-point penalty for each unsuccessful turn taken. For the **Same Rank Trio Level** the score should include 100 points for each uncovered trio, plus the sum of the ranks in the trio, minus a 5-point penalty for each unsuccessful turn taken. The values for cards with letter ranks should be 20 for A's, 13 for K's, 12 for Q's and 11 for J's.

HINT: Consider extending the MemoryGame class with a subclass that incorporates separate panel for the grid and the score area.

Phase II: Flush Level

This phase will incorporate a new **Flush Level** to the game. In this level, the grid should be the same size as the **Same Rank Trio** level (5 rows by 10 columns). However, players uncover five cards instead of three on each turn. A wining hand consists of all five cards having the same suit. The score for each hand should be computed as 700 points plus the sum of the ranks in all the cards. For instance, a hand with 2, 5, 10, K, and A of clubs will accumulate 700 + 2 + 5 + 10 + 13 + 20 for a total of 732. A 5-point penalty should be deducted from the score for each unsuccessful turn taken. Remember to update the Help page to include a clear and concise description of the level.

Phase III: Straight Level

This phase will incorporate a new **Straight Level** to the game. In this level, the grid should be the same size as the **Flush** Level (5 rows by 10 columns). Players uncover five cards on each turn. A wining hand consists of all five cards in sequence with at least two distinct suits. The score for each hand should be computed as 1000 points plus 100 times the rank of the highest card in the sequence. For instance, a hand with the ranks 10-J-Q-K-A will score 1000 + 100 * 20 (value of A) for a total of 3,000 points. A 5-point penalty should be deducted from the score for each unsuccessful turn taken. Remember to update the Help page to include a clear and concise description of the level.

Phase IV: Combo Level (Original)

This is your chance to be creative! This phase will incorporate of a new level that combines the existing 5-card levels (Flush and Straight) plus at least one additional poker hand of your choice. Each team should come up with a level that is unique among all teams. As a minimum requirement, the player will uncover five cards on each turn and the player will be able to choose, selecting from a set of buttons in a modal dialog frame, between poker hands formed by the uncovered hand. Alternatively, the player may choose to PASS. The PASS move will cover the cards again allowing the player to uncover a higher scoring hand in future turns. You may decide how to score each turn, but keep in mind that scores should be proportional to the unlikelihood that the hand is obtained. Remember to update the Help page to include a clear and concise description of the level.

Phase V: A New Ending for the Game

In this phase, you should modify all phases of the game in order to automatically finish the game as soon as no more wining hands remain among the covered cards. The game should pop up a window with an appropriate message to this effect.

Collaboration Using Git and GitHub

To facilitate collaboration among developers in a team we will be using GitHub Classroom. You will receive a URL to the assignment in GitHub classroom through which each team member can register and obtain the GitHub URI for the team's private repository (repo for short). Team members must agree on a unique team name. The first member to register will create the team repo, while the second team member should register with the already existing team.

The team private repository will store the master copy of the project source code. Each team member will work on its local git repo held on her/his Eclipse workspace. When a team member is ready to incorporate working changes into the master repo, he/she will push her/his local copy to the remote repository. The second team member could then pull the updates into her/his local copy with that of the repository. Make sure that both team members carefully watch the videos on using Git/GitHub under Eclipse available at our eCourses site.

Each team member MUST carry out her/his own individual commits to the team repo. The history of commits will be used to determine the evaluate the contributions of each team member to the project. If you confront difficulties finding, collaborating with or contacting your partner please talk to the professor immediately.

Extending (versus Modifying) the Base Code. Collaboration with Course Staff.

You are not allowed to modify the classes provided in the base code. All your changes must be incorporated by creating new classes and subclasses extending the existing ones. If for some reason you must definitely modify some class in the base code, you should ask the course staff to carry out this modification by means of a GitHub pull request.

Turning in your projects

The grader will pull your project from the GitHub classroom after the deadline into a fresh Eclipse Java project and run it as a Java Application. The program must run without further configurations to the IDE

nor further inclusion of additional libraries. The program should run on standard Java 8. No programs with compiler errors will be accepted nor graded.

Submit eCourses Activity

In addition to pushing your final code to GitHub classroom, each team member must complete the eCourses activity entitled **Programming Assignment I**. This activity will serve to record some basic information about your project and will allow team members to evaluate their own performance and their partner's in the project. This activity must be completed and submitted by each team member individually before the project deadline

Grading

Each phase of the project will have an equal weight of 20% towards the final grade and will be graded approximately according to the following rubric. Notice that each team member may receive a different grade based on her/his individual performance.

Criterion	Description	Weight
Functionality	Works according to specification	40%
Design	Follows code organization. Code reuse, Avoids unnecessary changes and complications with the code base.	20%
Teamwork	Team member contributed fairly and effectively to the success of the project. Partially based on individual GitHub commits.	30%
Efficiency	Your code does not carry out unnecessary or redundant calculations.	5%
Readability	Your code is easy to read and follow. Uses a reasonable amount of commenting. Does not over-comment.	5%

Bottom line: Getting the game to work correctly is a necessary, but not a sufficient condition to get full credit for your project.

Additional Ideas

Feel free to give the game your very own look-and-feel. Some ideas include having an original back side art for the card deck, add a more vivid color scheme to the display, add music or sounds effects to the card flips, moves and other events during the game. This is an opportunity to build something that you can feel proud of.

Collaboration vs. Cheating

We encourage all teams to help each other and share ideas as this is one of the best means of learning from your peers. However, **sharing of any source code by any means is strictly prohibited**. You may also get ideas for coding from the Internet, but the team members must write all their source code by themselves. Using code that was not written by your team will be treated as academic dishonesty. Not citing the sources of any source code used in the project will be considered an aggravating factor. Academic dishonesty on this and any other project will entail immediate failure in the course and possible formulation of charges at the appropriate UPR disciplinary bodies.

You may use any class from the standard Java 8 API. However, using a non-standard library written by a third-party developer will not be allowed.

You may divide the work among the team members, but both team members are responsible for understanding every aspect of the code developed by the team. You may instead consider working following an Extreme Programming or Working in Pairs Approach in which both team members work simultaneously on the same part of the code. Each team member should commit her/his own contributions to the project. The

Remember, the purpose of this project is to learn to program and develop an application that you can feel proud about. Please do not miss this opportunity.

Additional Hints

Here is a list of additional hints and recommendations for both team members:

- Start playing with the basic game and studying the base code IMMEDIATELY
- Talk to your team member and decide on a work schedule/structure appropriate for both
- This project cannot be completed in a couple of days. It requires gradual continuous progress
- If your partner is a bit behind offer help
- If your partner does not collaborate talk to the staff immediately

Good Luck and Have Fun!