

## Experiment 10: Abstract Class and Interface in Java

### 10(a). Abstract Class

#### **Aim**

To write a Java program to demonstrate the use of an **abstract class**.

#### **Theory:**

An abstract class is a class that cannot be instantiated. It can contain abstract methods (methods without a body) as well as concrete methods. Abstract classes are used when different classes share common behavior but implement it differently.

#### **Algorithm**

1. Create an abstract class
2. Declare an abstract method
3. Create a subclass that extends the abstract class
4. Implement the abstract method
5. Create an object of the subclass and call the method

## Program Code:

```
abstract class CricketMatch {

    String groundName;
    CricketMatch(String ground) {
        groundName = ground;
    }

    abstract void matchType();

    void showGround() {
        System.out.println("Match Ground: " + groundName);
    }

    void basicRules() {
        System.out.println("Each team has 11 players");
    }
}

class T20Match extends CricketMatch {
    T20Match(String ground) {
        super(ground);
    }
    void matchType() {
        System.out.println("This is a T20 Cricket Match");
    }
}

class AbstractClassDemo {
    public static void main(String[] args) {
        T20Match match = new T20Match("Narendra Modi Stadium");

        match.showGround();
        match.matchType();
        match.basicRules();
    }
}
```

## **Result**

The program successfully demonstrates an abstract class. The abstract method is implemented in the subclass and executed.

## 10(b). Interface

### **Aim**

To write a Java program to demonstrate the use of an interface.

### **Theory:**

An interface contains only method declarations. A class implements an interface and provides definitions for its methods. Interfaces help achieve standard behavior across different classes.

### **Algorithm**

1. Create an interface
2. Declare method inside the interface
3. Create a class that implements the interface
4. Define the method.
5. Call the method using object

## Program Code:

```
interface DigitalPayment {
    void startPayment();
    void verifyUser();
    void showPaymentMethod();
}

class GooglePay implements DigitalPayment {

    String userName;
    double amount;

    GooglePay(String userName, double amount) {
        this.userName = userName;
        this.amount = amount;
    }

    public void startPayment() {
        System.out.println("Starting payment for " + userName);
    }

    public void verifyUser() {
        System.out.println("User verified using mobile number and PIN");
    }

    public void showPaymentMethod() {
        System.out.println("Payment of ₹" + amount + " done using Google Pay");
    }
}

class InterfaceDemo {
    public static void main(String[] args) {
        GooglePay payment = new GooglePay("Rahul", 1500);

        payment.startPayment();
        payment.verifyUser();
        payment.showPaymentMethod();
    }
}
```

## **Result**

The program successfully demonstrates an interface. The class implements the interface and executes its method.