**Project 1- Rockstar**

**Project Explanation:**

The objective of the program is to understand and exercise Object Oriented Principles of using encapsulation using classes and objects. The program is a simulation of a die game, wherein the computer simulates the function of rolling a die and predicting win/loss based on the die value.

**Activity 2:**

The list of functions that I have used in the program are:

* **public static void welcome(String name):**

**Responsibilities handled by the function:**

1. Displaying the welcome message with the given user name input
2. Providing details about the game creator and a short description of the game
3. Instructions to play the game

**Error scenarios handled:** None

**Design Rationale:**

The responsibilities have been grouped to a single function due to the fact that it will be extended in the forthcoming Activity 3.

* **public static void startGame()**

**Responsibilities handled by the function:**

1. The function simulates the function of rolling a die by generating a random number that is taken as the face value of the die after rolling.
2. Based on the game rules, the win/loss is predicted.
3. The user is prompted to press the Enter key after every roll.
4. If the player wins/loses, he/she is prompted to choose to play again.

**Error Scenarios Handled:**

None, since the random number generator generates values between 4 and 24, which is the minimum and maximum values for a six faced die.

**Error Scenarios to be handled:**

The program might fail for dice that have more or less than six faces, such as a seven faced or ten faced die. In such cases, the maxRange value in the program should be appropriately updated to reflect the number of sides in the die, an extension which has been implemented in Activity 3.

**Design Rationale:**

The user is prompted to press Enter key so as to bring an element of realism into the program.

* public static void detectEnterPress()

**Responsibilities handled by the function:**

1. The program waits for the enter key press and returns when it receives it.
2. If there is any error, an IOException is thrown.

**Error Scenarios Handled:**

In case there is an I/O error, the exception is handled and the stack trace is displayed for the user to debug.

**Design Rationale:**

This functionality has been placed in a separate function since it is repeatedly called throughout the game, and it provides better reusability. For example, in Activity 3, I would be adding an extension of the computer automatically playing the game. In such a case, I would simply remove the function call rather than modify my existing code to remove all instances of waiting for the Enter key press.

* **public** **static** **void** promptToPlayAgain()

**Responsibilities handled by the function:**

The method is repeatedly called at the end of every game to prompt the user to press y or Y to start the next game or press n or N to stop the game and exit the program.

**Error Scenarios handled:**

The scenario where any other button other than the ones stated above has been handled. In such cases, the user will be again prompted to enter one of the valid characters.

**Design Rationale:**

This is kept as a separate method call so as to enable reusability in next activities.

**Activity 3:**

The list of functions that I have used in the program are:

* **public** **static** **void** welcome(String name)

The function is the same as that used in Activity 2.

* **public** **static** **void** startGame(P1A3\_DIE\_Raghuraman\_aparrnar dieObjects[])

**Responsibilities handled by the function:**

The function is an extension of that used in Activity 2 and is used by the player to play the game manually. Some additional aspects which have been customized to suit the requirements of Activity 3 are:

1. The random number generating function has been modified. Since a die class has been created, for each die object in the array dieObjects[], there is a die roll function that handles random number generation and setting the die value.
2. The prompt to start a new game when the current game ends has been removed.

**Error Scenarios handled:**

The random number generator generates values from 1 to the number of sides for each die.

**Design Rationale:**

An aspect of encapsulating logic into functions that perform some operations on data is that they can be easily extended and reused as and when needed. Here, the old function startGame() from activity 2 has been reused and little modifications have only been done.

* **public** **static** **void** startGameAuto(P1A3\_DIE\_Raghuraman\_aparrnar dieObjects[], **int** numTimes)

**Responsibilities handled by the function:**

The function is an extension of the startGame() function used in Activity 2 and is used to make the computer automatically play for a specific number of times. Some additional aspects which have been customized to suit the requirements of Activity 3 are:

1. The random number generating function has been modified. Since a die class has been created, for each die object in the array dieObjects[], there is a die roll function that handles random number generation and setting the die value.
2. The prompt to start a new game when the current game ends has been removed.
3. The enter press after each die roll has been eliminated.

**Design Rationale:**

Here, the old function startGame() from activity 2 has been reused and little modifications have only been done.

**Error Scenarios handled:**

The random number generator generates values from 1 to the number of sides for each die.

**DIE class:**

The die class provides a blueprint to create die objects for the game. It has been created with encapsulating attributes and getter, setter calls.

**Responsibilities handled by the class:**

1. The Die class has a set of attributes -> numberOfSides, color, numberOfInstances and currentValue for describing real world aspects of a die.
2. Functionalities for setting and retrieving die values have been defined.

**Design Rationale:**

To change the individual aspects of a die, the getter and setter methods for each of the parameter have been defined. Although they have not been used in the current implementation, in future, when there is a need to change the number of sides in a die and similar scenarios, the existing implementation can easily handle it.

**Other areas where this project can be extended:**

The project can be modified and reused for similar gaming applications, such as Card games and the like. Better support can be added by the provision of a GUI to use for other games such as Snake and Ladder, Minesweeper etc.