Student Name:-_____ **Student ID:** _____

CS401: Modern Programming Practices Final Exam

Computer Professionals Program

Date: 7 - 18 -2024

Part 1: Theory	Part 2: Cognitive skills			
Q1 (12)	Q2 (3)	Q3 (14)	Q4 (5)	Q5 (6)

- This exam contains 5 questions on 10 papers

Question 1:

A) Write 'True' or 'False' for the following statements: **(4 points – each 1)**

A.1- If a class implements multiple interfaces that defines default methods with the same signature, the compiler will choose one of them.	

A.2- Given the following code, it will produce a compiler error?
 	
`public class S {`
`public static void main (String []args) {`
`List<? super Integer> list = new ArrayList<>();`
`list.add(5);`
`}`
`}`	

A.3- Given the following code, it is eligible to be a functional interface?
 	
`public interface InterfTest {`
`public void print();`
`public String toString();`
`abstract void doSomething();`
`}`	

A.4- Applying streams in parallel is always more efficient than a sequential approach.	

B) What is the output of the following codes: **(4 points – each 1)**

`public class Test1 {`
`public static void main(String[] args) {`
 	
`String name = "Hafez";`
`Optional<String> nameO = Optional.ofNullable(name);`
`System.out.println(nameO.orElse( getName() ));`
`}`
`public static String getName(){`
`System.out.println(" Remember this example ;) ");`
`return "No Name";`
`}`
`}`	

`public class Test2 {`
`public static void main(String[] args) {`
`List<Integer> grades =`
`new ArrayList<>(Arrays.asList(3,4,3));`
`List<Student> students = new ArrayList<>(Arrays.asList(`
`new Student(111,"Yasmeen",grades),`
`new Student(112,"Mira",grades),`
`new Student(113,"Zaina",grades)`
`));`
`var result = students.stream()`
`.flatMap(s -> s.grades.stream())`
`.mapToInt(x -> x)`
`.sum();`
`System.out.println(result);`
`}`
`}`	

`public class Test3 {`
`  public static void main(String[] args) {`
`Stream<Integer> stream1 = Stream.iterate(1, n -> n + 1);`
`stream1`
`  .limit(5)`
`  .forEach(System.out::print);`
`  }`
`}`	

`public class Test3 {`
`  public static void main(String[] args) {`
`IntStream`
`  .range(1, 10)`
`  .skip(5)`
`  .forEach(x -> System.out.print(x));`
`  }`
`}`	

C) Explain the following statements **(4 points – each 2)**

(Choose 2 questions from Q1-C)

C.1- When overriding the `equals()` method from the Object class, why is it recommended to be consistent and override the `hashcode()` method too?

C.2- Why did Java introduce the **Optional** class?

C.3- Using the '*extends wildcard*' has some limitations of it could not insert but could get data. Why does the compiler give a compile error when trying to insert?

Question 2: Write a code to complete the following requirements: (3 points)

Write a code to define an Enum class 'Seasons' that has four fields (SPRING, SUMMER, FALL, WINTER). Assign the following fields with the given average temperatures in Celsius, SUMMER=30, WINTER=0, any fields not assigned should be by default =15.

```
public enum States {




}
```

Question 3: Write codes for the following requirements using streams API:

A) Write an implementation for `startsWithTarget()` method using streams API that takes a list of strings and a 'target' letter that returns a list of all the name that start with the given 'target' letter (upper case) and the list should be sorted. **(3 points)**

Example → `List = {"Adam", "Ibrahim", "Julliane", "Mike", "Moe", "John", "Mark"}`
`startsWithTarget(List, 'M');`

Result → Mark, Mike, Moe

```
public List<String> startsWithTarget(List<String> names, char target){
```

}

B) Write an implementation for `sumTwoNumbers()` method using streams API that takes an array of numbers, and only takes the **first** two numbers that are greater or equal to 30, then returns their sum. **(3 points)**

Example → {15, 45, 25, 50, 20, 35, 60, 10}

Result → 95

```
public int sumTwoNumbers(int [] nums){
```

}

C) Given an integer input, write an implementation for `streamSumOfSquares()` method using streams API that will return the sum of squares of all numbers from 1 to the given number (inclusive). **(3 points)**

Example → 3

Result → (1^2 + 2^2 + 3^2) = 14

```
public Integer streamSumOfSquares(int n){
```

}

D) According to the given classes (**Owner, Building, Apartment**) in page [4] in your external sheet, write the functionality for the `allOwnerTotalApartmentRent()` method using Streams API to calculate the sum of all apartments for all owners, by taking a list of **Owner**. Below is an implementation yet to be completed. **(3 points)**

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class Main {
     public static void main(String args[]) {
        List<Apartment> apts1 = Arrays.asList(new Apartment(100),new Apartment(200),new Apartment(300));
        List<Apartment> apts2 = Arrays.asList(new Apartment(500),new Apartment(300),new Apartment(100));
        List<Apartment> apts3 = Arrays.asList(new Apartment(400),new Apartment(100),new Apartment(500));

        List<Building> buildings1 = Arrays.asList(new Building(apts1),new Building(apts2));
        List<Building> buildings2 = Arrays.asList(new Building(apts3));

        Owner owner1 = new Owner(buildings1);
        Owner owner2 = new Owner(buildings2);
        List<Owner> ownersList = new ArrayList<>();
        ownersList.add(owner1);
        ownersList.add(owner2);

    System.out.println(allOwnerTotalApartmentRent(ownersList));

    }
    public static double allOwnerTotalApartmentRent(List<Owner> owners){
```

Should print the total of all apartments = 2500

E) Determine the intermediate operations and terminal operations in the following: **(2 points)**

```
IntStream
        .range(1, 10)
        .filter(x -> x >7)
        .skip(1)
        .limit(2)
        .forEach(x -> System.out.print(x));
```

Intermediate operation/s: ……

Terminal operation/s: …….

Question 4:

A) Given the stream in the box below that calculates the sum of all salaries after deducting taxes (12%) from each employee. To make this stream reusable, map it in the 'LambdaLibrary' class with a suitable functional interface and name it netSalary in order to call it as shown in the 'Main' class below. It should take a list of Employees and return a result. Employee class is provided in external sheet page [3], just for reference. **(2 points)**

```
list.stream()
.map(e -> e.getSalary())
.map(e -> e * 0.88)
.reduce(0.0,(x,y)->x + y);
```

.

```
package Final;

import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        List<Employee> emps = new ArrayList<>();
        emps.add(new Employee("Zaineh",7000));
        emps.add(new Employee("Yasmeen",6000));
        emps.add(new Employee("Dean",3000));

        System.out.println(LambdaLibrary.netSalary.apply(emps));
        // The result should return the total of all updated salaries
    }
}
```

```
package Final;

public class LambdaLibrary {
```

B) Given the following codes. The `calcUniqueItems()` method in the 'ImpToStreams' class is written in an imperative style, rewrite this method in `calcUniqueItemsStream()` using Streams API and lambda expressions that will produce the same result. **(3 points)**

```
package Final;
import java.util.ArrayList;
import java.util.List;

public class ImpToStreams {
    static int ItemCount = 0;
    static List<String> TrackList = new ArrayList<>(); //start empty

    public static long calcUniqueItems(List<Item> items) {
        for (Item i: items) {
                if (TrackList.contains(i.getName())) {
                    //do Nothing
            }
                else {
                    TrackList.add(i.getName());
                    ItemCount++;
                }
            }
        return ItemCount;
    }


    public static long calcUniqueItemsStream (List<Item> items) {
```

```
package Final;

public class Item {
    String name;
    public Item(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
}
```

Question 5: Write the implementation code for the following: (6 points – each 2)

A) Write a generic method `mySwap()` to exchange the positions of two different elements in an array.

B) Write a generic method called `displayElements()` that accepts a `Collection` of any type and prints out each element it contains.

C) Rewrite the class to be generic for any type:

```
public class Pair {
    private String first;
    private String second;

    public Pair(String first, String second) {
        this.first = first;
        this.second = second;
    }

    public String getFirst() {
        return first;
    }

    public String getSecond() {
        return second;
    }
}
```