Team Djaghrouri APCS1 pd5
Bayan Berri, Irene Lam, Jason Lam January 3, 2017

Final Project Proposal
JEOPARDY

Description:

Upon running Woo.java, the user is prompted with two options, either to play the pre-existing game or to create their own game, that is, the game of JEOPARDY! The codes will overlap, with creating a game simply necessitating an extra step of answers to several questions (e.g. how many rows/columns, money in each box, questions and answers). The code will then prompt the user with how many players exist, where 3 or more players invoke the option to make teams (the more the merrier!). The user is given an option to either play Normal Jeopardy or Special Edition Jeopardy (described below). From there, categories are chosen (e.g. Geography, Logic, Miscellaneous), and the game follows (to a degree) regular jeopardy, with a tie invoking either All or Nothing! (players tied play for all or nothing) or Bet! (every player has the option to bet and potentially win). Further explanations of the mechanics of the game and usage of topics learned this past semester are explained below.

"Jargon" Explained:

- Special Edition Jeopardy randomly enhances certain boxes with a chance for player(s) to double the amount of money they win (similar to Double Jeopardy), except failure comes with a randomly generated twist (e.g. losing double the money, forcing the player to pause the game to do an hour of homework if they're playing alone).
- Normal Jeopardy fills x location(s) with Double Jeopardy, and follows the regular format for Jeopardy (x is answered after the user chooses to play Normal Jeopardy).

Extra Aspects that may be included:

- An overarching riddle/ puzzle that when answered wins the game. Each correct answer in the Jeopardy game gives a hint corresponding to the point value (e.g. 100 translates to the worst hint, 500 to the most obvious hint)
- User attracting qualities (e.g. color, timer)
- Saving the game file by updating a text file with the information of the game.
    - Separate files-- make it easier to see separate information.

Usage of Topics covered in the semester:

- *ARRAYS:* The "board" that is displayed is a 2D array (e.g. a standard board will display an array[6][5]). Questions and answers will also be located in separate arrays, in which the index of the question has its answer in its corresponding index.
- *ARRAYLIST:* While the user is creating their own Jeopardy game, everything will be done through the use of an ArrayList, because the size of the user's desired game board is unpredictable, meaning that ArrayList's expand method would be more economical.
    - The ArrayList will be converted into an array after the game board is finalized.
- *INHERITANCE*: The game will incorporate inheritance by having a superclass of game that would branch out to different subjects, for example, math would be one branch and science would be another. These subjects would then extend to specific topics within that subject.
    - Abstract superclass Game would have non-abstract methods to calculate points, overwritten toString to display the questions, sort methods to rank people (explained below). We will also have an abstract method for comparing the user input to the correct answer.
    - Math, for example, would be a subclass to abstract superclass Game. Math would then branch out to topics such as logarithms, quadratics, and linear equations. "Logs is a math, math is a subject" is one example of how you would read the inheritance tree.
- *ITERATION*: For every single time the user chooses and answers a question, the board will be re-displayed but with that question already chosen. While the board still has at least one question still not chosen, the game will still be displayed until all questions have been answered. Once board has no more questions remaining, it will go into All or Nothing! or Bet!
- *SORT:* Following each round (when every player has attempted to answer a question), the scoreboard will be printed, in which each player's names will be listed in order based on their scores. This will be done through the use of early exit bubble sort, because the points following each round should be marginally close to the previous round's points. Furthermore, early exit bubble sort is more time efficient if there is an instance in which the array is already sorted.