**Name:** Shaikh Salif Aminuddin

**CLASS:** S2 **ROLL NO:** 2201094

# EXPERIMENT NO: 3

## AIM: Programs on class and objects.

## THEORY:

OBJECT: An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible). The example of an intangible object is the banking system

An object has three characteristics:

- State: represents the data (value) of an object.
- Behavior: represents the behavior (functionality) of an object such as deposit, withdraw, etc.
- Identity: An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

For Example, Pen is an object. Its name is Reynolds; color is white, known as its state. It is used to write, so writing is its behavior.

An object is an instance of a class. A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class. Object Definitions:

- An object is a real-world entity.
- An object is a runtime entity.
- The object is an entity which has state and behavior.
- The object is an instance of a class.

3 Ways to initialize object There are 3 ways to initialize object in Java.

- By reference variable
- By method
- By constructor

CLASS: A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical. A class in Java can contain:

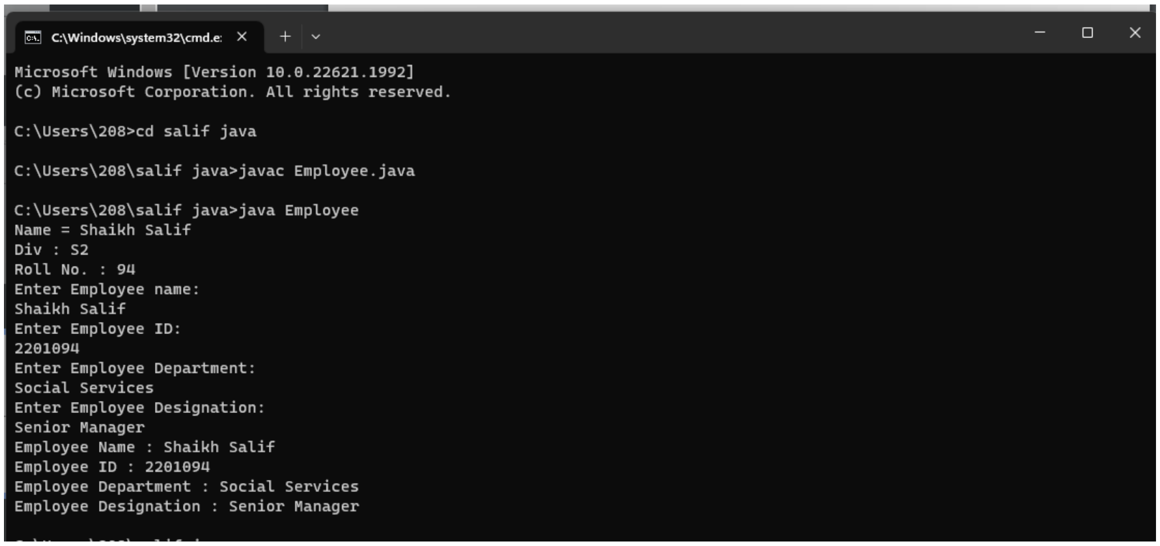
- Fields
- Methods
- Constructors
- Blocks
- Nested class and interface

**CODE-01:** WAP to read and display details of an employee using single class and its object.

```
import java.io.*;
import java.util.Scanner;
public class Employee
{
String name,id,desgn,dept;
void read()
{
System.out.println("Name = Shaikh Salif\nDiv : S2\nRoll No. : 94");
Scanner emp=new Scanner(System.in);
System.out.println("Enter Employee name:");
name=emp.nextLine();
System.out.println("Enter Employee ID:");
id=emp.nextLine();
```

```java
System.out.println("Enter Employee Department:");
dept=emp.nextLine();
System.out.println("Enter Employee Designation:");
desgn=emp.nextLine();
}
void display()
{
System.out.println("Employee Name : "+name);
System.out.println("Employee ID : "+id);
System.out.println("Employee Department : "+dept);
System.out.println("Employee Designation : "+desgn);
}
public static void main(String args[])
{
Employee sc=new Employee();
sc.read();
sc.display();
}
}
```

**OUTPUT:**

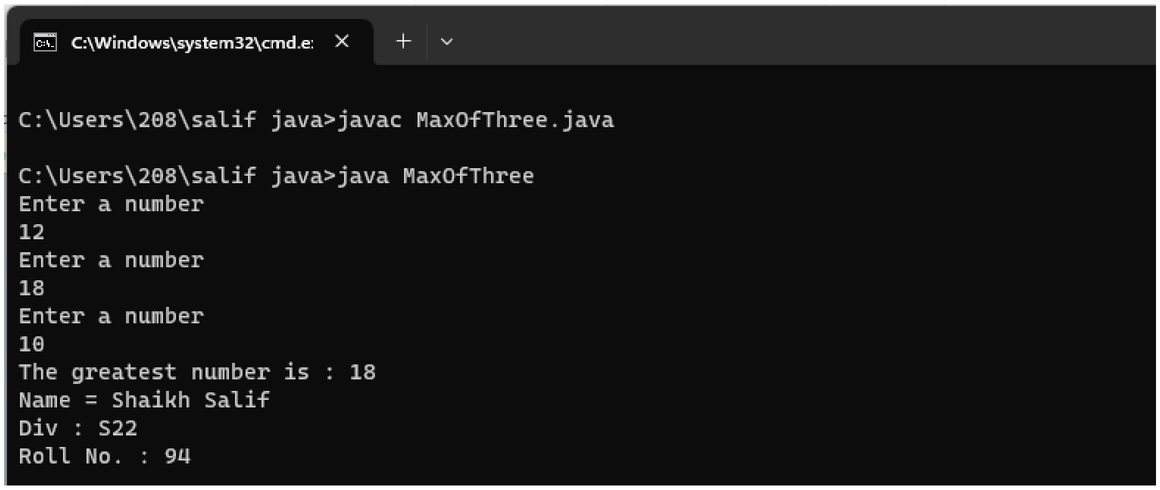


```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\208>cd salif java

C:\Users\208\salif java>javac Employee.java

C:\Users\208\salif java>java Employee
Name = Shaikh Salif
Div : S2
Roll No. : 94
Enter Employee name:
Shaikh Salif
Enter Employee ID:
2201094
Enter Employee Department:
Social Services
Enter Employee Designation:
Senior Manager
Employee Name : Shaikh Salif
Employee ID : 2201094
Employee Department : Social Services
Employee Designation : Senior Manager
```

**CODE-02:** WAP to find maximum of 3 numbers using conditional operator, using two classes and function resulting result.

```
import java.io.*;
import java.util.Scanner;
class MaxNum
{
int getdata()
{
Scanner sc=new Scanner(System.in);
System.out.println("Enter a number");
int n=sc.nextInt(); return n;
}
int Greatest(int n1, int n2, int n3)
{
int max=(n1>n2)?(n1>n3?n1:n3):(n2>n3?n2:n3);
return max;
} }
public class MaxOfThree
{
public static void main(String args[])
{
MaxNum obj=new MaxNum();
int n1=obj.getdata();
int n2=obj.getdata();
int n3=obj.getdata();
System.out.println("The greatest number is : "+obj.Greatest(n1,n2,n3));
System.out.println("Name = Shaikh Salif\nDiv : S22\nRoll No. : 94");
}
}
```

**OUTPUT:**



C:\Windows\system32\cmd.e

```
C:\Users\208\salif java>javac MaxOfThree.java

C:\Users\208\salif java>java MaxOfThree
Enter a number
12
Enter a number
18
Enter a number
10
The greatest number is : 18
Name = Shaikh Salif
Div : S22
Roll No. : 94
```

**CONCLUSION:** In conclusion, understanding classes and objects is fundamental in Java programming. Classes serve as blueprints, defining the structure and behavior of objects. Objects are instances of classes, encapsulating data and methods. By utilizing classes and objects effectively, developers can create modular, reusable, and organized code, enhancing code maintainability and promoting good software design practices.