Exercise 1:  
Create a Class ScoreAnalyzer with a private instance variable 'runsData' of type LinkedList.  
The linkedlist will store runs scored by individual players in One Day Cricket match.  
Create a constructor and initialize runsData variable.

* Create a method addRunsToList which accepts runs scored by a player as parameter and stores it in the List.
* Create a method calcRunRate which returns the runRate (Consider all 50 overs were played by a team).
* Create a method lowestRunsScored to return the lowest runs scored by a player.
* Create a method displayRuns to display the runs scored by all players.(use for each loop)

Create class TestScoreAnalyzer with a main method.

* Create an object of ScoreAnalyzer
* Use Scanner to get the runs scored and use addRuns method to store the runs
* Display the runs scored by all players
* Display the runrate
* Display the lowest runs scored
* Display the count of player who did batting

Sample input  
Enter Runs  
20 30 150 50 2Sample Output  
Runs Scored : 1-20 2-30 3-150 4-50 6-2  
Runrate : 5.04  
Lowest Runs : 2  
Count of player who batted: 5

[10:08](https://saketmehta199-gag5848.slack.com/archives/D03N54E8JTB/p1661315906634919)

Exercise 2:  
Create a class Patient with private instance variables patientId (int), name(String) and age(int). Include parameterized Constructor and Getters/SettersCreate a class TestPatientList with a main method.

* Create 3 objects of patient and add to an ArrayList
* Sort the List based on Name which should be the natural order and display using for each loop
* Sort the List based on age which should be the alternate sorting order and display using Iterator
* Add all the objects in to a TreeSet and display the TreeSet
* Create a static method getPatientAge which accepts name of the patient and TreeSet of patient. Method should return the age of patient. Invoke this method in the main method.

[10:09](https://saketmehta199-gag5848.slack.com/archives/D03N54E8JTB/p1661315940198859)

Exercise 3:  
Create abstract class PrepaidCard as per the below details

* Has attributes CardNo(int), availableBalance(double), swipeLimit(double)
* Abstract method swipeCard which takes amount as input and returns a boolean.
* Override toString to get object details in a formatted way
* Concrete method rechargeCard which takes amount as input and adds it to availableBalance

Create TravelCard as per below details

* subclass of PrepaidCard
* Has attribute rewardPoints.
* Travel Card implements the Rewardable interface for calculating reward points.
* Interface Rewardable has a method which takes amount as input and returns reward points

The swiped amount should not be more than the swipeLimit or the available balance for Travel Card.Travel Cards are used in USA(1$ = Rs 60).  
When TravelCard is swiped, the swiped amount in dollars is converted in to INR and deducted from the available balance.  Processing charge of 5% of amount is charged for every swipe and is reduced from the balance. 5 Reward points are given for every 100 INR swiped and added to the instance variable rewardPoints.Create a TestCard class. Create objects of TravelCard. Use the travel card by swiping and display the Card details.

**New**

[10:10](https://saketmehta199-gag5848.slack.com/archives/D03N54E8JTB/p1661316047947609)

Exercise 4:

### Stream operations

-----------------------

- Define a POJO class Country with countryId and CountryName as member variables

- Define a POJO class Player with playerName, matchesPlayed, runs, highestScore, Country as member variables

- Create a class StreamOperationsExercise as below

- Create a List of Players, as a static variable, with some player objects

Define following static methods and provide the implementation using Java 8 streams API as given below

- displayPlayers

Display the names of all players

- displayPlayersForCountry(String country)

Display the name of players whose highest score is more than 100 and belong to a particular country

- getPlayerNames

Return a LinkedList containing names of all Players, whose have scored more than 5000 runs, sorted in descending order of names

- getAverageRunsByCountry(String country)

Return the average runs scored by players from a particular Country

- getPlayerNamesSorted

Return a list with names of Players sorted as per country and then by matchesPlayed(descending)

- getPlayerCountry

return a map with the PlayerName and CountryName of all players who have played more than 200 matches

- getMaxRunsPlayer

Return the player who has scored maximum runs

- findPlayer(String name, String country)

Search and return the player for a given name and country

- checkHighScorerByCountry(String country)

Find whether there is any player in the given country who has scored more than 10000 runs. Return a boolean.

Execute the above methods in the main method.