CORE_JAVA_DXC_2022_001

Employee Id: _____
Duration: 3.0 Hrs **Max. Marks 60**

Instructions:

1. First 15 minutes are provided to read & understand the test paper.
2. Next 45 minutes for Question Part1 and 2.0Hrs for Part2
3. Create a project with Core followed by your first name and employee ID.
4. Take care of Plagiarism
5. Referring Java Documentation is allowed.
6. **Marks will only be awarded to working functionalities**

Question Part1

Q1. Multiple Choice Questions:

Q1.1

```
int num=7;
int[][] arr = { {7, 3, 6, 4},
                {9, 2, 0, 5},
                {1, 4, 3, 8}};
for (int j = 0; j < arr.length - 1; j++){
        for (int k = 0; k < arr[0].length; k++){
                if (arr[j][k] == num){
                        System.out.println(arr[j][k]);
                }
        }
}
```

Which of the following values of num does the code segment not work as intended?
A: num = 5 B: num = 6 C: num = 7 D: num = 8 E: num = 9

Q1.2

```
int[][] arr = {{6, 2, 5, 7},
               {7, 6, 1, 2}};
for (int j = 0; j < arr.length; j++){
        for (int k = 0; k < arr[0].length; k++){
                if (arr[j][k] > j + k){
                        System.out.println("!");
                }
        }
}
```

How many times will "!" be printed when the code segment is executed?
A: 0times B: 2times C: 4times D: 6times E: 8times

Q1.3 Consider the following class definitions.

```
public class Road{
        private String roadName;
        public Road(String name){
        roadName = name;
        }
}

public class Highway extends Road{
        private int speedLimit;
        public Highway(String name, int limit){
                super(name);
        speedLimit = limit;
        }
}
```

The following code segment appears in a method in another class.

```
Road r1 = new Highway("Interstate 101", 55); // line 1
Road r2 = new Road("Elm Street"); // line 2
Highway r3 = new Road("Sullivan Street"); // line 3
Highway r4 = new Highway("New Jersey Turnpike", 65); // line 4
```

Which of the following best explains the error, if any, in the code segment?
Line 1 will cause an error because a Road variable cannot be instantiated as an object of type Highway.

Line 2 will cause an error because the Road constructor is not properly called.
Line 3 will cause an error because a Highway variable cannot be instantiated as an object of type Road.
Line 4 will cause an error because the Highway constructor is not properly called.

Q1.4 Consider the following class definitions.

```
public class Drink{
// implementation not shown
}

public class Coffee extends Drink{
// There may be instance variables and constructors that are not shown.
// No methods are defined for this class.
}
```

The following code segment appears in a method in a class other than Drink or Coffee.

```
Coffee myCup = new Coffee();
myCup.setSize("large");
```

Which of the following must be true so that the code segment will compile without error?
A:The Drink class must have a public method named getSize that takes a String value as its parameter.
B: The Drink class must have a public method named getSize that takes no parameters.
C: The Drink class must have a public method named setSize that takes a String value as its parameter.
D: The Drink class must have a public method named setSize that takes no parameters.
E: The Drink class must have a String instance variable named size.

Q1.5 Consider the following class definitions.

```
public class Apple{
public void printColor(){
System.out.print("Red");
        }
}
public class GrannySmith extends Apple{
public void printColor(){
System.out.print("Green");
        }
}
public class Jonagold extends Apple{
// no methods defined
}
```

The following statement appears in a method in another class.

```
someApple.printColor();
```

Under which of the following conditions will the statement print "Red"?
When someApple is an object of type Apple
When someApple is an object of type GrannySmith
When someApple is an object of type Jonagold

I only
I and II only
I and III only
II and III only
I, II, and III

Q1.6 Consider the following class definitions.

```
public class Pet{
        public void speak(){
        System.out.print("pet sound");
        }
}
public class Dog extends Pet{
        public void bark(){
                System.out.print("woof woof");
        }
        public void speak(){
                bark();
        }
}
public class Cat extends Pet{
        public void speak(){
                System.out.print("meow meow");
        }
}
```

The following statement appears in a method in another class.
myPet.speak();
Under which of the following conditions will the statement compile and run without error?
When myPet is an object of type Pet
When myPet is an object of type Dog
When myPet is an object of type Cat

I only I and II only II and III only I, II, and III

Q2. Write output of the following code after execution:
Q2.1.
```
int num = 0, dum = 8;
while (num < 3){
        dum = 8;
        while (dum > 2){
                System.out.print(1 + dum);
                dum -= 2;
        }
        num++;
}
```

Q2.2.
```
int num = 5;
int dum = 5;
while (num > 1 && dum < 8){
        System.out.print(num + "-");
        num--;
        dum++;
}
```

Q2.3.
```
int rows = 4, columns = 3;
while (rows > 0){
        while (columns > 0){
                System.out.print("*");
                columns--;
        }
        rows--;
        columns = 3;
        System.out.println(); // moves cursor to next line
}
```

Q2.4.
```
boolean finished = false;
int num = 0;
while (!finished){
        num += 1;
        if (num % 5 == 0)
                finished = true;
        num += 201;
}
System.out.println(num);
```

Q2.5.
```
for (int i = 1; i < 5; i++){
        if (i == 2)
                continue;
        System.out.print(i + " ");
}
```

Q2.6.
```
for (int i = 1; i < 4; i++)
        System.out.print(i + " ");
System.out.print("\nbye");
```

Question Part2

Q1. An insurance company calculates the premium to be paid by drivers per year based on the driver's age and experience of driving.

Create a package called **com.dxc.insurance** containing the following classes.

- **Driver** is a class containing personal information of the driver such as
  - *name* String //name of the driver
  - *age* int //age of the driver

  It also contains the following methods:
  - *Driver* constructor to initialize the name and age of the person passed as parameters to it.
  - *displayInfo* to display the drivers details.

- TooYoungException is a user-defined Exception class that overrides the *toString* method to display an appropriate message if the driver is too young to be insured based on the PremiumCalculator class defined below.
- PremiumCalculator is a class having the following features
  - A static variable called *minAge* that specifies the minimum age required for the driver to be eligible for insurance (assume value 18).
  - *static int calcInsurance(Driver)* that calculates the premium and returns the same based on the criteria specified below.
    - This method first compares the age of the driver as against the *minAge*. If the driver's age is less than the required *minAge*, the driver cannot be insured. In this case, throw an exception called TooYoungException displaying an appropriate message.
    - Otherwise, the driver is eligible for insurance. Now calculate the premium based on the years of experience. For this, first calculate the drivenYears (a local variable) based on the age, that is, (age - minAge), and if the *drivenYears* is above 4 years, the premium to be paid per year is Rs. 600/-. But if drivenYears is less than or equal to 4 years, then the driver is charged Rs. 1000/-.

Outside the package, create a class **Insurance** that imports the above package and implements the main method. Within the main method, accept the driver's details from the user and create an instance of the **Driver** class. Calculate the insurance amount and display the driver's details along with the amount being charged for insurance. Handle exceptions wherever applicable.

Q1a	Package creation and its usage in the *Insurance* class	2 marks
Q1b	Proper creation of the *Driver* class	2 marks
Q1c	Proper triggering of *TooYoungException* class	3 marks
Q1d	Implementing the PremiumCalculator class	3 marks
Q1e	Implementing the *Insurance* class with all functionalities	4 marks
Q1f	Following proper coding conventions	1 mark

TOTAL	**15 marks**

Q2. [Marks:10]

Define an interface called **StudentInt** in package **dxc.student** with following methods :

acceptStudentMarks() // to accept the details of the students
displayStudentMarks() // display the details of the students

Define a class **StudentMarks** in package **dxc.student.marksinfo** with following properties:

rollNo (int) // unique identifications of the student
semester (String) // Sem-I to Sem-IV
cProgramming (float) // between 1 to 100
coreJava (float) // between 1 to 100
advanceJava (float) // between 1 to 100
totalMarks (float) // total marks in the subjects

Define proper constructors. **StudentMarks** class implements **StudentInt** interface.

Define a service class **StudentMarksService** to store or retrieve student marks.
The menu should display as follows:

1. **Store Student Marks**
2. **Display Student Marks**
3. **Exit**

If the user choice is 1, it should ask all necessary details of student from user. The rollno for every student must be unique. Before storing student details in file *StudentInfo.txt*, check whether details for the same rollno is already present in the file *StudentInfo.txt* or not. If already present then generate a user defined exception and display error message as **"Roll No already exist. Try again"**. Otherwise, store student details in the file *StudentInfo.txt*.

If the user choice is 2, it displays all the records in the file

If entered 3, the program should close.

	Steps	**Marks**
Q1	Design **StudentInt** interface with package	2
Q2	Design **StudentMarks** class with package	2
Q3	**StudentMarksService** with menu	2
Q4	Searching **rollNo** from file properly	2
Q5	Writing the user defined exception class	2

Q6	Handling the exception properly	2
Q7	Write records to the file properly	2
Q8	Coding conventions & Indentation	1