**Dice Game:**

The problem statement given to me is Dice Game. The constraints given to me are:

- Implement a standalone program in your favourite programming language which takes the

values N (number of players) and M (points of accumulate) as command line arguments.

- Name the players as Player-1 to Player-N and randomly assign the order in which they

will roll the dice.

- When it's the turn for Player-X to roll the dice prompt a message like “Player-3 its your

turn (press ‘r’ to roll the dice)

- Randomly simulate a dice roll, display the points achieved and add the points to the

user’s score.

- Print the current rank table which displays the points of all users and their rank after

each roll.

- If the user gets another chance because they rolled a ‘6’ or they are penalised because

they rolled ‘1’ twice consecutively then print appropriate message on standard output to

inform the user.

- If a user completes the game, print an appropriate message on the output displaying

their rank.

I have divided this problem into four parts:

1. UI – User Interface – which is the front-end part visible to the customer
2. Service Layer – which provides service to the UI.
3. Entity – This refers to the objects present in the project Ex: Players, Dice, Board, etc.
4. Repository – This is used as a storage part where we will store all the details related to game.

**UI Layer:**

This layer is the front-end part visible to the players. We have a class Dice Game which is the real world Dice Game. The Dice Game uses objects of the Player, Board, Dice, Ranking System, Logs Printer System, DiceRepoService, PlayingOrderSystem.

1. Player – This class is used to store the details of the players

Attributes: playerId- int attribute is used to the unique Id for each player.

playerName – This is a String and used to store the name of the player.

points- int attributes used to store the score of the player at runtime.

Rank – int attribute used to store the rank after each roll

previousRollOne – Boolean variable stores true if previous roll is 1 else false.

hasExtraChance – Boolean variable stores true if roll is 6 and opts for an extra roll in same chance.

hasNoNextChance – If two consecutive rolls are 1’s then it will be true so that player doesn’t have next chance.

1. Board : It is used to show the game board to the players. It initializes the players at the start of the game.
2. Dice : This is the simulation of original dice in the game. The max and min scores are set to 6 and 1. It uses SecureRandom class for the random value generation between min and max including. I have used SecureRandom because it can generate upto 2^128 values and mod by 6. If mod is 0 add max to the score else return absolute value of score.
3. Ranking System: I have used the PriorityQueue for ranking the points based on descending order. I have given the players with same points different rank based on the playing orders. We can also give the same rank for the players with same points but I thought the playing orders is pure luck and they should be awarded for there luck. One may think I am poised on the playing order but it will good to argue on that.
4. LogsPrinterSystem: I have designed this class so that we can put logs out of the game as a separate block. I have printed logs on the console but we can create a file and then append the logs to the file also that would be good if there any issue in the system we will find them easy to figure out and solve the issue.
5. DiceRepoService : It used the DiceRepo class which is like a repository for storing the players information and game information. DiceRepoService will help the UI Layer by providing the functionalities related to the storage system.
6. DiceRepo : It is just like a database. In this we have used repository but in the real life implementation we can use the database instead of this. We can use JDBC/ hibernate framework for the interface between the database and the UI.
7. PlayingOrderSystem: This class is used to generate the random sequence in which players can have their dice rolling. It is based on the SecureRandom class same as in dice game. I have used set to make unique combination and appended it to the arraylist which can used later in the game to iterate through the playing order.

I have added the low level design diagram according to my project, which would be helpful for understanding the project. I think my design may not be the best but according to the project requirement I have implemented the system and which is functional. But I think it is good to argue over things.