Student Name:-_____ Student ID: _____

CS401: Modern Programming Practices
Final Exam

Computer Professionals Program

Date: 2 - 6 -2020

Part 1: Theory	Part 2: Cognitive skills			
Q1 (10)	**Q2 (8)**	**Q3 (10)**	**Q4 (6)**	**Q5 (6)**

- This exam consists of 5 questions on 10 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)
(Only answer 3 questions, if you answer all four, the first three will be taken).

B.1- When overriding the `equals()` method from the Object class, there are two common strategies, explain how a potential error could occur when applying "same-classes" strategy.

B.2- Java 8 introduced default and static methods to be implemented in an interface, how does this affect the functionality of an Enum class.

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

B.4- Why was Java motivated to introduce functional programming in its source codes?

Question 2: Write a code to complete the following requirements: (8 points – each 2)

A) Write a code to define an Enum class called 'States' that has four fields (IA, PE, CA, VA). Assign the following fields with the given tax rates, CA=0.072, VA=0.043, any fields not assigned should be by default =0.06.

```
public enum States {

}
```

B) Given the following code, write an addition to this class to make it execute the `countHowManyAs( )` method in parallel using two threads to read the sample string in the code below and printing out the answer.

```
public class MultiCount {
    private static String sample= "I am a student in the compro program at MIU";
```

```
public static int countHowManyAs (String s) {
            int sum = 0;
            for (char i : s.toCharArray()) {
                  if (i == 'a' || i=='A')
                        sum++;
            }
            return sum;
                        }
```

```
      public static void main(String[] args) {
```

```
      } }
```

C) Write a code to create a user-defined annotation called @PersonIncharge that will only be applied on methods, and its value could be accessed at run-time. The value should take a 'name' of the person, if a name is not inserted, it should be "unknown".

D) Given the following stream, develop a unit test on the lambda expression shown below. Full implementation of 'Account' and 'Customer' classes are in the external sheet page 3.

- *To create an object you may use this simple code:*

```
// Customer cust1 = new Customer("000", "XXX", "XXX");
// cust1.getCheckingAccount().updateBalance(000);
```

```
public static List<Customer> specialAccounts(List<Account> accounts) {
        return accounts.stream().filter(a -> a.getBalance() > 50)
                        .map((Account a) -> a.getCustomer())
       .sorted(Comparator.comparing((Customer c) -> c.getLastName()))
       .collect(Collectors.toList());
}
```

Question 3: Write codes for the following requirements using streams API:
(10 points – each 2)

A) Given a list of Strings, write a method called `namesWithM( )` using streams API that returns a list of all strings that start with the letter 'M' (upper case) and sorted with no duplications.
Example → `{"Moe", "Adam", "Ibrahim", "Julliane", "Mike", "Moe", "John", "Mark"}`
Result → Mark, Mike, Moe

B) Given an array of numbers, write a method called `avgFirstFive( )` using streams API will take that only takes the first 5 numbers that are above 50, then returns the average of them.
Example → `{90, 45, 50, 30, 80, 70, 60, 40, 90}`
Result → 70

C) Given an integer input, write a method using streams API that will factorial the factorial of the given number.
Example → 4
Result → 24

D) Write the following code as a stream pipeline with lambda expressions.

```
for (Person p : roster) {
    if (p.getGender() == Person.Sex.MALE) {
        System.out.println(p.getName());
    }
}
```

E) Determine the intermediate and terminal operations in the following code:

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

Intermediate operations: ……

Terminal operations: …….

Question 4: Answer the following questions **(6 points – each 3)**

A) Given the Employee class (full implementation in the external sheet on page 4).

Create a reusable lambda function using the lambda expression below to take a salary integer as input and calculate the final salary after deducting taxes. Name the function netSalary( ).

```
(Employee e) -> e.getSalary() * .88);
```

Write the implementation to apply for all employees that exist in a List of Employees.

B) From the Employee class (ext. p4), given the following code that was written in an imperative style, rewrite the code to be in declarative-style that should have the same functionality.

```
public class Main {

	public static void main(String[] args) {
		Employee e1 = new Employee("John", 10000);
		Employee e2 = new Employee("John", 20000);
		Employee e3 = new Employee("Moe", 30000);
		Employee e4 = new Employee("Jullz", 40000);

		List<Employee> elist = new ArrayList<>();
		elist.add(e1);
		elist.add(e2);
		elist.add(e3);
		elist.add(e4);
	// REWRITE THIS PART IN A DECLARATIVE STYLE.
		for (Employee e : elist) {
			int sal = e.getSalary();
			double newSal = sal * 0.88;
			String name = e.getName();
			String nameTitle = "Mr." + name;
			System.out.print(nameTitle +" ");
			System.out.println(newSal);
		}

				}
```

Question 5: Answer the following questions **(6 points – each 2)**

A) Write a generic method called `swapE( )` 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 OpenPair {
       private int key;
    private String value;


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

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