

OBJECT ORIENTED PROGRAMMING

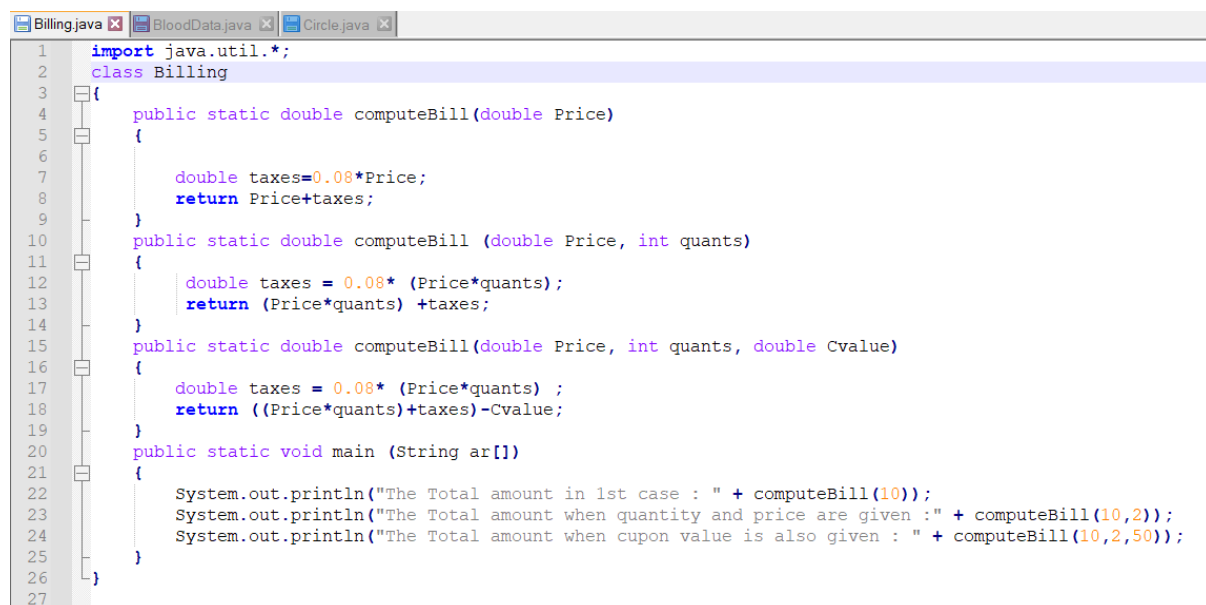
LAB2

1) Create a class named **Billing** that includes three overloaded **computeBill()** methods for a photo book store.

- When **computeBill()** receives a single parameter, it represents the price of one photo book ordered. Add 8% tax, and return the total due.
- When **computeBill()** receives two parameters, they represent the price of a photo book and the quantity ordered. Multiply the two values, add 8% tax, and return the total due.
- When **computeBill()** receives three parameters, they represent the price of a photo book, the quantity ordered, and a coupon value. Multiply the quantity and price, reduce the result by the coupon value, and then add 8% tax and return the total due.

Write a **main()** method that tests all three overloaded methods. Save the application as **Billing.java**.

CODE:



```
1  import java.util.*;
2  class Billing
3  {
4      public static double computeBill(double Price)
5      {
6
7          double taxes=0.08*Price;
8          return Price+taxes;
9      }
10     public static double computeBill (double Price, int quants)
11     {
12         double taxes = 0.08* (Price*quants);
13         return (Price*quants) +taxes;
14     }
15     public static double computeBill(double Price, int quants, double Cvalue)
16     {
17         double taxes = 0.08* (Price*quants) ;
18         return ((Price*quants)+taxes)-Cvalue;
19     }
20     public static void main (String ar[])
21     {
22         System.out.println("The Total amount in 1st case : " + computeBill(10));
23         System.out.println("The Total amount when quantity and price are given : " + computeBill(10,2));
24         System.out.println("The Total amount when cupon value is also given : " + computeBill(10,2,50));
25     }
26 }
27
```

Output:

```
The Total amount in 1st case : 10.8
The Total amount when quantity and price are given : 21.6
The Total amount when cupon value is also given : -28.4
```

2.

a.

Create a class named BloodData that includes fields that hold a blood type (the four blood types are O, A, B, and AB) and an Rh factor (the factors are + and -). Create a default constructor that sets the fields to O and +, and an overloaded constructor that requires values for both fields. Include get and set methods for each field. Save this file as BloodData.java. Create an application named TestBloodData that demonstrates each method works correctly. Save the application as TestBloodData.java.

code:

```
import java.util.*;

class BloodData
{
    char blood,rh;

    BloodData()
    {
        blood='O';
        rh='+';
    }

    BloodData(char blood,char rh)
    {
        this.blood=blood;
        this.rh=rh;
    }

    void setblood(char a)
    {
        blood=a;
    }
}
```

```
void setrh(char a)
{
    rh=a;
}

char getblood()
{
    return blood;
}

char getrh()
{
    return rh;
}
}

public class TestBloodData
{
    public static void main(String args[])
    {
        BloodData ob1=new BloodData();
        BloodData ob2=new BloodData('A','-');
        BloodData ob3=new BloodData();
        ob3.setblood('B');
        ob3.setrh('+');
        System.out.println("Blood for Default Constroctor "+ob1.blood);
        System.out.println("rh for Default Constroctor "+ob1.rh);
        System.out.println("Blood for Parametarized Constroctor "+ob2.blood);
        System.out.println("Rh for Parametarized Constroctor "+ob2.rh);
        System.out.println("Blood for Set functionr "+ob3.getblood());
        System.out.println("Rh for Set functionr "+ob3.getrh());
    }
}
```

```
}
```

Output:

```
Blood for Default Constroctor 0  
rh for Default Constroctor +  
Blood for Parametarized Constroctor A  
Rh for Parametarized Constroctor -  
Blood for Set functionr B  
Rh for Set functionr +
```

2b.

Create a class named Patient that includes an ID number, age, and BloodData. Provide a default constructor that sets the ID number to 0, the age to 0, and the BloodData values to O and +. Create an overloaded constructor that provides values for each field. Also provide get methods for each field. Save the file as Patient.java. Create an application that demonstrates that each method works correctly and save it as TestPatient.java.

code:

```
import java.util.*;
```

```
class BloodData
```

```
{
```

```
    char blood,rh;
```

```
    BloodData()
```

```
    {
```

```
        blood='O';
```

```
        rh='+';
```

```
    }
```

```
    BloodData(char blood,char rh)
```

```
{
    this.blood=blood;
    this.rh=rh;
}
void setblood(char a)
{
    blood=a;
}

void setrh(char a)
{
    rh=a;
}

char getblood()
{
    return blood;
}

char getrh()
{
    return rh;
}
}
```

```
class Patient
{
    BloodData ob=new BloodData();
    int id,age;
    Patient()
    {
        id=0;
        age=0;
        ob.blood='O';
        ob.rh='+';

    }

    Patient(int id,int age,BloodData ob)
    {
        this.id=id;
        this.age=age;
        this.ob=ob;
    }

    int getid()
    {
```

```
        return id;
    }

    int getage()
    {
        return age;
    }

    BloodData getBloodData()
    {
        return ob;
    }

}

public class TestPatient
{
    public static void main(String args[])
    {
        Patient ob1=new Patient();
        BloodData obBD=new BloodData('A','-');
        Patient ob2=new Patient(01425,62,obBD);
        System.out.println("ID for Default Constroctor "+ob1.getid());
        System.out.println("Age for Default Constroctor "+ob1.getage());
    }
}
```

```
        System.out.println("Blood Data for Default Constructor  
"+ob1.ob.blood);  
  
        System.out.println("Rh for Default Constructor "+ob1.ob.rh);  
  
        System.out.println();  
  
        System.out.println("ID for Parametarized Constructor  
"+ob2.getid());  
  
        System.out.println("Age for Parametarized Constructor  
"+ob2.getage());  
  
        System.out.println("Blood Data for Parametarized Constructor  
"+ob2.ob.blood);  
  
        System.out.println("Rh for Parametarized Constructor  
"+ob2.ob.rh);  
  
    }  
}
```

Output:

```
ID for Default Constructor 0  
Age for Default Constructor 0  
Blood Data for Default Constructor 0  
Rh for Default Constructor +  
  
ID for Parametarized Constructor 789  
Age for Parametarized Constructor 62  
Blood Data for Parametarized Constructor A  
Rh for Parametarized Constructor -
```

Question 3a.

Create a class named Circle with fields named radius, diameter, and area.

Include a constructor that sets the radius to 1 and calculates the other two

values. Also include methods named setRadius() and getRadius(). The

setRadius() method not only sets the radius, but it also calculates the other

two values. (The diameter of a circle is twice the radius, and the area of a circle

is pi multiplied by the square of the radius. Use the Math class PI constant for this calculation.) Save the class as Circle.java.

3.b.

Create a class named TestCircle whose main() method declares several Circle objects. Using the setRadius() method, assign one Circle a small radius value, and assign another a larger radius value. Do not assign a value to the radius of the third circle; instead, retain the value assigned at construction. Display all the values for all the Circle objects. Save the application as TestCircle.java.

code:

```
class Circle{  
  
int radius;  
  
int diameter;  
  
double area;  
  
public Circle() {  
radius = 1;  
diameter = 2 * radius;  
area = 3.141 * radius * radius;  
}  
  
public void setRadius(int radius){  
this.radius = radius;  
this.diameter = 2 * radius;  
this.area = 3.141* radius * radius;  
//compute area  
}  
  
public int getRadius (){  
return this.radius;  
}
```

```
public int getDiameter(){
return this.diameter;
}
public double getArea(){
return this.area;
}
}
//class testcircle
public class TestCircle{
public static void main(String Args[]) {
Circle c1 = new Circle();
c1.setRadius(23); //
System.out.println("radius is " + c1.getRadius() + " diameter is " +
c1.getDiameter() + " Area is " + c1.getArea());
}
}
```

Output:

```
radius is 23 diameter is 46 Area is 1661.589
```

```
** Process excited -Return code:0 **
```

4.

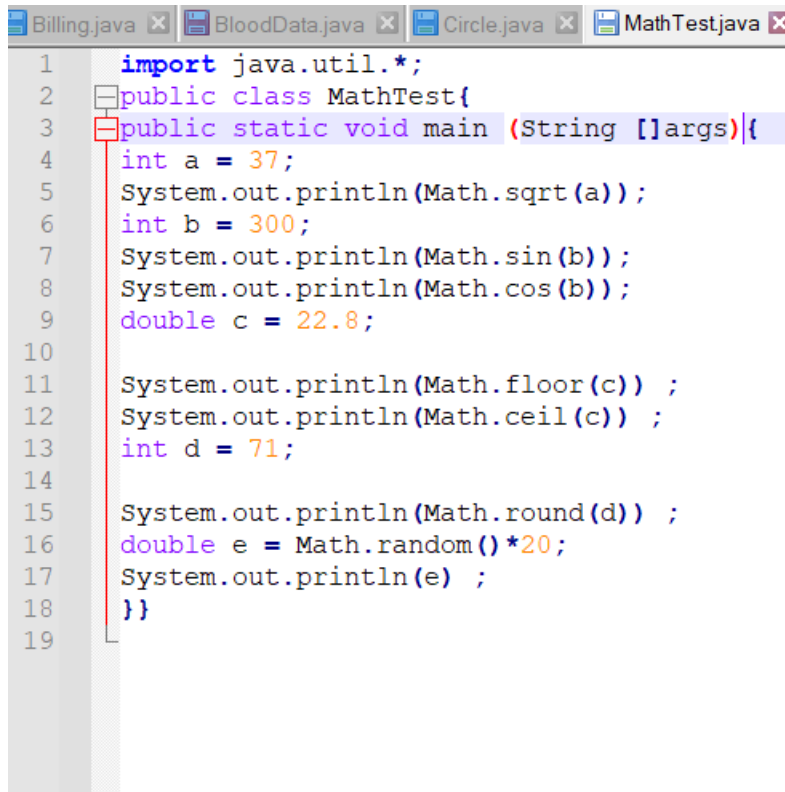
Write a Java application that uses the Math class to determine the answers for each of the following:(Use java.lang.Math class)

- a. The square root of 37
- b. The sine and cosine of 300
- c. The value of the floor, ceiling, and round of 22.8
- d. The larger and the smaller of the character 'D' and the integer 71
- e. A random number between 0 and 20 (Hint: The random() method returns a

value between 0 and 1; you want a number that is 20 times larger.)

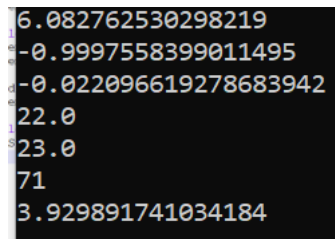
Save the application as MathTest.java.

Code:



```
1 import java.util.*;
2 public class MathTest{
3     public static void main (String []args){
4         int a = 37;
5         System.out.println(Math.sqrt(a));
6         int b = 300;
7         System.out.println(Math.sin(b));
8         System.out.println(Math.cos(b));
9         double c = 22.8;
10
11         System.out.println(Math.floor(c)) ;
12         System.out.println(Math.ceil(c)) ;
13         int d = 71;
14
15         System.out.println(Math.round(d)) ;
16         double e = Math.random()*20;
17         System.out.println(e) ;
18     }
19 }
```

Output:



```
6.082762530298219
-0.9997558399011495
-0.022096619278683942
22.0
23.0
71
3.929891741034184
```

5.a.

Write a program that declares two LocalDate objects and assign values that represent January 31 and December 31 in the current year. Display output that demonstrates the dates displayed when one, two, and three months are added to each of the objects. Save the application as TestMonthHandling.java.

code:

```

1  import java.time.*;
2  public class TestMonthHandling {
3      public static void main (String[] args)
4      {
5
6          LocalDate date = LocalDate.parse ("2021-1-31");
7          System.out.println("LocalDate before" + " adding months: " + date) ;
8          LocalDate returnvalue1 = date.plusMonths (1) ;
9          LocalDate returnvalue2 = date.plusMonths (2) ;
10         LocalDate returnvalue3 = date.plusMonths (3) ;
11
12         System.out.println("LocalDate after" + "adding months: " +returnvalue1);
13         System.out.println("LocalDate after" + "adding months: " +returnvalue2);
14         System.out.println("LocalDate after" + "adding months: " +returnvalue3);
15     }
16 }
17

```

Output:

```

LocalDate before adding months: 2018-12-31
LocalDate after  adding months: 2019-01-31
LocalDate after  adding months: 2019-02-28
LocalDate after  adding months: 2019-03-31

```

5.b.

Write an application that computes and displays the day on which you become (or became) 10,000 days old. Save the application as TenThousandDaysOld.java.

```

1  import java.time.*;
2  import java.util.Scanner;
3  public class TenThousandDaysOld{
4      public static void main (String [] args){
5          Scanner in = new Scanner (System.in) ;
6          int month;
7          int day;
8          int year ;
9          int daysOld = 10000;
10         System.out.println("Enter month");
11         month = in.nextInt () ;
12         System.out.println("Enter day");
13         day = in.nextInt();
14         System.out.println("Enter year");
15         year = in.nextInt();
16         LocalDate birthDate = LocalDate.of(year, month, day) ;
17         LocalDate futureDate = birthDate.plusDays (daysOld) ;
18         System.out.println("I will be " + daysOld + " old on " + futureDate);
19     }
20 }

```

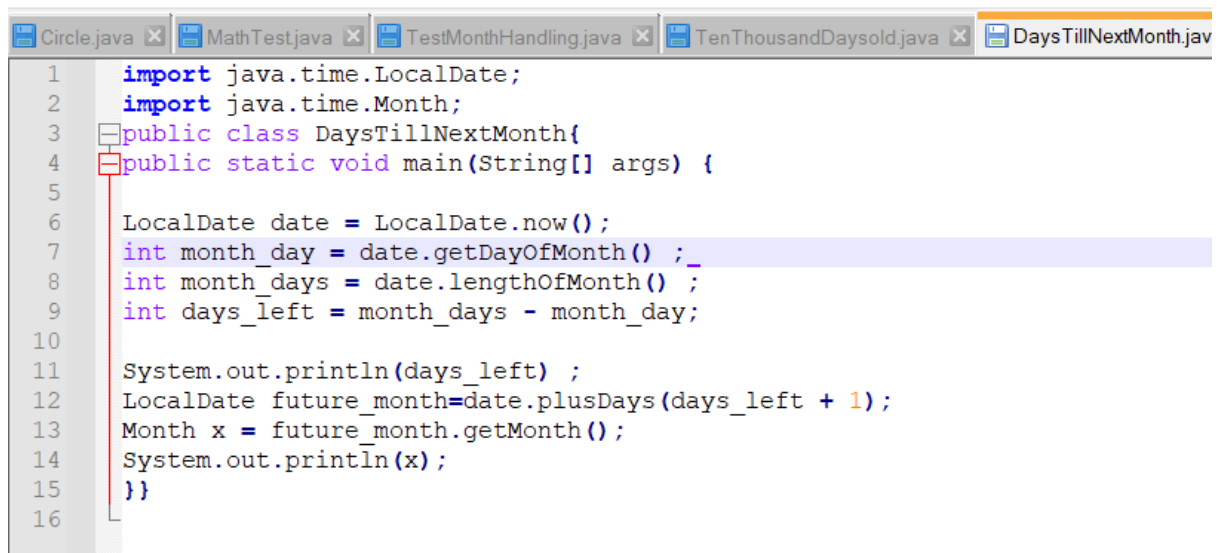
Output:

```
Enter month
03
Enter day
05
Enter year
2002
I will be 10000 old on 2029-09-18
```

5.c.

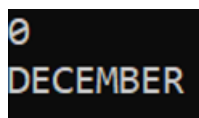
The `LocalDate` class includes an instance method named `lengthOfMonth()` that returns the number of days in the month. Write an application that uses methods in the `LocalDate` class to calculate how many days are left until the first day of next month. Display the result, including the name of the next month. Save the file as `DaysTillNextMonth.java`.

code



```
1  import java.time.LocalDate;
2  import java.time.Month;
3  public class DaysTillNextMonth{
4  public static void main(String[] args) {
5
6      LocalDate date = LocalDate.now();
7      int month_day = date.getDayOfMonth();
8      int month_days = date.lengthOfMonth();
9      int days_left = month_days - month_day;
10
11     System.out.println(days_left);
12     LocalDate future_month = date.plusDays(days_left + 1);
13     Month x = future_month.getMonth();
14     System.out.println(x);
15 }
16 }
```

Output:



```
0
DECEMBER
```