Practical no. 5

FS19CO042

Aim: Using Stream API implement following programs.

- 5.1 Write a generic method to count the number of elements in a collection that have a specific property (for example, odd integers, prime numbers, palindromes).
- 5.2 Write a method which takes a list of words as an argument, groups the words by their lengths and returns the groupings in the form of Map>. (The keys in the map are the lengths and the values are the lists of words of that length.)
- 5.3 Given a List<List> write a program to convert it into a List. (Hint: Use flatMap method in Stream interface)
- 5.4 Given: class Album{ public final String name; public final int yearOfRelease; public final List tracks; }
  class Track{ public final int rating; }
  a) Write a method which takes a list of albums as an argument and returns a list of names of all albums sorted by the year of release.
  b) Write a method which takes a list of albums as an argument and returns a list of names of all albums containing at least one track having rating more than four. The returned list should be sorted by the year of release.

Tool used: Editor (Notepad/Intellij IDE), JDK and JRE

Theory:

Stream In Java

Introduced in Java 8, the Stream API is used to process collections of objects. A stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result.

The features of Java stream are –

- A stream is not a data structure instead it takes input from the Collections, Arrays or I/O channels.
- Streams don't change the original data structure, they only provide the result as per the pipelined methods.
- Each intermediate operation is lazily executed and returns a stream as a result, hence various intermediate operations can be pipelined. Terminal operations mark the end of the stream and return the result.

Different Operations On Streams-

Intermediate Operations:

1. **map:** The map method is used to returns a stream consisting of the results of applying the given function to the elements of this stream.

```
List number = Arrays.asList(2,3,4,5);
List square = number.stream().map(x->x*x).collect(Collectors.toList());
```

2. **filter:** The filter method is used to select elements as per the Predicate passed as argument.

```
List names = Arrays.asList("Reflection","Collection","Stream");
List result = names.stream().filter(s->s.startsWith("S")).collect(Collectors.toList());
```

3. **sorted:** The sorted method is used to sort the stream.

```
List names = Arrays.asList("Reflection","Collection","Stream");
List result = names.stream().sorted().collect(Collectors.toList());
```

Terminal Operations:

1. **collect:** The collect method is used to return the result of the intermediate operations performed on the stream.

```
List number = Arrays.asList(2,3,4,5,3);
Set square = number.stream().map(x->x*x).collect(Collectors.toSet());
```

2. **forEach:** The forEach method is used to iterate through every element of the stream.

```
List number = Arrays.asList(2,3,4,5);
number.stream().map(x->x*x).forEach(y->System.out.println(y));
```

3. **reduce:** The reduce method is used to reduce the elements of a stream to a single value.

The reduce method takes a BinaryOperator as a parameter.

```
List number = Arrays.asList(2,3,4,5);
int even = number.stream().filter(x->x%2==0).reduce(0,(ans,i)-> ans+i);
```

Here ans variable is assigned 0 as the initial value and i is added to it .

Code:

- **5.1 Write a generic method to count the number of elements in a collection that have a specific property (for example, odd integers, prime numbers, palindromes).**

```
import java.util.Arrays;
import java.util.List;
import java.util.stream.Stream;
public class  exp5_1 {
   static class Student {
      String name = "";
      public int roll = 0;
      public int marks = 0;
      public Student(String name, int roll, int marks) {
         this.name = name;
         this.roll = roll;
         this.marks = marks;
      }
   }

   public static <T extends  Number> long  evenNumbers(List<T> list) {
      Stream<T> stream = list.stream();
      return stream.filter(number -> number.doubleValue() % 2 != 0).count();
   }

   public static <T extends Student> long  numberOfPassedStudents(List<? extends Student> list) {
      Stream<T> stream = (Stream<T>) list.stream();
      return stream.filter(student -> student.marks >= 35).count();
   }

   public static void main(String[] args) {
      Student s1 = new Student("Roy", 43, 60);
      Student s2 = new Student("Niel", 44, 49);
      Student s3 = new Student("Leo", 30, 75);
      Student s4 = new Student("lisa", 35, 30);
      Student s5 = new Student("Russ", 40, 28);

      List<Integer> list = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
      List<Student> list1 = Arrays.asList(s1, s2, s3, s4, s5);

      long evenNumbers = evenNumbers(list) ;
      long numberOfPassedStudents = numberOfPassedStudents(list1) ;

      System.out.println("There are "+ evenNumbers +" even numbers.");
      System.out.println(numberOfPassedStudents+" students have passed the exam.");

   }

}
```

Output:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS E:\AssignmentCodes\Java_Practicals> java exp5_1
There are 5 even numbers.
3 students have passed the exam.
PS E:\AssignmentCodes\Java_Practicals>
```

- **5.2 Write a method which takes a list of words as an argument, groups the words by their lengths and returns the groupings in the form of Map>. (The keys in the map are the lengths and the values are the lists of words of that length.)**

```
import java.util.*;
import java.util.stream.Collectors;
import java.util.stream.Stream;
public class exp5_2{
    private static Stream<String> stream;
   public static void main(String[] args)    {
     List<String> strings = Arrays.asList("this", "is", "a", "long", "list", "of",
      "strings", "to", "use", "as", "a", "trial");
          stream = strings.stream();
          Map<Integer, List<String>> lengthMap = stream.collect(Collectors.groupingBy(String::length));
          lengthMap.forEach((k,v) -> System.out.printf("%d: %s%n", k, v));
   }
}
```

Output:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS E:\AssignmentCodes\Java_Practicals> java exp5_1
There are 5 even numbers.
3 students have passed the exam.
PS E:\AssignmentCodes\Java_Practicals>
```

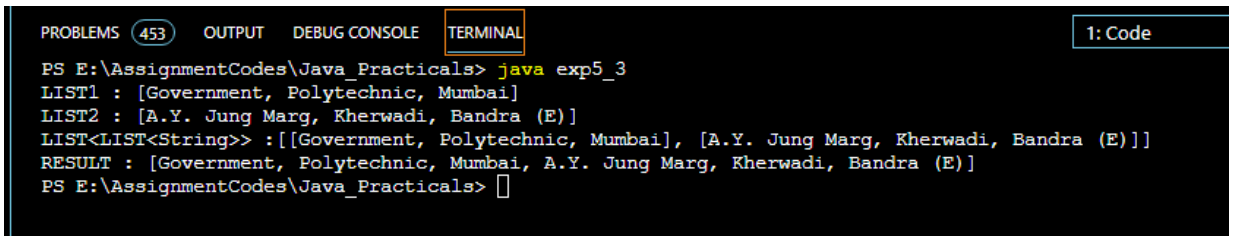
- **5.3 Given a List<List> write a program to convert it into a List. (Hint: Use flatMap method in Stream interface)**

```
import java.util.*;
import java.util.stream.*;

class exp5_3 {
   public static void main(String[] args) {
      ArrayList<String> list1 = new ArrayList();
      list1.add("Government");
      list1.add("Polytechnic");
      list1.add("Mumbai");
      ArrayList<String> list2 = new ArrayList();
      list2.add("A.Y. Jung Marg");
      list2.add("Kherwadi");
      list2.add("Bandra (E)");
      ArrayList<ArrayList<String>> listOflist = new ArrayList();
      listOflist.add(list1);
      listOflist.add(list2);
      System.out.println("LIST1 : " + list1);
      System.out.println("LIST2 : " + list2);
      System.out.println("LIST<LIST<String>> :" + listOflist);
      ArrayList<String> result = new ArrayList();
      listOflist.forEach(result::addAll);
      System.out.println("RESULT : " + result);

   }
}
```

Output:



- **5.4 Given: class Album{ public final String name; public final int yearOfRelease; public final List tracks;**
  class Track{ public final int rating; }

  a) Write a method which takes a list of albums as an argument and returns a list of names of all albums sorted by the year of release.

  b) Write a method which takes a list of albums as an argument and returns a list of names of all albums containing at least one track having rating more than four. The returned list should be sorted by the year of release.

```
import java.util.*;
import java.util.stream.*;
public class exp5_4{
 static class Track{
       public final String name;
       public final int rating;
       public Track(String name, int rating){
         this.name = name;this.rating = rating;
        }
        public String toString(){
          return this.name + " (" + this.rating + ") ";
                }
 }
static class Album{
 public final String name;
private final List<Track> tracks;
private int yearOfRelease;
public int getYear(){
   return yearOfRelease;
}
public List<Track> getTracks(){
   return tracks;
}
public int maxRating(){
     Track maxTrack = tracks.stream().reduce(new Track("temp",0), (maxTrackYet, currTrack) -> {
      if(maxTrackYet.rating < currTrack.rating)
          return currTrack;
       return maxTrackYet;
   });

    return maxTrack.rating;
   }
   public Album(String name, List<Track> trackList, int yearOfRelease){
        this.name = name;
        this.yearOfRelease = yearOfRelease;
        this.tracks = trackList;
     }
     public String toString(){
        return this.name+" ("+this.yearOfRelease+") ";
     }
```

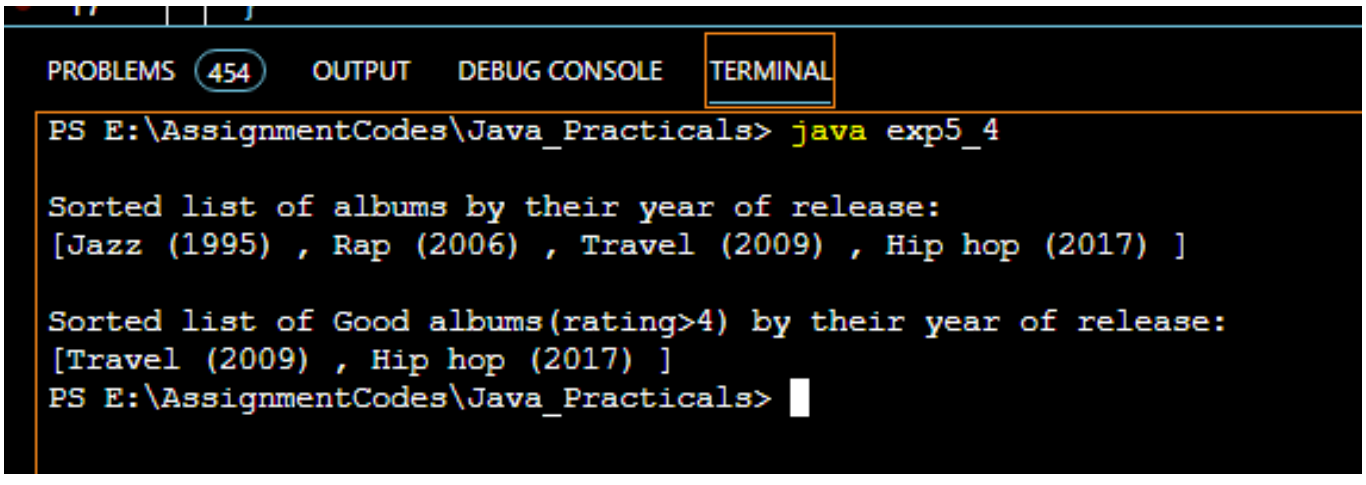
```
}
static List<String>  sortAlbumsByYear(List<Album> albums){
   Stream albumStream = albums.stream();
   Object sorted[] = albumStream.sorted(Comparator.comparingInt(Album::getYear)).toArray();
   List<String> sortedList = new ArrayList<>();
   for(Object obj : sorted)
      sortedList.add(String.valueOf(obj));
   return sortedList;
}
static public List<String> filterGoodAlbums(List<Album> albums){
   List<String> goodAlbums = new ArrayList<>();
   albums.stream().forEach(album -> {
      if(album.maxRating() >4)
         goodAlbums.add(album.toString());
   });
    return goodAlbums;
}
public  static void main(String[] args) {
   System.out.println();
   // Preparing albums and tracks
   String[] travelNames = {"Ve maahi", "Duniya", "Bolna", "Kabira", "Vaaste"};
   int[] travelRatings= {5,6,4,8,5};
   List<Track> travelSongs = new  ArrayList<>();
   for(int i=0; i<travelNames.length; i++)
      travelSongs.add(new Track(travelNames[i], travelRatings[i]));
   Album travelAlbum = new Album("Travel",travelSongs, 2009);

   String[] rapNames = {"Mirchi", "ChalBombay", "Kohinoor"};
   int[] rapRatings = {1, 3,2};
   List<Track> rapSongs = new  ArrayList<>();
   for(int i=0; i<rapNames.length; i++)
         rapSongs.add(new Track(rapNames[i], rapRatings[i]));
   Album rapAlbum = new Album("Rap",rapSongs, 2006);
   String[] hipHopNames = {"Ve maahi", "Duniya", "Bolna", "Kabira", "Vaaste"};
   int[] hiphopRatings = {5,9,9,4,7};
   List<Track> hipHopSongs = new  ArrayList<>();
   for(int i=0; i<hipHopNames.length; i++)
         hipHopSongs.add(new Track(hipHopNames[i], hiphopRatings[i]));
   Album hipHopAlbum = new Album("Hip hop",hipHopSongs, 2017);
         String[] jazzNames = {"Sham", "Masakali","Lovely"};
   int[] jazzRatings = {3,1,2};
   List<Track> jazzSongs = new  ArrayList<>();
   for(int i=0; i<jazzNames.length; i++)
      jazzSongs.add(new Track(jazzNames[i], jazzRatings[i]));
   Album jazzAlbum = new Album("Jazz",jazzSongs, 1995);
   List<String> sortedAlbums =  sortAlbumsByYear(Arrays.asList(travelAlbum, jazzAlbum, hipHopAlbum, rapAlbum));
   System.out.println("Sorted list of albums by their year of release: \n"+sortedAlbums+"\n");

   List<String> goodAlbums =  filterGoodAlbums(Arrays.asList(travelAlbum, jazzAlbum, hipHopAlbum, rapAlbum));

   System.out.println("Sorted list of Good albums(rating>4) by their year of release: \n"+goodAlbums);

}
```



Conclusion: In this experiment, we used various methods of Java Stream API and performed various programs.