

Programs Using Static Final and constructors

LAB - 2

Object Oriented Programming (CSE2005)



Tharshith Goud

19BCN7112

SLOT - L6

Faculty: Dr Aravapalli Rama Satish



Question 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.

Solution:



VIT-AP

```
class Billing{
    public static double computeBill(double p){
        return p*(108/100);
    }
    public static double computeBill(double p,double p1){
        return p1*(p*(108/100));
    }
    public static double computeBill(double p,double p1,double p2){
        return (p1*(p*(108/100)))-p2;
    }
}


class BillingMain{
    public static void main(String[] ar){
        double a = Billing.computeBill(5);
        System.out.println("Cost of book is "+a);
        double b = Billing.computeBill(7,2);
        System.out.println("Cost of books is "+b);
        double c = Billing.computeBill(6,3,5);
        System.out.println("Cost of book after coupon is "+c);
    }
}
```

Output:

```
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> javac .\Billing.java
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> java BillingMain
Cost of book is 5.0
Cost of books is 14.0
Cost of book after coupon is 13.0
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> █
```

Question 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.

Program 1:



```
class BloodData{
    String type;
    String rh;
    BloodData(){
        this.type = "O";
        this.rh = "+";
    }
    BloodData(String type, String rh){
        this.type = type;
        this.rh = rh;
    }
    void settype(String type){
        this.type = type;
    }
    void setrh(String rh){
        this.rh = rh;
    }
    String gettype(){
        return type;
    }
    String getrh(){
        return rh;
    }
}
```

Program 2:

```
class TestBloodData{
    public static void main(String[] ar){
        BloodData ob1 = new BloodData();
        System.out.println("Default Blood type is "+ob1.gettype()+ob1.getrh());
    ;
        BloodData ob2 = new BloodData("A","-");
        System.out.println("Blood type using default constructor is "+ob2.gett
ype()+ob2.getrh());
        ob2.settype("AB");
        ob2.setrh("-");
        System.out.println("Blood type using getters and setters is "+ob2.gett
ype()+ob2.getrh());
    }
}
```

Output:



VIT-AP

```
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> javac .\BloodData.java
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> javac .\TestBloodData.java
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> java TestBloodData
Default Blood type is O+
Blood type using default constructor is A-
Blood type using getters and setters is AB-
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> █
```

Question 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.

Program 1:

```
class Patient{
    int ID;
    int age;
    BloodData b;
    Patient(){
        this.ID = 0;
        this.age = 0;
        this.b = new BloodData();
    }
    Patient(int id, int age, String type, String rh){
        this.ID = id;
        this.age = age;
        b = new BloodData();
        this.b.type = type;
        this.b.rh = rh;
    }
    public int getID() {
        return ID;
    }
    public void setID(int iD) {
        ID = iD;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public BloodData getB() {
        return b;
    }
    public void setB(String type, String rh) {
        this.b.type = type;
        this.b.rh = rh;
    }
}
```

Program 2:

```
class TestPatient{
    public static void main(String[] args){
        String newline = System.getProperty("line.separator");
        Patient a = new Patient();
        System.out.println("The values of Default constructor of Patient class
are"+ newline+" 1. Patient age is "+a.getAge()+newline+" 2. Patient ID is "+
a.getID()+newline+" 3. Patient Blood Group is "+a.getB().gettype()+a.getB().ge
trh());
        Patient b = new Patient(1792,18,"A","-");
        System.out.println("The values of OverLoadead constructor of Patient c
lass are"+ newline+" 1. Patient age is "+b.getAge()+newline+" 2. Patient ID i
s "+b.getID()+newline+" 3. Patient Blood Group is "+b.getB().gettype()+b.getB(
).getrh());
        Patient c = new Patient();
        c.setID(278);
        c.setAge(42);
        c.setB("B", "+");
        System.out.println("The values Using Setters and Getters of Patient cl
ass are"+ newline+" 1. Patient age is "+c.getAge()+newline+" 2. Patient ID is
"+c.getID()+newline+" 3. Patient Blood Group is "+c.getB().gettype()+c.getB()
.getrh());
    }
}
```



UNIVERSITY

Output:

```
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> javac .\Patient.java
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> javac .\TestPatient.java
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> java TestPatient
The values of Default constructor of Patient class are
1. Patient age is 0
2. Patient ID is 0
3. Patient Blood Group is O+
The values of OverLoadead constructor of Patient class are
1. Patient age is 18
2. Patient ID is 1792
3. Patient Blood Group is A-
The values Using Setters and Getters of Patient class are
1. Patient age is 42
2. Patient ID is 278
3. Patient Blood Group is B+
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> █
```

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.

Solution:

```
import java.lang.Math.*;

class Circle{
    double r;
    double diameter;
    double area;
    Circle(){
        this.r = 1;
        diameter = 2*this.r;
        area = Math.PI*this.r*this.r;
    }
    public void setRadius(double r){
        this.r = r;
        diameter = 2*this.r;
        area = Math.PI*this.r*this.r;
    }
    public double getRadius(){
        return this.r;
    }
}
```

Output:

In the output of next Question

Question 3B: 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.

Solution:

```
class TestCircle{
    public static void main(String[] args) {
        String newline = System.getProperty("line.separator");
        Circle a = new Circle();
        Circle b = new Circle();
        Circle c = new Circle();
        a.setRadius(5);
        b.setRadius(157523);
        System.out.println("The circle values with small radius is:"+newline+"
1. Radius is "+a.getRadius()+newline+"2. Diameter is "+a.diameter+newline+"3.
Area is "+a.area);
        System.out.println("The circle values with Big radius is:"+newline+"1.
Radius is "+b.getRadius()+newline+"2. Diameter is "+b.diameter+newline+"3. Ar
ea is "+b.area);
        System.out.println("The circle values with Default constructor is:"+ne
wline+"1. Radius is "+c.getRadius()+newline+"2. Diameter is "+c.diameter+newli
ne+"3. Area is "+c.area);
    }
}
```

Output:

```
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> javac .\Circle.java
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> javac .\TestCircle.java
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> java TestCircle
The circle values with small radius is:
1. Radius is 5.0
2. Diameter is 10.0
3. Area is 78.53981633974483
The circle values with Big radius is:
1. Radius is 157523.0
2. Diameter is 315046.0
3. Area is 7.795389526378958E10
The circle values with Default constructor is:
1. Radius is 1.0
2. Diameter is 2.0
3. Area is 3.141592653589793
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> █
```


Question 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.

Solution:

```
import java.lang.Math;

class MathTest{
    public static void main(String[] args) {
        System.out.println("1. The Square root of 37 is "+Math.sqrt(37));
        System.out.println("2. The Sine and Cosine of 300 is "+Math.sin(Math.toRadians(300))+" and "+Math.cos(Math.toRadians(300))+" Respectively");
        System.out.println("3. The value of 22.8 for floor is "+Math.floor(22.8)+" for ceilig is "+Math.ceil(22.8)+" and for round is "+Math.round(22.8));
        System.out.println("4. The Larger of 'D' and 71 is "+Math.max((int)'D',71)+" The smaller is "+Math.min((int)'D',71));
        System.out.println("5. A Random number Between 0 and 20 is "+Math.random()*20);
    }
}
```

Output:

```
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> javac .\MathTest.java
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> java MathTest
1. The Square root of 37 is 6.082762530298219
2. The Sine and Cosine of 300 is -0.8660254037844386 and 0.5000000000000001 Respectively
3. The value of 22.8 for floor is 22.0 for ceilig is 23.0 and for round is 23
4. The Larger of 'D' and 71 is 71 The smaller is 68
5. A Random number Between 0 and 20 is 5.127005415170213
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> █
```

Question 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.

Solution:

```
import java.time.*;

class TestMonthHandling{
    public static void main(String[] args) {
        LocalDate date1 = LocalDate.parse("2020-01-31");
        LocalDate date2 = LocalDate.parse("2020-12-31");
        System.out.println("Adding one month to jan-31-2020 is "+date1.plusMonths(1));
        System.out.println("Adding two months to jan-31-2020 is "+date1.plusMonths(2));
        System.out.println("Adding three months to jan-31-2020 is "+date1.plusMonths(3));
        System.out.println("Adding one month to dec-31-2020 is "+date2.plusMonths(1));
        System.out.println("Adding two months to dec-31-2020 is "+date2.plusMonths(2));
        System.out.println("Adding three months to dec-31-2020 is "+date2.plusMonths(3));
    }
}
```

Output:


```
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> javac .\TestMonthHandling.java
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> java TestMonthHandling
Adding one month to jan-31-2020 is 2020-02-29
Adding two months to jan-31-2020 is 2020-03-31
Adding three months to jan-31-2020 is 2020-04-30
Adding one month to dec-31-2020 is 2021-01-31
Adding two months to dec-31-2020 is 2021-02-28
Adding three months to dec-31-2020 is 2021-03-31
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> █
```

Question 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.

Solution:

```
import java.time.*;
import java.util.Scanner;
class TenThousandDaysOld{
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter The Date in the Format YYYY-MM-DD: ");
        String s = in.nextLine();
        LocalDate birthday = LocalDate.parse(s);
        System.out.println("The Date on which you will be 10,000 days old is "
+birthday.plusDays(10000));
    }
}
```

Output:



```
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> javac .\TenThousandDaysOld.java
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> java TenThousandDaysOld
Enter The Date in the Format YYYY-MM-DD:
2007-05-13
The Date on which you will be 10,000 days old is 2034-09-28
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> █
```

Question 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 `DaysTilNextMonth.java`.

Solution:

```
import java.time.*;
import java.util.Scanner;
class DaysTilNextMonth{
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter The Date in the Format YYYY-MM-DD: ");
        String s = in.nextLine();
        LocalDate date = LocalDate.parse(s);
        int days_left = date.lengthOfMonth()-date.getDayOfMonth();
        LocalDate future_month = date.plusDays(days_left + 1);
        Month x = future_month.getMonth();

        System.out.println("Days Left until next Month is "+days_left+" The Ne
xt Month Is "+x);
    }
}
```

Output:

```
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> javac .\DaysTilNextMonth.java
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> java DaysTilNextMonth
Enter The Date in the Format YYYY-MM-DD:
2020-11-24
Days Left until next Month is 6 The Next Month Is DECEMBER
PS C:\Users\Dracarys\Desktop\LABS\OOP\19BCN7112> █
```

GITHUB LINK:

<https://github.com/tharshith44/OOPLab>

THE END