SCALA ASSIGNMENT

PEER LEARNING DOCUMENT

Problem Statement -

Question 1. Bucketise the given array[Double] into buckets having range interval (x, x+0.049).

0.000 - 0.049
0.050 - 0.099
0.100 - 0.149
0.150 - 0.199
0.200 - 0.249
0.250 - 0.299
0.300 - 0.349
0.350 - 0.399
...
...
100.000 - 100.049
Sample -
12.05, 12.03, 10.33, 11.45, 13.50
Output- [12.050-12.099, 12.050-12.099, 10.300-10.349, 11.450-11.499, 13.500-13.549]

Question 2. For given players statistics..

Found the below -

1. Player with the best highest run scored.
2. Top 5 players by run scored.
3. Top 5 players by wicket taken.
4. Rank players with overall performance give weight 5x to wicket taken and (5/100)x to run scored.

Sample -

Year, PlayerName, Country, Matches, Runs, Wickets
2021, Sam, India, 23, 2300, 3
2021, Ram, India, 23, 300, 30
2021, Mano, India, 23, 300, 13

Aayush Sinha

Approach -

Question - 1

The written Scala code represents a program that takes a number as input and rounds it off to 3 decimal points. It then finds the nearest 0.05 multiple lower than the number and prints the range of values that the number belongs to.
The *roundOff* function takes a Double as input and returns a BigDecimal rounded off to 3 decimal points using the setScale method with the RoundingMode.HALF_UP rounding mode.
The *main* method starts by prompting the user to enter a number. It then reads the input.

The operations in the code are performed as below -

- we read the input. using the readDouble method of the scala.io.StdIn object and stores it in the num variable.
- Next, it rounds off the number using the roundOff function and stores it in the roundNum variable.
- Then, it calculates the index of the bucket that the number belongs to by dividing the rounded number by 0.05 and taking the integer part using the toInt method.
- It then calculates the lower bound of the range by multiplying the index by 0.05 and rounding it off to 3 decimal points. It calculates the upper bound by adding 0.049 to the lower bound.
- Finally, it prints the lower and upper bounds of the range in the format "lowerBound - upperBound".

If the input is not a valid Double, the program catches the NumberFormatException and prints "Invalid Input".

Question - 2

The below Scala program stores information about cricket players using a case class called Player and then performs sorting based on the runs scored, wickets taken, and a weighted score (combining runs and wickets).

The getPlayers() function returns a list of Player objects, which contain player information such as year, name, country, matches, runs, and wickets. The data is hardcoded as a list of dummy players.

The sortByRun(players:List[Player]) function takes a list of players and sorts them in descending order by the runs scored.
Similarly, the sortByWicket(players:List[Player]) function sorts players by wickets taken.

Finally, the sortByWeightage(players:List[Player]) function calculates the weighted score for each player and sorts them in descending order.

In the main() function, the program first gets the list of players using getPlayers(), then sorts the players by run, wicket, and weightage using the respective functions. The program then prints out the highest scored player, top 5 players by runs, top 5 players by wickets, and top 5 players by weightage.

Rohith Boodireddy

Approach -

Question - 1

The given code defines a class BucketRange with a single method getRange that takes a Double value num as input, calculates the bucket range for the given value based on the given rules, and prints the fetched limits. The main method creates an object of the BucketRange class and calls the getRange method for an array of Double values.

The operations of the code are as below:

- The getRange method fetches the last two digits of the input value num using the formula ((num * 1000) % 100).toDouble.
- If the value of the last two digits after the decimal point is greater than or equal to 50, the bucket range is calculated using the formula num - (last two digits/1000) + 0.050 to num + (99 - last two digits)/1000. Otherwise, the bucket range is calculated using the formula num - (last two digits/1000) to num + (49 - last two digits)/1000.
- The print statement prints the fetched limits of the bucket range using the f string interpolator to format the values to three decimal places.
- The main method creates an object of the BucketRange class and calls the getRange method for an array of Double values. The for loop iterates over the array of input values and calls the getRange method on each value to print the corresponding bucket range limits.

Question - 2

The given code below defines a case class Player with fields such as year, name, country, matches played, runs scored, and wickets taken. It also defines a class CricketStats with methods for printing player information and for ranking players based on their statistics.

The main method of the program creates a list of Player objects, and then calls the methods of CricketStats to perform various operations on the list of players, such as finding the player with the maximum runs, printing the top 5 players by runs, and so on.

The methods, objects, and thier operations of the code are as below:

- printPlayerInfo(players: List[Player]): This method takes a list of Player objects and prints out their information such as name, country, year, runs, matches, and wickets.
- printPlayerInfoWithRanks(players: List[Player]): This method takes a list of Player objects and prints out their information such as name, country, year, runs, matches,

wickets, and Performance, which is defined as 5 times the number of wickets taken plus 0.05 times the runs scored by the player. The players are also ranked based on their performance, with the highest performer being ranked first.

- The object CricketStats contains utility methods to find the player with maximum runs using the MAXBY function, the top 5 players by runs, top 5 players by wickets, and the top 5 players based on given criteria are sorted by using sortby function. The main method creates an instance of the CricketStats class and calls these methods on a list of Player objects.