**Project P2- Chip-A-Rookie**

**Project Explanation:**

The project aims at exercising object oriented principles by simulating a game of Guessing A Secret Word or Hangman. The project gives us a good insight into how inheritance works, and the concept of specialization and generalization.

**Program Flow and Description:**

**Activity 2:**

With regard to activity 2, it is required to play the game manually in both Single and Multi player mode. The procedure that I have followed are as follows.

1. In the class P2A2\_Raghuraman\_aparrnar,the main method is defined to take a user name as input. Once the name is given, a welcome message with the username is printed and the game instructions are displayed.
2. The user then enters a choice of choosing single or multiplayer mode. A single startGame() method is defined with the number of players as argument (one if single player and the user entered number if multiplayer mode).
3. Some assumptions that I have made whilst playing the game:
   1. A user can guess both character or word at a particular round.
   2. A user cannot enter the same character twice in two different rounds. If the user enters a previously entered character in any other round, then the game prompts him/her to enter another character.
   3. With regard to guessing a word, the user will be able to enter the same word guess in two different rounds. This has not been handled, and if the guess is wrong, the game will increment the wrong guesses for that player.
   4. InputMismatch exceptional flows have not been modelled, that is entering a string in place of an integer etc.. However, if the right datatype is given, then the program does implement some error checking regarding the validity of the input.
   5. It is assumed that all characters entered are lowercase. Using uppercase is not required.
   6. Checks for whether a word is a word or not has not been made. However, a character has been checked to see if it is a legitimate alphabet.
   7. In case the player enters a character which has already been displayed as being part of the guess word (him/her being allowed to guess that character since someone else had guessed it before and made it appear in the guess word), then it is flagged as a wrong guess and the player’s wrong guesses are incremented.
4. For the multiplayer mode, the game rubric is as follows.
   1. The game initializes a common guess word for all the players. The hint and the number of blanks in the word are displayed initially.
   2. Each player takes turns in guessing either a character or the word at every round.
   3. The player who COMPLETES guessing the word or the remaining characters of the word is deemed the winner of the game.
   4. Each player has his/her own incorrect guess counter and bomb, that is set only according to the wrong guesses that he/she makes.
   5. Each player is not allowed to enter a character that he/she previously entered character in another round. This is player specific and not common to all players.
5. At the end of each game, the player who won is displayed. If all the players have lost, then that result is also displayed.
6. Scenarios handled:
   1. In a multiplayer scenario, it is required to keep track of each player’s game status, that is, whether they are still playing or not, by means of the number of incorrect guesses made. If there is a situation where three players are playing the game, and player A has exhausted his/her six incorrect guesses, then the game ends only for player A and player B and C can still continue with the game.
   2. If all three players lose their six guesses, then the game must terminate. This has also been handled.

**Design rationale behind P2A2\_Raghuraman\_aparrnar class:**

1. The first choice was to provide a single method for the startGame() method for both single and multi player. This saves a lot of code since only the number of players needs to be passed as an input to the function, based on which the game would take place.
2. The next choice was to abstract the logic for the word guessing and the character guessing. This was done by placing common logic in the guessIsRight and guessIsWrong method.
3. The logic to find bad input is also done and updated in the checkBadGuess method. In future, the checking for a legitimate word can also be added to the method, in addition to character specific validation.
4. The main method also contains validation to see if a user enters negative number of players, as such scenarios are hypothetical.

**Activity 3:**

With regard to Activity 3, it is required to play the game in Manual and Auto Player mode, where the computer randomly generates the input and tries to guess the word.

1. In the class P2A3\_Raghuraman\_aparrnar,the main method starts with the input prompt of the user name and the display of the welcome message for the user.
2. Then, there is a prompt for manual playing or auto playing.
3. If manual mode is chosen, then the P2A3\_Raghuraman\_GAME\_aparrnar class is instantiated and the question is loaded. Then the startGame() method is called.
4. In case the auto play mode is chosen, the computer takes the number of players and the number of times as an input. Some assumptions that I have made are:
   1. The number of players for auto player mode remains constant for all the games. This is to reduce the manual intervention. This could also be automated by generating a random number, which has not been implemented.
   2. The computer makes only character guesses and not word guesses.
   3. The computer is also not allowed to make the same character guess twice for a particular game.
5. Once all the games are run, the results are tabulated and displayed to the end user.
6. Scenarios handled:
   1. In a multiplayer scenario, it is required to keep track of each player’s game status, that is, whether they are still playing or not, by means of the number of incorrect guesses made. If there is a situation where three players are playing the game, and player A has exhausted his/her six incorrect guesses, then the game ends only for player A and player B and C can still continue with the game.
   2. If all three players lose their six guesses, then the game must terminate. This has also been handled.

**Design rationale behind P2A3\_Raghuraman\_AUTOPLAY\_aparrnar class:**

The class **P2A3\_Raghuraman\_AUTOPLAY\_aparrnar** extends the **P2A3\_Raghuraman\_GAME\_aparrnar** class.

1. The startGame() method is overridden to remove all manual intervention and have the computer automatically generate a random character for each round.
2. In addition to that, two additional methods to load a guess word (loadWordAndHint()) and generate a random character (generateRandomCharacter()) have been defined.

**Design rationale behind P2A3\_Raghuraman\_GAME\_aparrnar class:**

The code for the P2A3\_Raghuraman\_GAME\_aparrnar is written with the reuse of code from the

P2A2\_Raghuraman\_aparrnar class.

**Possible Usage of this implementation:**

The implementation for this game can be reused in any contexts that involve a single/multiplayer mode game, such as a Quiz game. In such a case, the Question class can be modified to include the appropriate logic for the questions. The separation of concerns makes it easy for such modifications to be effected, however, care must be taken since several other classes depend on the Question class and hence it may be better to use Question as an interface and have several applications implement it based on the game context, eg for Hangman or quiz.