This program plays the gambling game Craps. It uses 6 classes to do so that being of one main class, three classes, and two interfaces.

The App(main) class plays craps with the user and has all the logic that comes with it. The way it is played is that the user chooses the amount of chips to bet being 5, 10, 15 or 20 chips. If it’s not any of that or they don’t have enough chips bet any of those amounts, the user will be prompted to choose their betting amount again. After the correct betting amount is chosen, then the game plays with the user on the pass line bet, rolling two 6-sided dice with roll method of TheDice object. Wins increase the chip amount by the bet and losses lose the betted chips for the bet. The pass line bet has two phases: The first roll and the rest of it. In the first roll, a 7 or 11 is a win, a 2, 3, or 12 is a loss, and other numbers become a point for the second phase. After the first roll, then the rest of the game begins where the question is that will the point roll (the first roll) again or will there be 7 first? The first case is a win for the user while the latter case is a loss for the user.

After the game is played with the pass line bet, the user is asked if they want to continue. On the case they want to, they can press enter and ignore it. Though, the method for it is looking for their reply to contain “no” inside it. If the user types something with “no” in it, the program will end. If the reply doesn’t contain no, the logic of the program continues looping back to before the bet was asked. It loops this until the user types a reply that contains “no” when asked if they want to continue with the game or when the user inevitably runs out of their virtual chips.

The Three classes are the Player class, Dice, and Die classes. The Player class is a class that stores all the player data for the program while it plays Craps. This data is the current chip amount, the bet amount, and the player name. One thing that happens before the bet is placed is that the data from the player class is shown to the user mainly the name they chose with the amount of chips they have. Before the bet is placed on bootup, the program makes the user’s player name by user input that is scanned by the scanner. It also manages what happens on a win or loss of the pass line bet with the bid method in the Bidder interface.

The Dice and Die classes are classes that store the Dice and Die respectively, but the Dice class has Die with the chosen amount of it that is an integer. Die is not directly used in the program but is used by Dice in aggregation. The Die has the sides and current roll variables which are integers that are used by the method that are shown in the interface, TheDice.

For the two interfaces, there is the Bidder and TheDice interfaces. The Bidder interface is for the Player class and TheDice interface is for the Die and Dice classes, respectively. The Bidder has the bid method where it takes a boolean (situation). For the player class’s implementation, the win or loss situation of the pass line depends on if it’s true(win) or false(loss). On a win, the player’s chips are increased with no prior deduction, on a loss their chips are deducted. TheDice interface has two methods: the roll method which takes nothing and returns a String for what is rolled and the value method takes nothing and returns an int. The interface is implemented in the Die and Dice classes to update their roll data variables and return a String of the roll using the roll method and to return those variables in the value method.

A picture of the UML:

Graphical user interface, application

Description automatically generated