# INTE2512 Object-Oriented Programming

## Lab – OOP 2

1. What is the printout of running the class C in (a)? What problem arises in compiling the program in (b) and why?

```
//(a)
class A {
    public A() {
        System.out.println("A's no-arg constructor is invoked");
    }
}
class B extends A {
}
public class C {
    public static void main(String[] args) {
        B b = new B();
    }
}
```

```
//(b)
class A {
    public A(int x) {
    }
}
class B extends A {
    public B() {
    }
}
public class C {
    public static void main(String[] args) {
        B b = new B();
    }
}
```

2. Identify an fix the errors in the following code.

```
public class Circle {
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    public double getRadius() {
        return radius;
    }
    public double getArea() {
        return radius * radius * Math.PI;
    }
}
class B extends Circle {
    private double length;
    B(double radius, double length) {
        Circle(radius);
        length = length;
    }
    public double getArea() {
        return getArea() * length;
    }
}
```

3. Show the output of the following code without running.

```
// (a)
public class Test {
    public static void main(String[] args) {
        new Person().printPerson();
        new Student().printPerson();
    }
}
class Student extends Person {
    @Override
    public String getInfo() {
        return "Student";
    }
}
class Person {
    public String getInfo() {
        return "Person";
    }
    public void printPerson() {
        System.out.println(getInfo());
    }
}
```

```
// (b)
public class Test {
    public static void main(String[] args) {
        new Person().printPerson();
        new Student().printPerson();
    }
}
class Student extends Person {
    String getInfo() {
        return "Student";
    }
}
class Person {
    String getInfo() {
        return "Person";
    }
    public void printPerson() {
        System.out.println(getInfo());
    }
}
```

4. 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. Find a suitable class from the Java libraries for the data type of 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.

   Draw the UML diagram for the classes and implement them. Write a test program that creates a Person, Student, Employee, Faculty, and Staff, and invokes their toString() methods.

5. Rewrite the following **MyCalendar** class to display a calendar for a specified month using the **Calendar** and **GregorianCalendar** classes from the **java.util** package. Sample run:

```
Enter full year (e.g., 2012): 2012 Enter

Enter month as a number between 1 and 12: 3 Enter

         March 2012
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
                   1   2   3
   4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29  30  31
```

You can input 0 for the year or month to select current year or current month, respectively.

```
import java.util.Scanner;
public class MyCalendar {
    public static void main(String[] args) {
        int year;            // the calendar is displayed for this year
        int month;           // and month
        // Create an instance of a standard input stream
        Scanner input = new Scanner(System.in);
        // The user enters year
        System.out.println("Enter full year (i.e. 1999 or 2000)");
        year = input.nextInt();
        // The user enters the month
        System.out.println("Enter month as an integer between 1 and 12");
        month = input.nextInt();
        // Print calendar for the month of the year
        printMonth(year, month);
    }

    // This method prints the calendar for the month of the year.
    public static void printMonth(int year, int month) {
        // Get start day of the week for the first date in the month
        int startDay = getStartDay(year, month);
        // Get number of days in the month
        int numOfDaysInMonth = getNumOfDaysInMonth(year, month);
        // Print headings for the calendar
        printMonthTitle(year, month);
        // Print body of the calendar
        printMonthBody(startDay, numOfDaysInMonth);
    }

    // This method determines the day (Sunday, Monday, etc. on which
    // the first of the month starts.
    public static int getStartDay(int year, int month) {
        int startDay1800 = 3;       //January 1, 1800 was on Wednesday
        // Get total number of days since 1/1/1800
        long totalNumOfDays = getTotalNumOfDays(year, month);

        // Return the start day
        return (int)((totalNumOfDays + startDay1800) % 7);
    }

    // This method determines the number of days that have
    // elapsed since 1/1/1800.
    public static long getTotalNumOfDays(int year, int month) {
        long total = 0;       //contains the total # of days since 1/1/1800
```

```
        //get the total days from 1800 to year -1
        for (int i = 1800; i < year; i++)
            if (leapYear(i))
                total = total + 366;
            else
                total = total + 365;
        // Add days from Jan to the month prior to the calendar month
        for (int i = 1; i < month; i++)
            total = total + getNumOfDaysInMonth(year, i);
        return total;
    }

    // This method determines the number of days in the selected month.
    // The year is needed to determine the number of days in February.
    public static int getNumOfDaysInMonth(int year, int month) {
        if (month == 1 || month==3 || month == 5 || month == 7 ||
                month == 8 || month == 10 || month == 12)
            return 31;
        if (month == 4 || month == 6 || month == 9 || month == 11)
            return 30;
        if (month == 2)
            if (leapYear(year))
                return 29;
            else
                return 28;
        return 0; // if month is incorrect.
    }

    // This method determines whether the selected year is a leap year.
    public static boolean leapYear(int year) {
        if ((year % 400 == 0) || ((year % 4 == 0) && (year % 100 != 0)))
            return true;
        return false;
    }

    // This method prints the body of the calendar for the given month given
    // day of the week on which the month starts and number of days in the month.
    public static void printMonthBody(int startDay, int numOfDaysInMonth) {
        // Print padding space before the first day of the month
        for (int i = 0; i < startDay; i++) {
            System.out.print("    ");
        }
        for (int i = 1; i <= numOfDaysInMonth; i++) {
            if (i < 10)
                System.out.print("   " + i);
            else
                System.out.print("  " + i);
            if ((i + startDay) % 7 == 0)
                System.out.println();
        }
        System.out.println();
    }

    // This method prints the title for the calendar for
    // the particular month of the year.
    public static void printMonthTitle(int year, int month) {
        System.out.println("         " + getMonthName(month) + ", " + year);
        System.out.println("-----------------------------");
        System.out.println(" Sun Mon Tue Wed Thu Fri Sat");
    }

    // This method is given an integer representation of the month.
    // It determines and returns a string representation of the month.
    public static String getMonthName(int month) {
        String monthName = null;      //string to contain the month's name
        switch (month)  {
            case 1: monthName = "January"; break;
            case 2: monthName = "February"; break;
```

```
            case 3: monthName = "March"; break;
            case 4: monthName = "April"; break;
            case 5: monthName = "May"; break;
            case 6: monthName = "June"; break;
            case 7: monthName = "July"; break;
            case 8: monthName = "August"; break;
            case 9: monthName = "September"; break;
            case 10: monthName = "October"; break;
            case 11: monthName = "November"; break;
            case 12: monthName = "December";
        }
        return monthName;
    }
}
```