Student Name:-_____ Student ID: _____

CS401: Modern Programming Practices
Final Exam

Computer Professionals Program

Date: 06 - 16 -2022

Part 1: Theory	Part 2: Cognitive skills			
Q1 (10)	Q2 (6)	Q3 (13)	Q4 (5)	Q5 (6)

- This exam contains 5 questions on 9 papers
- You should have an external sheet that has 4 pages

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 force you to override this method for the class. **( )**

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 InterfTest2 {
	public void print();
	public String toString(); }
```

( )

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

B) Explain the following statements (6 points – each 2)

(Choose 3 questions from Q1.B)

B.1- Given the following code, is it efficient to use parallel streams in this case? Explain your answer.

```
List<Integer> nums = new ArrayList<>(Arrays.asList(2,5,7,9,1,4));
nums.parallelStream()
        .filter(n -> n>5)
        .findFirst();
```

B.2- What was the motivation for Java 8 to introduce 'default' and 'static' methods to be implemented in interfaces.

B.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?

B.4- Why did Java introduce the **Optional** class, what is the problem and what are they solving?

Question 2: Write a code to complete the following requirements: (6 points – each 3)

A) Write a code to define an Enum class 'States' that has four fields (IA, PE, CA, VA). Assign the following fields with the given tax rates, CA=0.07, VA=0.04, any fields not assigned should be by default =0.3.

```
public enum States {

}
```

B) Given the class (Person) below, create a static method that will take a list of type 'Person' and sort it according to the following criteria: [*comparison should be by* **`firstName, lastName,`** *then* **`salary`**]. Write the code in the Person class using a **functional** approach. You may assume that the getters and setters are defined.

```
public class Main {

    public static void main(String[] args) {
        Person p1 = new Person("Saul","Goodman",80000);
        Person p2 = new Person("Kim","Wexler",100000);
        Person p3 = new Person("Howard","Hamilton",250000);

        List<Person> people = new ArrayList<>(Arrays.asList(p1,p2,p3));
        Person.sortPeople(people);
        System.out.println(people);
    }
}
        //Output
        [Howard Hamilton: 250000.0, Kim Wexler: 100000.0, Saul Goodman: 80000.0]
```

```
package Final;

public class Person {
    final private String firstName;
    final private String lastName;
    final double salary;

    public Person(String fname, String lname, double salary) {
        this.firstName = fname;
        this.lastName = lname;
        this.salary = salary;
    }
    @Override
    public String toString() {
        return firstName + " " + lastName + ": " + salary;
    }

      // getters …
      // setters …
```

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 `avgThreeNumbers()` method using streams API that takes an array of numbers, and only takes the first three numbers that are greater or equal to 50, then returns the average of them. **(2 points)**

Example → `{90, 45, 50, 30, 80, 70, 60, 40, 90}`

Result → `73.33`

```
public double avgThreeNumbers(int [] nums){

}
```

C) Given an integer input, write an implementation for `streamFactorial()` method using streams API that will return the factorial of the given number. **(2 points)**

Example → **4**

Result → = (4×3×2×1) = **24**

```
public Integer streamFactorial(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: **(3 points)**

```
IntStream
        .range(1, 10)
        .filter(x -> x >7)
        .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)**

```java
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) {
```

```java
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 3)**
** (Choose 2 questions from Q5)*

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

B) Write a generic method called `printAll()`, that will take a ‘List’ of any type and print out all its objects.

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

```
public class DefinedPair {
    private int key;
    private String value;


    public DefinedPair(int key, String value) {
             this.key = key;
             this.value = value;

    }
    public int getKey()   { return key; }
    public String getValue() { return value; }
}
```