Lab 6: To illustrate the concept of Abstract class and Interface

Objectives:

1. To understand the concept of an abstract class.
2. Create and extend abstract classes and methods.
3. To be familiar with the interface.

Theory:

Abstract class:

- An abstract class is a class that is declared 'abstract' .
- It can, but does not have to, include abstract methods
- Abstract classes can not be instantiated but they can be subclassed using the extends keyword.
- An abstract method is a method that is declared without an implementation.
- Requires the abstract keyword

  If a class includes abstract methods, then the class itself must be declared as abstract.

```
public abstract class Person
{
 abstract void getData( );   //declaring method abstract
}
```

- When a child class extends an abstract class, it must either:
  - Provide implementations for all abstract methods from its parent class, or
  - Also be abstract
- Child classes can reference the constructor of an abstract class by using super().

Interface:

- Interface is a group of methods and field declarations with a name.
- It has static constants and abstract methods only.
- It cannot be instantiated.
- It represents the IS-A relationship.

Reason for using interface:

- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance

interface <interface_name>

```
interface Teacher
{
 void display();
 }
class Student implements Teacher
{
 public void display()
{
System.out.println(“Hello”);
}
 }
```

Program 1:

```
interface Bark {
   void bark();
}

class Dog implements Bark   // class Dog implements interface Bark

 {

   public void bark() {
      System.out.println("A dog barks");
   }
}

class Test {

   public static void main(String[] args) {
      Dog d = new Dog();
```

```
        d.bark();
    }
}
```

Program 2:
Create two subclasses ‘FirstSubclass’ and ‘SecondSubclass’ that both extend the ‘Parent’ class. Each subclass should use the ‘displayMessage’ method with the first subclass printing 'This is first subclass’ and the second subclass printing ‘This is the second subclass’. In the main method create an object of each subclass and call the ‘displayMessage’ method for each object.

Program 3:
Create an abstract class 'Bank' with an abstract method 'getBalance'. 'BankA', and 'BankB' are subclasses of class 'Bank', each having a method named 'getBalance' with a parameter of balance. Consider BankA contains implementation details and Rs 10,000 gets deposited in BankA. But BankB does not contain implementation. Extend class BankC from class BankB with the same method and deposit Rs20,000. Call this method with arguments by creating an object of the classes from the main class.

Program 4:
Create an abstract class 'Marks' with an abstract method 'getPercentage'. It is inherited by two other classes 'A' and 'B' each having a method with the same name which returns the percentage of the students. The constructor of student A takes the marks in three subjects as its parameters and the marks in four subjects as its parameters for student B.Calculate the percentage of marks obtained in three subjects (each out of 100) by student A and in four subjects (each out of 100) by student B. Create an object for each of the two classes and print the percentage of marks for both the students.

Program 5:
WAP to create an interface named Shape with its data members and methods for calculating volume. Also, derive a class cuboid which implements the Shape and calculate the volume of the cuboid. Also, derive a class cube from the class Shape and method for calculating the volume. Now, Using the main class InterfaceTest , access the methods and variables from it for displaying the volume of cube and cuboid.