90 Points Total. **MPP - Final - Proctortrack**

April 15, 2021

Please write Very Very Clearly! If I can Not read your writing, I can NOT give you credit.

1) True or False (two points each) :

a) Consider the following signature for method 1 (m1)
m1 (? extends T). For example if we have m1 (? extends Employee), this means the argument can be
an Employee object, or any parent of Employee _____

b) A functional interface is an interface that has only one abstract method that must be implemented _____

c) A method reference is used when you want a constant.

d) If you have a List <String> and you want to send this list into the sort method, then you must write a
Comparator to tell the method sort how to do the sorting.

e) An Interface in Java can have a method with a **'default'** implementation that consists of only a left curly
brace and a right curly brace, i.e., { }

f) An Interface in Java can have a method that is **'static'**, with an implementation that consists of only a left
curly brace and a right curly brace, i.e., { }

---

2) Consider the following class :

```
public  class  Company
{
  private  String  companyName;  //  for example Microsoft
  private  String  companyCity;  //  for example Seattle
  private  String companyState;  //  for example Washington
  private  double  totalProfitForThisYear;
```

Assume we have an arraylist of all companies in the United States, maybe 10,000 companies, and assume the arraylist is completely populated with data.

```
  List  <Company>  arr1 = new ArrayList  <Company>;
```

Use **Lambdas and Streams** to do the following. **(7 points each)**.

a) Print out all of the company cities whose city name begins with the letter 'S', in sorted order, and get rid of all the duplicates. Print out only the company city data.

b) Print out the average of all the totalProfitForThisYear.

c) Use the **'reduce'** method to get the total(sum) of All totalProfitForThisYear data.

d) Print out all of the company objects, but if the city name begins with the letter 'W', then capitalize the city name. Remember to print out the whole company object.

e) Create a list of all Company objects whose company state is Iowa, and whose totalProfitForThisYear is less than $7,000,000.

f) Print out All of the company objects with the following restriction. Sort (by companyState) only the objects whose companyState begins with the letter 'G' (All other company objects are Not sorted), and Print out those sorted objects at the very beginning, BEFORE all of the Unsorted objects have been printed out.

g) Save to a list, all of the company objects, but if the city name begins with the letters "Mo", then capitalize all of the city name. For this problem you can **Not** change the original data. We do Not want to change the original data. Hint : Use new …

h) Print out All of the company objects with the following restriction. If the 'companyCity' begins with the letter 'D', then print these first. If the 'companyCity' begins with the letter 'M', then print all of these second, immediately after all of the 'D' cities. Then at the end, after the 'D' cities, and after the 'M' cities, print out All of the rest of the company objects. You Must use only ONE 'System.out.println' statement in your code. All of the objects must be printed out with the same one statement.

****

3) (7 points). Write a method named 'stringToStream' that takes in one argument, a String, and returns a Stream of Character, from the argument String.

The first line is Stream <Character> stringToStream (String s1)

For example, if we make the method call stringToStream (“moon”)

Then this would return a Stream that looks something like [‘m’, ‘o’, ‘o’, ‘n’]

4) (3 points). What is the purpose of the clone ( ) method? What is it used for?

5) (4 points). Write a paragraph or two relating STC/SCI to JavaFX.

6) What is your name?

7) I just added this one. (8 points). For the class above (it has four data members) in questions 2, 3, 4, etc., write the toString method and the equals method for that class. Put this answer at the end of your answer to the last question on the exam.

Good Luck!

****
*
****
*

- 1) True or False (two points each) :

  a) Consider the following signature for method 1 (m1)
  m1 (? extends T). For example if we have m1 (? extends Employee), this means the argument can be
  an Employee object, or any parent of Employee _____

  b) A functional interface is an interface that has only one abstract method that must be implemented _____

  c) A method reference is used when you want a constant.

  d) If you have a List <String> and you want to send this list into the sort method, then you must write a
  Comparator to tell the method sort how to do the sorting.

  e) An Interface in Java can have a method with a ‘default’ implementation that consists of only a left curly
  brace and a right curly brace, i.e., { }

  f) An Interface in Java can have a method that is ‘static’, with an implementation that

consists of only a left
curly brace and a right curly brace, i.e., { }

AMANUEL TESFAMICHAEL
ID 112391
1
a) False Good
b)False -2
C)True -2
d)True -2
e)True Good.
f)false -2

---

2) Consider the following class :

```
public class Company
{
private String companyName; // for example Microsoft
private String companyCity; // for example Seattle
private String companyState; // for example Washington
private double totalProfitForThisYear;
```

Assume we have an arraylist of all companies in the United States, maybe 10,000 companies,
and assume the arraylist is completely populated with data.

```
List <Company> arr1 = new ArrayList <Company>;
```

Use Lambdas and Streams to do the following. (7 points each).

a) Print out all of the company cities whose city name begins with the letter ‘S’, in sorted order, and get rid of all the duplicates. Print out only the company city data.

```
//first lets write the list to an arrays list
//and Assuming having the get methods for all datas
List<Company> list = Arrays.asList(arr1);
                    list.stream()
```

```
                    .filter(c -> c.getCompanyCity().startsWith("S"))
                    .map(Company :: getCompanyCity)
                   .sorted()
                   .distinct()
                   .forEach(System.out::println);
```

Good.

Consider the following class :

```
public class Company
{
private String companyName; // for example Microsoft
private String companyCity; // for example Seattle
private String companyState; // for example Washington
private double totalProfitForThisYear;
```

Assume we have an arraylist of all companies in the United States, maybe 10,000 companies,
and assume the arraylist is completely populated with data.

```
List <Company> arr1 = new ArrayList <Company>;
```

Use Lambdas and Streams to do the following. (7 points each).

b) Print out the average of all the totalProfitForThisYear.

```
//similarly assuming there is getMethods and transferring the ArrayList to Arrays asList
List<Company> list = Arrays.asList(arr1);
            System.out.printf("$%,.2f %n",
                    list.stream()
                        .mapToDouble(Company :: getTotalProfitForThisYear)
                        .average().getAsDouble());
```

Good.

Consider the following class :

```
public class Company
```

{
private String companyName; // for example Microsoft
private String companyCity; // for example Seattle
private String companyState; // for example Washington
private double totalProfitForThisYear;

Assume we have an arraylist of all companies in the United States, maybe 10,000 companies,
and assume the arraylist is completely populated with data.

List <Company> arr1 = new ArrayList <Company>;

Use Lambdas and Streams to do the following. (7 points each).

c) Use the 'reduce' method to get the total(sum) of All totalProfitForThisYear data.

```
//creating the Arrays as list object
List<Company> list = Arrays.asList(arr1);

      System.out.printf("$%,.2f %n ",
                  list.stream()
                    .map(Company :: getTotalProfitForTheYear)    Use mapToDouble
                    .reduce(0,(value1, value2) -> value1 +value2));
```

6 / 7

Consider the following class :

```
public  class  Company
{
  private  String  companyName;  //  for example Microsoft
  private  String  companyCity;  //  for example Seattle
  private  String companyState;  //  for example Washington
  private  double  totalProfitForThisYear;
```

Assume we have an arraylist of all companies in the United States, maybe 10,000 companies,

and assume the arraylist is completely populated with data.

List <Company> arr1 = new ArrayList <Company>;

Use **Lambdas and Streams** to do the following. **(7 points each)**.

d) Print out all of the company objects, but if the city name begins with the letter 'W', then capitalize the city name. Remember to print out the whole company object.

```
// creating an an Arrays asList object with reference list and having getMethods
List<Company> list = Arrays.asList(arr1);

        list.stream()
            .filter(c -> c.getCompanyCity().startsWith("W"))
            .map(c -> c.getCompanyCity().toUpperCase() +" " +
                        c.getCompanyName() + " " +
                        c.getCompanyState() + " " +
                        c.getTotalProfitFoerTheYear())

            .forEach(System.out::println);   // What about the other  non-W  objects?
```


****

Consider the following class :

```
public class Company
{
private String companyName; // for example Microsoft
private String companyCity; // for example Seattle
private String companyState; // for example Washington
private double totalProfitForThisYear;
```

Assume we have an arraylist of all companies in the United States, maybe 10,000 companies,
and assume the arraylist is completely populated with data.

List <Company> arr1 = new ArrayList <Company>;

Use Lambdas and Streams to do the following. (7 points each).

e) Create a list of all Company objects whose company state is Iowa, and whose totalProfitForThisYear is less than $7,000,000.

// assuming three is getMethods and creating anObject arrays asList with ref list
List<Company> list = Arrays.asList(arr1);

```
list.stream()
    .filter(c -> c.getCompanyState().equals("Iowa"))
    .filter(c -> c.getTotalProfitForTheYear() < 7000000)
    .map(Company :: toString). // assuming we have a toString method prints all objects

    .forEach(System.out::println);  // We are supposed to create a  List  at the end!
```

6 / 7

Consider the following class :

```
public class Company
{
private String companyName; // for example Microsoft
private String companyCity; // for example Seattle
private String companyState; // for example Washington
private double totalProfitForThisYear;
```

Assume we have an arraylist of all companies in the United States, maybe 10,000 companies,
and assume the arraylist is completely populated with data.

List <Company> arr1 = new ArrayList <Company>;

Use Lambdas and Streams to do the following. (7 points each).

f) Print out All of the company objects with the following restriction. Sort (by companyState) only the objects whose companyState begins with the letter ‘G’ (All other company objects are Not sorted), and Print out those sorted objects at the very beginning, BEFORE all of the

Unsorted objects have been printed out.

// assuming similarly about the Array as list and having getMethods

```
LIst<Company> list = Arrays.asList(arr1);
        list.stream()
            .filter(c -> c.getCompanyState().startsWith("G")
            .map(c -> c.getCompanyState().toUpperCase() +" " +
                      c.getCompanyName() + " " +
                      c.getCompanyCity() + " " +
                       c.getTotalProfitForTheYear())
             .sorted()
             .forEach(System.out::println);
```

// What about the other non-G objects?

4. 5 / 7

****

Consider the following class :

```
public  class  Company
{
   private  String  companyName;  //  for example Microsoft
   private  String  companyCity;  //  for example Seattle
   private  String companyState;  //  for example Washington
   private  double  totalProfitForThisYear;
```

Assume we have an arraylist of all companies in the United States, maybe 10,000 companies,

and assume the arraylist is completely populated with data.

```
   List  <Company>  arr1 = new ArrayList  <Company>;
```

Use **Lambdas and Streams** to do the following. **(7 points each)**.

g) Save to a list, all of the company objects, but if the city name begins with the letters "Mo", then capitalize all of the city name. For this problem you can **Not** change the original data. We do Not want to change the original data. Hint : Use new …

```
// having the same assumption
        Stream<Company> coStream1 =
                    list.stream()
                        .filter(c -> c.getCompanyCity().startsWith("Mo")
                        .map(c -> new Company (c.getCompanyCity().toUpperCase(),
                                                c.getCompanyName(),
                                                c.getCompanyState(),
                                                c.getTotalProfitForTheYear());
        Stream<Company> coStream2 =
                    list.stream()
                        .filter(c -> !c getCompanyCity().startsWith("Mo")
                        .map(c -.> new Company (c.getCompanyCity(),
                                                c.getCompanyName(),
                                                c.getCompanyState(),
                                                c.getTotalProfitForTheYear());
         List<Company> coList =
                Stream.concat(coStream1, coStream2)
                        .collect(Collectors.toList());

                    coList.forEach(System.out::println);//this last statement forEach could be
omitted because the question said save.
```

Very Good!

Consider the following class :

```
public class Company
{
private String companyName; // for example Microsoft
private String companyCity; // for example Seattle
private String companyState; // for example Washington
private double totalProfitForThisYear;
```

Assume we have an arraylist of all companies in the United States, maybe 10,000 companies,
and assume the arraylist is completely populated with data.

```
List <Company> arr1 = new ArrayList <Company>;
```

Use Lambdas and Streams to do the following. (7 points each).

h) Print out All of the company objects with the following restriction. If the 'companyCity' begins with the letter 'D', then print these first. If the 'companyCity' begins with the letter 'M', then print all of these second, immediately after all of the 'D' cities. Then at the end, after the 'D' cities, and after the 'M' cities, print out All of the rest of the company objects. You Must use only ONE 'System.out.println' statement in your code. All of the objects must be printed out with the same one statement.

//similar assumption about the get methods and array as list object

List<Company> list = Arrays.asList(arr1);

```
Stream<Company> coStream1 =
        list.stream()
          .filter(c -> c.getCompanyCity().startsWith("D"))
          .map(c -> new Company(c.getCompanyCity(),          // Not necessary!
                                c.getCompanyName(),
                                c.getCompanyState(),
                                c.getTotalProfitForTheYear());
Stream<Company> coStream2 =
        list.stream()
          .filter(c -> c.getCompanyCity().startsWith("M"))
          .map(c -> new Company(c.getCompanyCity(),          // Not
necessary
                                c.getCompanyName(),
                                c.getCompanyState(),
                                c.getTotalProfitForTheYear());

    List<Company> coList =
          Stream.concat(coStream1, coStream2)
                .collect(Collectors.toList());
          coList.forEach(System.out::println);
```

What about the non-D and the non-M cities, for example Chicago?

5.5 / 7

(7 points). Write a method named 'stringToStream' that takes in one argument, a String, and returns a Stream of Character, from the argument String.

The first line is Stream <Character> stringToStream (String s1)

For example, if we make the method call stringToStream (“moon”)

Then this would return a Stream that looks something like [‘m’, ‘o’, ‘o’, ‘n’]

Stream<Character> stringToStream(String s1)

```
//writting the string as Arrays as list
System.out.println ????(
            Arrays.stream(s1)  ????  s1 is Not an array, it is a String!

                    .map(c - {for(int i =0, 1 < s1.length(), i++){
                                return  c.s1.charAt[i];
                             } })
                  .collect(Collectors.toList()));
```

Not correct. -7

(3 points). What is the purpose of the clone ( ) method? What is it used for?

The purpose of clone() method is to copy the original data of the objects.
It's mainly used for the security purpose, in order not to be hacked(breached) or changed the original data which are not supposed to be changed.

Good.

```
//=================================================================
========
```

//We have been asked to write toString() and equal() method for the class we have. I guess the class is Company class. //assuming in the same class if it is in different class we use this..

```
public  boolean isEqual(Object obj){
```

```
if(obj == null || this.getClass() != obj.getClass){
            return false;
}
Company co =(Company)obj;
 return  companyName.equals(obj.companyName) &&
             companyCity.equals(obj.companyCity) &&
            companyState.equals(obj.companyState) &&
           totalProfitForTheYear == obj.totalProfitForTheYear;
}
```

Very Good!

```
//=================================================================
============
//2nd toString() method
public string toString(){
   String result = " ";
   result = "Company Name" + companyName +
                "Company City" + companyCity +
                "Company State" + companyState+
                "Total Profit" + totalProfitForTheYear;
return result;
}
```

Good.

```
//there is one more we have been asked for extra credit.
//=================================================================
==========
//we have been asked also if we can make a deep clone if the OrderClass have
List<OrderLine> list as an extra credit
Date d1;
List<OrdeLIne> orderList;
public object clone(){
    Order o1 = (Order)super.clone();
     o1.d1 = (Date)d1.clone();    // What about orderList?
 return o1
```

```
}
//the list can have its own clone() Method, this could be the deep copy.
public object clone(){

public static List<OrderLIne> cloneList(List<OrderLine> orderList){
        List<OrderLine> clonedList = new ArrayList<OrderLine>(orderList.size());
                for(OrderLine oL : orderList){
                    clonedList.add(new OrderList(oL));    ????
                }
            return clonedList;
}
```

****

(4 points). Write a paragraph or two relating STC/SCI to JavaFX.

Good Luck!

JavaFX in java programming helps the users to use many actions in a friendly way, it is related with the Graphical interfaces. some of the methods is being used are event handling methods and event listening methods. It can be called one out of many abstractions used to simplify many details and complicated things in java programming. Through this technique(javaFX) many things can be done.

Transcendental Consciousness is an interface in which gives. an experience the stillness of ones awareness, it is a moment helps one to know self, ease or gives one relief from stress. in general it is an abstraction of ease pain, stress,.. relieved through a simple technique called Transcending. This technique can be done easily by smooth breathing, closing eyes and sitting comfortably. Taking as routing daily activity, at least twice a day each 20 mins.
