OOP EXPERIMENT-2

NAME-ANMOL

SAP-590011794

BATCH-20

DATE-9 FEB 2026

SUBMITTED TO- PROF. Kalluri Shareef Babu

Theory covered in exe2:

Theory covered in exe2 is mainly based on constructors, objects, and the hashCode() method in Java. The exercises demonstrate how to create default and parameterized constructors, and how they are used to initialize instance variables at the time of object creation. They also show how multiple objects of a class can be created and how each object has its own unique identity, which is illustrated using the hashCode() method. The programs further explain how methods are used to display object data. Overall, the exe2 exercises provide a practical understanding of constructors, object creation, and object identity in Java within the context of object-oriented programming.

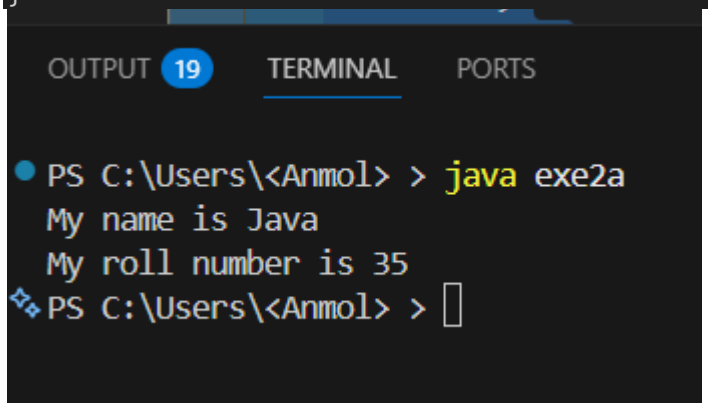
Topics covered in exe2 are:

- Constructors in Java
- Default constructors
- Parameterized constructors
- Constructor overloading
- Initializing instance variables using constructors
- Creating objects using new
- Methods for displaying object data
- hashCode() method in Java
- Object identity and uniqueness
- Role of constructors in OOP

Class Exercises

EX-2A

```java
// constructor
class student
{
    String name;
    int roll;
    void details()
    {
        System.out.println("My name is "+name);
        System.out.println("My roll number is "+roll);
    }
}
public class exe2a
{
    public static void main(String args[])
    {
        student s=new student();
        s.name="Java";
        s.roll=35;
        s.details();
    }
}
```



//observation: The class student has instance variables name and roll, which are initialized in the main method. The details method is called to print the values of name and roll to the console using System.out.println() method inside the details method.

EX-2B

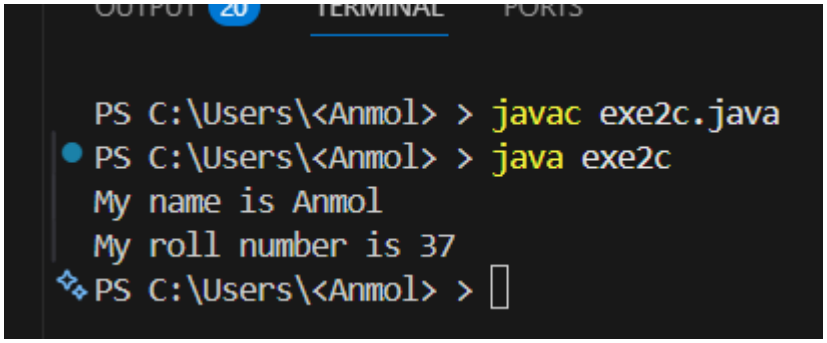
```java
class Student
{
    String name = "Java";
    int roll = 35;
    void details()
    {
        System.out.println("My name is " + name);
        System.out.println("My roll number is " + roll);
    }
}
public class exe2b
{
    public static void main(String args[])
    {
        Student s=new Student();
        s.details();
    }
}
```

```
PS C:\Users\<Anmol> > java exe2b
My name is null
My roll number is 0
PS C:\Users\<Anmol> >
```

observation: The class Student has instance variables name and roll, which are initialized with default values. The details method is called to print the values of name and roll to the console using System.out.println() method inside the details method. When the main method is executed, it creates an instance of the Student class and calls the details method, which displays the default values of name and roll on the console.

EXE-2C

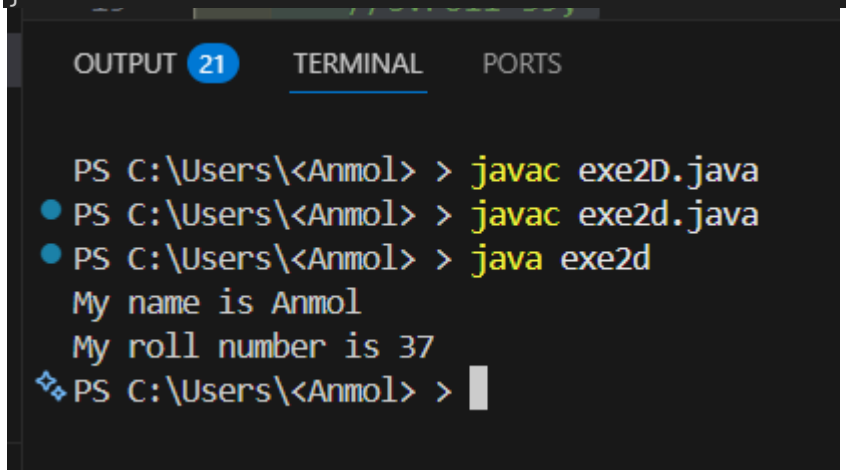
```java
//Creating a class Student with attributes name and roll number
class Student
{
    String name = "Java";
    int roll = 35;
    void details()
    {
        System.out.println("My name is " +name);
        System.out.println("My roll number is " +roll);
    }
}
public class exe2c

{
    public static void main(String args[])
    {
        Student s=new Student();
        s.name="Anmol";
        s.roll=37;
        s.details();
    }
}
```



observation: The class Student has instance variables name and roll, which are initialized with default values. In the main method, the values of name and roll are updated to "Anmol" and 37 respectively. The details method is called to print the updated values of name and roll to the console using System.out.println() method inside the details method. When the main method is executed, it creates an instance of the Student class, updates the values of name and roll, and calls the details method, which displays the updated values on the console.

EXE-2D

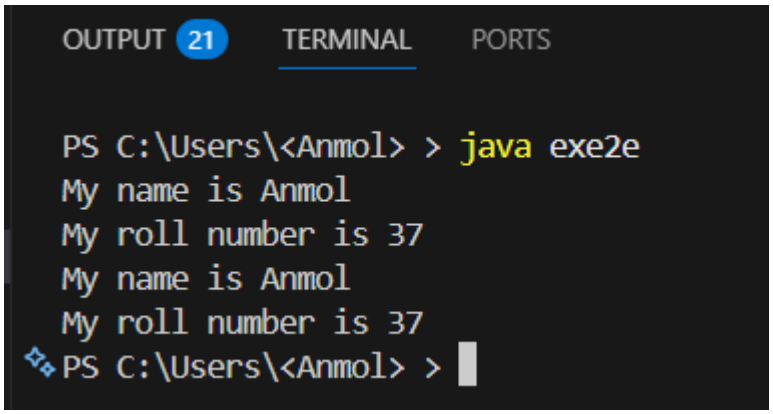
```java
class Student
{
    private String name = "Java";
    private int roll = 35;
    void details()
    {
        String name ="Anmol";
        int roll =37;
        System.out.println("My name is " +name);
        System.out.println("My roll number is " +roll);
    }
}
public class exe2d
{
    public static void main(String args[])
    {
        Student s=new Student();
        //s.name="Damn";
        //s.roll=39;
        s.details();
    }
}
```



observation: The variables name and roll are declared as private, so they cannot be accessed or modified directly from the main method. However, the details method can access both variables since it is within the same class. In this case, the details method has its own local variables name and roll, which are initialized with different values than the instance variables. When the details method is called, it prints the values of the local variables instead of the instance variables.

EXE-2E

```java
class Student
{
    private String name ="Java";
    private int roll =35;
    void details()
    {
        System.out.println("My name is " +name);
        System.out.println("My roll number is " +roll);
    }
}
public class exe2e
{
    public static void main(String args[])
    {
        Student s1=new Student();
        s1.details();
        Student s2=new Student();
        //s2.name="Anmol"; causes error
        //s2.roll=37; causes error
        s2.details();

    }
}
```



observation: The variables name and roll are declared as private, so they cannot be accessed or modified directly from the main method. However, the details method can access both variables since it is within the same class. When the details method is called for both s1 and s2 objects, it prints the values of the instance variables name and roll, which are initialized with default values "Java" and 35 respectively. Since we cannot modify the private variables directly, both s1 and s2 will display the same values when their details method is called.

EXE-2F

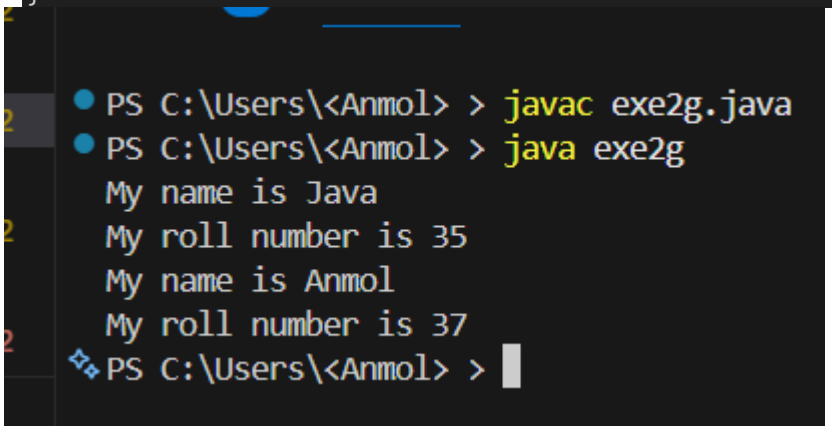
```
class Student
{
    private String name;
    private int roll ;
    Student()
    {
        name="Java";
        roll=35;
    }
    void details()
    {
        System.out.println("My name is " +name);
        System.out.println("My roll number is " +roll);
    }
}
public class exe2f
{
    public static void main(String args[])
    {
        Student s1=new Student();
        s1.details();
    }}
```

```
PS C:\Users\<Anmol> > java exe2f
My name is Anmol
My roll number is 37
PS C:\Users\<Anmol> >
```

observation: The class Student has a default constructor that initializes the instance variables name and roll with default values. The details method is called to print the values of name and roll to the console using System.out.println() method inside the details method. When the main method is executed, it creates an instance of the Student class, which triggers the execution of the default constructor and initializes the variables with default values. The details method is then called to display the values on the console.

EXE-2G

```java
class Student
{
    private String name;
    private int roll ;
    Student()
    {
        name="Java";
        roll=35;
    }
    Student(String s,int n)
    {
        name=s;
        roll=n;
    }
    void details()
    {
        System.out.println("My name is " +name);
        System.out.println("My roll number is " +roll);
    }
}
public class exe2g
{
    public static void main(String args[])
    {
        Student s1=new Student();
        s1.details();
        Student s2=new Student("Anmol",37);
        s2.details();
    }
}
```



observation: The class Student has two constructors, a default constructor that initializes the name and roll variables with default values, and a parameterized constructor that takes two parameters to initialize the variables with specific values. In the main method, two objects of the class are created

EXE-2H

```java
//print hashcode of two objects
class stu
{
    String name;
    int roll;
}
public class exe2h
{
    public static void main(String args[])
    {
        stu s1=new stu();
        stu s2=new stu();
        System.out.println("Hashcode of s1:"+s1.hashCode());
        System.out.println("Hashcode of s2:"+s2.hashCode());
    }
}
```

```
PS C:\Users\<Anmol> > java exe2h
Hashcode of s1:681842940
Hashcode of s2:1746572565
PS C:\Users\<Anmol> >
```

observation: The class stu has instance variables name and roll, but they are not initialized in the main method. When two objects of the class stu are created, they will have different hashcodes since they are different instances of the class. The hashcode is a unique identifier for each object and is generated based on the memory address of the object. Therefore, when the hashcode of s1 and s2 is printed to the console, it will show different values for each object.

Homework

Exe 2_1a

```java
//declare a variable int x inside main() and pass it to a method to print its
value
public class exe2_1a
{
    void printValue(int x)
    {
        System.out.println("The value of x is: "+x);
    }
    public static void main(String args[])
    {
        int x=10;
        exe2_1a obj=new exe2_1a();
        obj.printValue(x);
    }
}
```

```
PS C:\Users\<Anmol> > java exe2_1a
The value of x is: 10
PS C:\Users\<Anmol> >
```

observation: The variable x is declared inside the main method and is passed as an argument to the printValue method, which prints its value.

Exe2_1b

```java
//initialize two numbers in main() and pass them to another class to calc sum
public class exe2_1b
{
    void sum(int a,int b)
    {
        int s=a+b;
        System.out.println("The sum is: "+s);
    }
public static void main(String args[])
    {
        int a=5;
        int b=10;
        exe2_1b obj=new exe2_1b();
        obj.sum(a,b);
    }
}
```

```
PS C:\Users\<Anmol> > java exe2_1b
The sum is: 15
PS C:\Users\<Anmol> >
```

observation: The variables a and b are initialized in the main method and passed as arguments to the sum method, which calculates and prints their sum to the console using System.out.println() method.

Exe 2_2a

```
//create a class with instance variables name and age intialize them using a
method and display values
class Student
{
    String name;
    int age;
    void display()
    {
        System.out.println("Name:"+name);
        System.out.println("Age:"+age);
    }
}
public class exe2_2a {
    public static void main(String args[])
    {
        Student s = new Student();
        s.name="Anmol";
        s.age=20;
        s.display();
    }
}
```

```
PS C:\Users\<Anmol> > java exe2_2a
Name:Anmol
Age:20
PS C:\Users\<Anmol> >
```

observation: The class Student has instance variables name and age, which are initialized in the main method. The display method is called to print the values of name and age to the console using System.out.println() method inside the display method.

Exe2_2b

```
//initailize class variables using a constructor and display values
class exe2_2b
{
    String name;
    int age;
```

```
    exe2_2b(String n,int a)
    {
        name=n;
        age=a;
    }
public void display()
    {
        System.out.println("Name:"+name);
        System.out.println("Age:"+age);
    }
    public static void main(String args[])
    {
        exe2_2b s=new exe2_2b("Anmol", 20);
        s.display();
    }
}
```

```
PS C:\Users\<Anmol> > java exe2_2b
Name:Anmol
Age:20
PS C:\Users\<Anmol> >
```

observation: The class exe2_2b has instance variables name and age, which are initialized using a constructor. The display method is called to print the values of name and age to the console using System.out.println() method inside the display method.

Exe 2_3a

```
/*create a class with:
1) one public variable
2) one private variable
access them from another class and observe the result */
class Student
```

```
{
    public String name = "Java";
    private int roll = 35;
    void details()
    {
        System.out.println("My name is " +name);
        System.out.println("My roll number is " +roll);
    }
}
public class exe2_3a
{
    public static void main(String args[])
    {
        Student s=new Student();
        s.name="Anmol";
        //s.roll=37; //error
        s.details();
    }
}
```

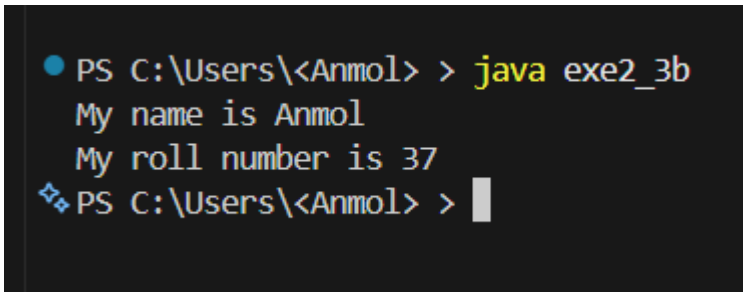
```
PS C:\Users\<Anmol> > java exe2_3a
My name is Anmol
My roll number is 35
PS C:\Users\<Anmol> >
```

observation: The variable name is declared as public, so it can be accessed and modified from the main method. However, the variable roll is declared as private, so it cannot be accessed or modified directly from the main method, resulting in a compilation error if we try to access it. The details method can access both variables since it is within the same class.

Exe 2_3b

```
//wap to showtaht private members cannot be accessed directly outside the class
class Student
{
    private String name = "Java";
    private int roll = 35;
```

```
    void details()
    {
        String name ="Anmol";
        int roll =37;
        System.out.println("My name is " +name);
        System.out.println("My roll number is " +roll);
    }
}
class exe2_3b
{
    public static void main(String args[])
    {
        Student s=new Student();
        s.details();
    }
}
```



observation: The variables name and roll are declared as private, so they cannot be accessed or modified directly from the main method. However, the details method can access both variables since it is within the same class. In this case, the details method has its own local variables name and roll, which are initialized with different values than the instance variables. When the details method is called, it prints the values of the local variables instead of the instance variables.

Exe 2_3c

```
// demonstrate default access by accessing variables within the same package
using inheritance
class Student
{
    String name="Java";
```

```
    int roll=35;

    void details()
    {
        System.out.println("Name: "+name);
        System.out.println("Roll: "+roll);
    }
}
public class exe2_3c extends Student
{
    public static void main(String args[])
    {
        exe2_3c s=new exe2_3c();
        s.name="Anmol";
        s.roll=37;
        s.details();
    }
}
```

```
PS C:\Users\<Anmol> > java exe2_3c
Name: Anmol
Roll: 37
PS C:\Users\<Anmol> >
```

observation: The variables name and roll are declared with default access, which means they can be accessed and modified from any class within the same package. In this case, the class exe2_3c extends the Student class, so it can access and modify the variables name and roll directly. The details method is called to print the updated values of name and roll to the console using System.out.println() method inside the details method

Exe2_4

```
//create a class with a default constructor that prints a message when an object
is created
class exe2_4
{
    exe2_4()
```

```
    {
        System.out.println("Object is created using default constructor");
    }
    public static void main(String args[])
    {
        exe2_4 obj=new exe2_4();
    }
}
```

```
PS C:\Users\<Anmol> > java exe2_4
Object is created using default constructor
PS C:\Users\<Anmol> >
```

observation: The class exe2_4 has a default constructor that is called when an object of the class is created. The constructor prints a message to the console using System.out.println() method, indicating that an object has been created. When the main method is executed, it creates an instance of the exe2_4 class, which triggers the execution of the default constructor and displays the message on the console.

Exe 2_5

```
//create a class with a parametarized constructor to initialize name and marks
class exe2_5
{
    String name;
    int marks;
```

```
    exe2_5(String n,int m)
    {
        name=n;
        marks=m;
    }
    public static void main(String args[])
    {
        exe2_5 s=new exe2_5("Anmol",85);
        System.out.println("Name:"+s.name);
        System.out.println("Marks:"+s.marks);
    }
}
```

```
PS C:\Users\<Anmol> > javac exe2_5.java
PS C:\Users\<Anmol> > java exe2_5
Name:Anmol
Marks:85
PS C:\Users\<Anmol> >
```

observation: The class exe2_5 has a parameterized constructor that takes two parameters, name and marks, to initialize the instance variables of the class. When an object of the class is created in the main method, the constructor is called with the arguments "Anmol" and 85, which initializes the name and marks variables. The values of name and marks are then printed to the console using System.out.println() method.

Exe2_6

```
/* write a program having:
1) one default constructor
2) one parameterized constructor
create objects using both constructors and display values of variables*/
class exe2_6
```

```
{
    String name;
    int roll;
    exe2_6()
    {
        name="Java";
        roll=35;
    }
    exe2_6(String n,int r)
    {
        name=n;
        roll=r;
    }
    void display()
    {
        System.out.println("Name:"+name);
        System.out.println("Roll:"+roll);
    }
    public static void main(String args[])
    {
        exe2_6 s1=new exe2_6();
        s1.display();
        exe2_6 s2=new exe2_6("Anmol",37);
        s2.display();
    }
}
```

```
PS C:\Users\<Anmol> > javac exe2_6.jav
PS C:\Users\<Anmol> > java exe2_6
Name:Java
Roll:35
Name:Anmol
Roll:37
PS C:\Users\<Anmol> >
```

observation: The class exe2_6 has two constructors, a default constructor that initializes the name and roll variables with default values, and a parameterized constructor that takes two parameters to initialize the variables with specific values. In the main method, two objects of the class are created