Stream API

```
 /**
* given [1, 2, 3, 4, 5] you should return [1, 4, 9, 16, 25].
*/
        List<Integer> list=Arrays.asList(1,2,3,4,5);
        list.stream().map(i->i*i).map(i-> i+" ").forEach(System.out::print);
        System.out.println();
        System.out.println("============");

        /**
         * given a list [1, 2, 3] and a list [3, 4] you should return [(1, 3), (1, 4), (2, 3), (2, 4), (3, 
3), (3, 4)].
         */
        List<Integer> numbers1 = Arrays.asList(1, 2, 3);
        List<Integer> numbers2 = Arrays.asList(3, 4);
         List<List<Integer>> pairs =numbers1.stream().
                flatMap(i -> numbers2.stream().
                        map(j -> Arrays.asList(i, j))).
                collect(Collectors.toList());
        System.out.println(pairs);

        /**
         * given a list [1, 2, 3] and a list [3, 4] you should return the previous output but the 
pairs whose sum is divisible by 3.
         */
        List<Integer> numbers3 = Arrays.asList(1, 2, 3);
        List<Integer> numbers4 = Arrays.asList(3, 4);
         List<List<Integer>> pairs1 =numbers3.stream().
                flatMap(i -> numbers4.stream().filter(j->(i+j)%3==0).
                        map(j -> Arrays.asList(i, j))).
                collect(Collectors.toList());
        System.out.println(pairs1);


        System.out.println();
        /**
         * Find sum of the values of an arraylist.
         */
        List<Integer> list1=Arrays.asList(1,2,3,4,5,6,7,8,9,10);
        int number = list1.stream().reduce(0, (a,b)->a+b);
        int num = list1.stream().reduce(0, Integer::sum);
        System.out.println("The sum of integer: "+number);
```

```
/**
 * Find multiplication of the values of an arraylist.
 */
int multy = list1.stream().reduce(1, (a,b)->a*b);
System.out.println(multy);

/**
 * find maximum from an arraylist
 */
Optional<Integer> maximum = list1.stream().reduce(Integer::max);
System.out.println(maximum);

/**
 * find minimum from an arraylist
 */
        Optional<Integer> minimum = list1.stream().reduce(Integer::min);
        System.out.println(minimum);

/**
 * Given a list of strings, find the count of strings that have length greater than 5.
 */
List<String> listOf =Arrays.asList("Rajnish","Kumar","Singh");
int count = (int) listOf.stream().filter(i-> i.length()>5).count();
System.out.println(count);

/**
 * Find the sum of all even numbers from a list of integers.
 */
List<Integer> list2=Arrays.asList(1,2,3,4,5,6,7,8,9,10);
int sum = (int) list2.stream().reduce(0, (a,b)->a+b);
System.out.println(sum);

/**
 * Given a list of names, create a new list that contains only the unique names in
uppercase.
 */
List<String> listOfNames =Arrays.asList("Rajnish","Mahesh","Raj","Rajnish","Suresh");
List<String> outputList = listOfNames.stream().distinct().collect(Collectors.toList());
System.out.println(outputList);

/**
 * Given a list of integers, find the average of all the numbers.
 */
List<Integer> list3=Arrays.asList(1,2,3,4,5,6,7,8,9,10);
double average = list3.stream().reduce(0, (a,b)->a+b)/2;
System.out.println(average);
```

```java
        /**
         * Given a list of strings, find the length of the longest string.
         */
        List<String> listOfNames1 =Arrays.asList("Rajnish","Mahesh","Raj","Rajnishwaaaa","Suresh");
        Optional<Integer> count1 = listOfNames1.stream().map(i->i.length()).reduce(Integer::max);
        System.out.println(count1.get());

        /**
         * Remove all duplicates from a list of integers.
         */
        List<Integer> list4=Arrays.asList(1,2,2,2,3,4,5,6,1,7,8,9,2,10);
        List<Integer> outputList1= list4.stream().distinct().collect(Collectors.toList());
        System.out.println(outputList1);

        /**
         * Given a list of students, find the student with the highest grade.
         */
        HashMap<Integer,String> grade = new HashMap<Integer,String>();
        grade.put(89,"Rajnish");
        grade.put(56,"Raj");
        grade.put(100,"Mahesh");
        grade.put(23,"Manoj");

        /**
         * find minimum and maximum from an array in single line
         */
        List<Integer> list5=Arrays.asList(64,89,90,7,8,9,2,10);
        List<Integer> li = list5.stream().sorted().skip(6).collect(Collectors.toList());
        System.out.println(li);

        List<String> list6=Arrays.asList("a","b","a","c","o","a");
        list6.stream().distinct().limit(2).forEach(System.out::println);

        //["Hello", "World"]          you'd like to return the list ["H", "e", "l", "o",
//      "W", "r", "d"].
        List<String> l = Arrays.asList("Rajnish","kumar","singh");
        List<String> list9 = l.stream().map(i->i.split("
")).flatMap(Arrays::stream).collect(Collectors.toList());
        System.out.println(list9);

        //first square divided by three
        List<Integer> list10=Arrays.asList(1,2,3,4,5,6,7,8);
        Optional<Integer> number1 =list10.stream().filter(i->(i*i)%3==0).findFirst();
        System.out.println(number1.get());

        //creating Stream of Strings
        Stream<String> stream = Stream.of("Rajnish","Kumar");
        stream.map(String::toUpperCase).forEach(System.out::println);
        //we can also get empty stream also
        Stream<String> emptyStream = Stream.empty();
        emptyStream.forEach(System.out::println);

        //soting
        List<Integer> list11=Arrays.asList(11,2,3,4,90,6,7,8);
        List<Integer> sort = list11.stream().sorted().collect(Collectors.toList());
        System.out.println(sort);


List<Student> student = Arrays.asList(
                new Student(1,"Rajnish"),
                new Student(2,"Raj"),
                new Student(3,"Manoj"),
                new Student(4,"Suresh"));

        /**
         * Convert a list of objects to a map, using a specific attribute as the key.
         */
        Map<Integer, String> hashmap = student.stream().
                collect(Collectors.toMap(Student::getSid, Student::getName));

        System.out.println(hashmap);

        /**
         * Given a list of strings, concatenate them into a single string, separated by a comma.
         */
        //create a list of Strings
        List<String> studentName = student.stream().
                map(s -> s.getName()).
                collect(Collectors.toList());
        System.out.println(studentName);

        //concatenate them into a single string, separated by a comma.
        String concatString = studentName.stream().reduce( "", (a,b)->a+" ,"+b );
        System.out.println(concatString);

        /**
         * Filter a list of books by a specific genre and return the book titles.
         */
        //create a map
        HashMap<String, String> map1 = new HashMap<>();
        map1.put("Shrimad BhagwatGeeta", "Spritual Book");
        map1.put("Harry Potter", "Fiction Book");
        map1.put("I am the Mind", "Non-Fiction");

        System.out.println(map1);
        //book titles
        List<String> bookTitle = map1.entrySet().stream().map(i->
i.getKey()).collect(Collectors.toList());

        //book genere
        List<String> bookGenere = map1.entrySet().stream().map(i->
i.getValue()).collect(Collectors.toList());

        System.out.println(bookTitle);
        System.out.println(bookGenere);

        /**
         * Given a list of transactions, find the total sum of all transactions for a specific user.
         */
        HashMap<String, List<Integer>> transaction = new HashMap<>();
        transaction.put("Rajnish", Arrays.asList(200,400,550,654));
        transaction.put("Raj", Arrays.asList(200,400,550,654));
        transaction.put("Manoj", Arrays.asList(200,4078,550,654));
        transaction.put("Suresh", Arrays.asList(200,4560,550,654));
        transaction.put("Santosh", Arrays.asList(200,4780,550,684));

        /**
         * Sort a list of employees based on their salaries in descending order.
         */
        HashMap<String, Integer> salary = new HashMap<>();
        salary.put("Rajnish", 2000);
        salary.put("Raj", 8000);
        salary.put("Manoj", 6780);
        salary.put("Suresh",8763);
        salary.put("Santosh", 4322);



        Map<Object, Object>salaryDesc = salary.entrySet().stream()
                .sorted(Map.Entry.comparingByValue())
                .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue));
        System.out.println(salaryDesc);
        /**
         * Given a list of numbers, find the product of all the numbers using the reduce()
method.
         */

        List<Integer> list = Arrays.asList(1,2,3,4,5,6,7,8,9,10);
        int product = list.stream().reduce(1, (a,b)->a*b);
        System.out.println(product);

        /**
         * Group a list of people by their age and create a map where the key is the age and
the value is a list of people of that age.
         */
        //creating a map of objects

        HashMap<Integer,String> hm = new HashMap<>();
        hm.put(20,"Scott");
        hm.put(21,"Smith");
        hm.put(22,"Adam");
        hm.put(23,"Smesh");
        hm.put(20,"Vijendra");
        hm.put(21,"Mangesh");
        hm.put(22,"Manoj");
        hm.put(23,"Satish");
        hm.put(24,"Ram");
        hm.put(25,"Rajnish");

        System.out.println(hm);


        /**
         * Find the maximum and minimum values from a list of doubles.
         */
        List<Double> list3 =Arrays.asList(23.5,90.23,76.9,46.56,89.3,34.9);
        Optional<Double> max = list3.stream().reduce(Double::max);
        System.out.println(max.get());

        Optional<Double> min = list3.stream().reduce(Double::min);
        System.out.println(max.get());

        /**
         * Given a list of strings, find the three longest strings in the list.
         */
        List<String> listOfString= Arrays.asList("aa","aaa","aaaa","aaaaaaa");
        listOfString.stream().sorted(Comparator.reverseOrder()).limit(3).forEach(System.out::println);
        /**
         * Given a list of integers, find the second smallest number.
         */
        List<Integer> list12 = Arrays.asList(1,2,3,4,5,6,7,8,9,10);
        list12.stream().sorted().skip(1).limit(1).forEach(System.out::println);

        /**
         * Given a list of email addresses, filter out the invalid email addresses using regular
expressions.
         */
        /**
         * Partition a list of integers into two lists, one containing even numbers and the other
containing odd numbers.
         */
        List<Integer> list13 = Arrays.asList(1,2,3,4,5,6,7,8,9,10);
        List<Integer> list14 = list13.stream().filter(i-> i%2==0).collect(Collectors.toList());
        System.out.println(list14);
        List<Integer> list15 = list13.stream().filter(i-> i%2!=0).collect(Collectors.toList());
        System.out.println(list15);

        /**
         * Find the distinct characters present in a list of strings.
         */

        List<String> string =Arrays.asList("Hello","world");
        string.stream().map(i->i.split("")).flatMap(Arrays::stream).distinct().forEach(System.out::print);

    }
}
```

Trader class

```
package com.java.Features.Streams;

public class Trader {
            private final String name;
            private final String city;

            //Constructor
            public Trader(String n, String c){
              this.name = n;
              this.city = c;
            }

            //getter method
            public String getName(){
              return this.name;
            }

            //getter method
            public String getCity(){
              return this.city;
            }

            public Trader getTrader() {
              // TODO Auto-generated method stub
              return null;
            }

            //overriding the toString Method
            public String toString(){
              return "Trader:"+this.name + " in " + this.city;
            }

}
```

Transaction class

```java
package com.java.Features.Streams;

public class Transaction {
        private final Trader trader;
        private final int year;
        private final int value;

        //constructor
    public Transaction(Trader trader, int year, int value){
            this.trader = trader;
            this.year = year;
            this.value = value;
    }

    //getter methods
    public Trader getTrader(){
           return this.trader;
    }

    public int getYear(){
           return this.year;
    }

    public int getValue(){
           return this.value;
    }

    //overriding toString()
    public String toString(){
        return "{" + this.trader + ", " +
        "year: "+this.year+", " +
        "value:" + this.value +"}";
    }
}
```

Driver class

```
package com.java.Features.Streams;

import java.util.Arrays;
import java.util.stream.Collectors;
import java.util.*;
import java.util.Comparator;

public class Driver {

   public static void main(String[] args) {
        Trader mario = new Trader("Mario","Milan");
          Trader alan = new Trader("Alan","Cambridge");
          Trader brian = new Trader("Brian","Cambridge");
          Trader raoul = new Trader("Raoul", "Cambridge");
          List<Transaction> transactions = Arrays.asList(
                                            new Transaction(brian, 2011, 300),
                                            new Transaction(raoul, 2012, 1000),
                                            new Transaction(raoul, 2011, 400),
                                            new Transaction(mario, 2012, 710),
                                            new Transaction(mario, 2012, 700),
                                            new Transaction(alan, 2012, 950)
                                                        );
          /**
           * find all the transaction in year 2011 and sort them
           */
          List<Transaction> list = transactions.stream().
                  filter(i->i.getYear()==2011).
                  sorted(Comparator.comparing(Transaction::getValue)).
                  collect(Collectors.toList());
          System.out.println(list);

          /**
           * 1. what are the following unique cities where the trader work
           */
          List<String> cities = transactions.stream().
                  map(transaction -> transaction.getTrader(). getCity()).
                  distinct().
                  collect(Collectors.toList());
          System.out.println(cities);
```

```java
/**
 *2.  find all the traders from Cambridge and sort them by name
 */
List<Trader> namesOfTraders = transactions.stream().
          map(Transaction::getTrader).
          filter(trader -> trader.getCity().equals("Cambridge")).
          distinct().
          sorted(Comparator.comparing(Trader::getName)).
          collect(Collectors.toList());
System.out.println(namesOfTraders);

/**
 * 3. Return a list of String of all trader's name and sorted alphabetically
 */
List<String> traderName = transactions.stream().
          map(transaction->transaction.getTrader().getName()).
          distinct().
          sorted().
          collect(Collectors.toList());
System.out.println(traderName);

/**
 * 4. Return a  String of all trader's name and sorted alphabetically
 */
String traderNames = transactions.stream().
          map(transaction->transaction.getTrader().getName()).
          distinct().
          sorted().
          reduce("",(a,b)->a+" "+b);
System.out.println(traderNames);

/**
 * 5. Are any trader based in milan
 */
boolean checkTraderExist = transactions.stream().
         anyMatch(transaction->transaction.getTrader().getCity().equals("Milan"));
System.out.println(checkTraderExist);

/**
 * 6.Print all transactions’ values from the traders living in Cambridge
 */
transactions.stream().
filter(transaction -> transaction.getTrader().getCity().equals("Cambridge")).
map(Transaction::getValue).
forEach(System.out::println);
```

```
        /**
         * 7.What's the highest value of all the transactions?
         */
        Optional<Integer> max = transactions.stream().
                map(Transaction::getValue).
                reduce(Integer::max);
        System.out.println(max.get());

        /**
         * 8.Find the transaction with the smallest value
         */
        Optional<Integer> min = transactions.stream().
                map(Transaction::getValue).
                reduce(Integer::min);
        System.out.println(min);
    }
}
```