CSC 110 2.0 Object Oriented Programming
Tutorial 05

Instructions:

- All questions must be attempted and answers submitted in a handwritten document, **on or before, 4.00pm on Monday, 19th August 2019, to the Department Office**.
- You must indicate your **Index Number and the Tutorial Class** to which you belong to (LCS1/ LCS2/ NFC3.1) clearly on the front page of your submission.

A. Understanding Code Segments

For each of the code segments below, write the output of the main method in the respective Driver Class.

1. Inheritance

```
public class Test {
     public static void main(String[] args) {
          new A(); new B();
     }
}

class A {
     int i = 7;

     public A() {
          setI(20);
          System.out.println("i from A is " + i);
     }

     public void setI(int i) {
          this.i = 2 * i;
     }
}

class B extends A {
     public B() {
          System.out.println("i from B is " + i);
     }

     public void setI(int i) {
          this.i = 3 * i;
     }
}
```

2. Suppose we have a *Car* class with several attributes (instance variables).

```
public class Car {
    private String name;
    private String engine;

    public static int numberOfCars;
```

```
    public Car(String name, String engine) {
        this.name = name;
        this.engine = engine;
        numberOfCars++;
    }

    // getters and setters
}
```

```
public class TestCar {
     public static void main(String[] args) {
          Car c1 = new Car("Honda Civic","1000CC");
          Car c2 = new Car("Toyota C-HR","1200CC");
          Car c2 = new Car("Mitsubishi Eclipse","1500CC");

          System.out.println("Number of Cars: " + Car.numberOfCars);
     }
}
```

3. Consider the following class.

```
public class Sum {

    public int sum(int x, int y)
    {
        return (x + y);
    }

    public int sum(int x, int y, int z)
    {
        return (x + y + z);
    }

    public double sum(double x, double y)
    {
        return (x + y);
    }

    public static void main(String args[])
    {
        Sum s = new Sum();
        System.out.println(s.sum(10, 20));
        System.out.println(s.sum(10, 20, 30));
        System.out.println(s.sum(10.5, 20.5));
    }
}
```

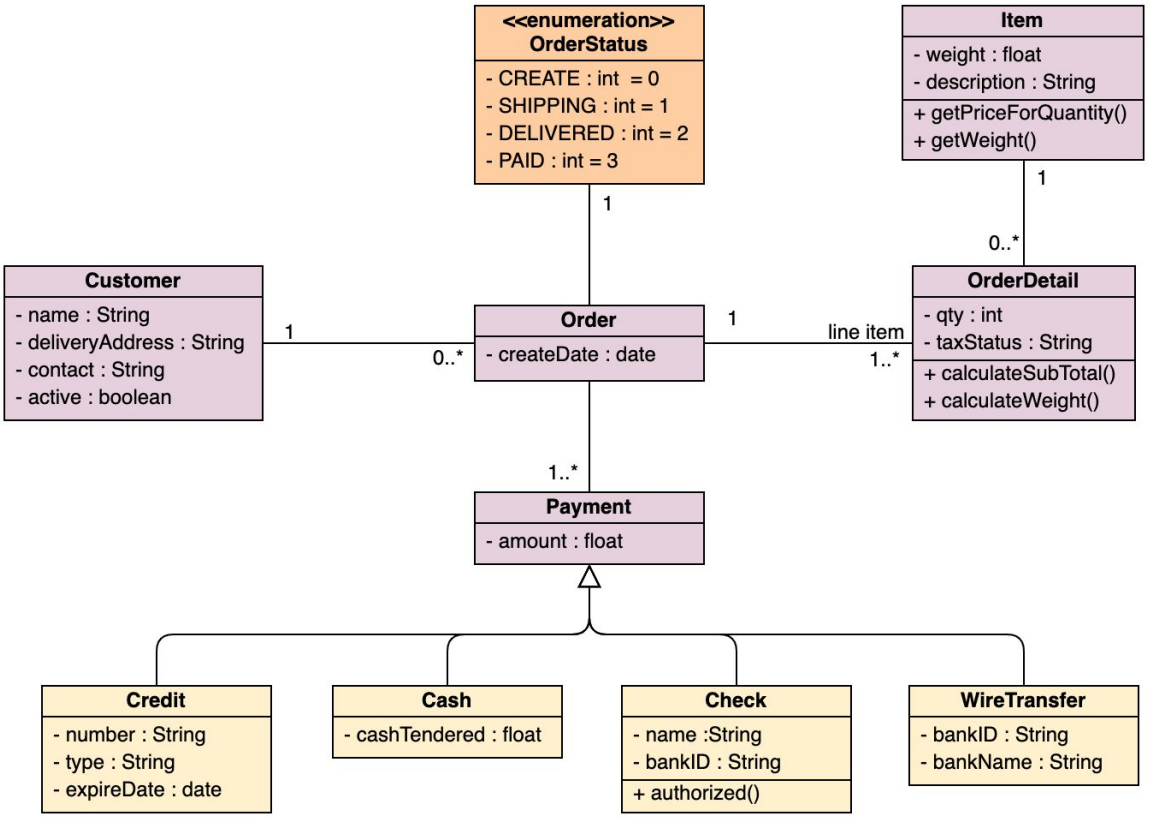
4. Consider the following class.

```
class Parent {

     private void m1()
```

```
     {
     System.out.println("From parent m1()");
     }
     protected void m2()
     {
          System.out.println("From parent m2()");
     }
}

class Child extends Parent {

     private void m1()
     {
          System.out.println("From child m1()");
     }

     @Override
     public void m2()
     {
          System.out.println("From child m2()");
     }
}

class Main {
     public static void main(String[] args)
     {
          Parent obj1 = new Parent();
          obj1.m2();
          Parent obj2 = new Child();
          obj2.m2();
     }
}
```

B. Coding in Java

For the following UML class diagram for a Sales Ordering System, write the source code in Java following Object Oriented Programming concepts.



C. Solving Problems: the Object-Oriented way

Read the given passage and design a solution to the said problem using Object Oriented Programing Concepts.

1. Design a class named MyDate.

The class contains:
■ The data fields year, month, and day that represent a date. month is 0-based, i.e., 0 is for January.
■ A no-arg constructor that creates a MyDate object for the current date.
■ A constructor that constructs a MyDate object with a specified elapsed time since midnight, January 1, 1970, in milliseconds.
■ A constructor that constructs a MyDate object with the specified year, month, and day.
■ Three getter methods for the data fields year, month, and day, respectively.
■ A method named setDate(long elapsedTime) that sets a new date for the object using the elapsed time.

Draw the UML diagram for the class and then implement the class. Write a test program that creates two MyDate objects (using new MyDate() and new MyDate(34355555133101L)) and displays their year, month, and day.

Hint: The first two constructors will extract the year, month, and day from the elapsed time. For example, if the elapsed time is 561555550000 milliseconds, the year is 1987, the month is 9, and the day is 18.
Java API has the GregorianCalendar class in the java.util package, which you can use to obtain the year, month, and day of a date. The no-arg constructor constructs an instance for the current date, and the methods get(GregorianCalendar.YEAR), get(GregorianCalendar.MONTH), and get(GregorianCalendar.DAY_OF_MONTH) return the year, month, and day. The GregorianCalendar class has setTimeInMillis(long), which can be used to set a specified elapsed time since January 1, 1970.

2. Design a class named Person and its two subclasses named Student and Employee. Make Faculty and Staff subclasses of Employee. A person has a name, address, phone number, and email address. A student has a class status (freshman, sophomore, junior, or senior). Define the status as a constant. An employee has an office, salary, and date hired. Use the MyDate class developed in the previous question to create an object for date hired. A faculty member has office hours and a rank. A staff member has a title. Override the toString method in each class to display the class name and the person's name.
Write a test program that creates a Person, Student, Employee, Faculty, and Staff, and invokes their toString() methods.

* * * * * * * *