# Abstraction

```
/*
 * 1. Create an abstract class 'Parent' with a method 'message'. It has two subclasses each having a method with the same
 * name 'message' that prints "This is first subclass" and "This is second subclass" respectively. Call the methods 'message'
 * by creating an object tier each subclass.
 */

public class callingParent{
    public static void main(String[] args){
        subClass1 obj1 = new subClass1();
        subClass2 obj2 = new subClass2();
        obj1.message();
        obj2.message();
    }
}

abstract class Parent{
    void message(){}
}

class subClass1 extends Parent{
    void message(){
        System.out.println("This is first subclass");
    }
}

class subClass2 extends Parent{
    void message(){
        System.out.println("This is second subclass");
    }
}
```

```
/*
 * 2. Create an abstract class 'Bank' with an abstract method 'getBalances. $100, $150 and $200 are deposited in banks
 * A, B and C respectively. 'BankA', 'BankB' and 'BankC' are subclasses of class 'Bank', each having a method named
 * 'getBalance'. Call this method by creating an object of each of the three classes.
 */
public class callingBank {
    public static void main(String[] args) {
        BankA objBankA = new BankA();
        BankB objBankB = new BankB();
        BankC objBankC = new BankC();
        objBankA.getBalance();
        objBankB.getBalance();
        objBankC.getBalance();
    }
}

abstract class Bank{
    void getBalance(){
    }
}

class BankA extends Bank{
    void getBalance(){
        System.out.println("$100");
    }
}

class BankB extends Bank{
    void getBalance(){
        System.out.println("$150");
    }
}

class BankC extends Bank{
    void getBalance(){
        System.out.println("$200");
    }
}
```

```
/*
 * 3. We have to 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 abstract class 'Marks' with an abstract method getPercentages.
```

```
 * 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. Create an object for each of the two classes and print the percentage of
 * marks for both the students.
 */
import java.util.Scanner;

public class calculatePersentage {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.print("Input the marks of Student A: ");
        double marks1 = scan.nextDouble();
        double marks2 = scan.nextDouble();
        double marks3 = scan.nextDouble();
        A objA = new A(marks1, marks2, marks3);
        objA.getPercentage();

        System.out.print("Input the marks of Student B: ");
        marks1 = scan.nextDouble();
        marks2 = scan.nextDouble();
        marks3 = scan.nextDouble();
        double marks4 = scan.nextDouble();
        B objB = new B(marks1, marks2, marks3, marks4);
        objB.getPercentage();

        scan.close();
    }
}

abstract class Marks{
    void getPercentage(){}
}

class A extends Marks{
    double marks1, marks2, marks3;
    A(double marks12, double marks22, double marks32){
        this.marks1 = marks12;
        this.marks2 = marks22;
        this.marks3 = marks32;
    }

    void getPercentage(){
        double totalMarks = marks1 + marks2 + marks3;
        double percentage = (totalMarks / 300) * 100;
        System.out.println("The percentage of Student A is: " + percentage);
    }
}

class B extends Marks{
    double marks1, marks2, marks3, marks4;
    B(double marks12, double marks22, double marks32, double marks42){
        this.marks1 = marks12;
        this.marks2 = marks22;
        this.marks3 = marks32;
        this.marks4 = marks42;
    }

    void getPercentage(){
        double totalMarks = marks1 + marks2 + marks3 + marks4;
        double percentage = (totalMarks / 400) * 100;
        System.out.println("The percentage of Student B is: " + percentage);
    }
}
```

```
/*
 * 4. An abstract class has a constructor which prints "This is constructor of abstract class", an abstract method named
 * 'a method' and a non-abstract method which prints "This is a normal method of abstract class". A class 'SubClass' inherits
 * the abstract class and has a method named 'a_method' which prints "This is abstract method". Now create an object of
 * 'SubClass' and call the abstract method and the non-abstract method. (Analyze the result)
 */
public class problem4 {
    public static void main(String[] args) {
        SubClass objSubClass = new SubClass();
        objSubClass.a_method();
        objSubClass.normal_method();
    }
}

abstract class abstractClass{
    abstractClass(){
        System.out.println("This is constructor of abstract class");
```

```
    }

    abstract void a_method();

    void normal_method(){
        System.out.println("This is a normal method of abstract class");
    }
}

class SubClass extends abstractClass{
    @Override
    void a_method(){
        System.out.println("This is abstract method");
    }
}
```

```
/*
 * 5. Create an abstract class 'Animals' with two abstract methods 'cats' and 'dogs'. Now create a class 'Cats' with a method
 * 'cats' which prints "Cats meow" and a class 'Dogs' with a method 'dogs' which prints "Dogs bark", both inheriting the class
 * 'Animals'. Now create an object for each of the subclasses and call their respective methods.
 */
public class problem5 {
    public static void main(String[] args) {
        Animals Dog= new Dogs();
        Animals Cat= new Cats();

        Cat.cats();
        Dog.dogs();
    }
}

abstract class Animals {
    abstract void dogs();
    abstract void cats();
}

class Cats extends Animals {
    @Override
    void cats() {
        System.out.println("Cat meow!");
    }

    @Override
    void dogs() {
        // unimplemented abstract method

    }
}

class Dogs extends Animals {
    @Override
    void dogs() {
        System.out.println("Dog barks!");
    }

    @Override
    void cats() {
        // unimplemented abstract method

    }
}
```

```
/*
 * 6. We have to calculate the area of a rectangle, a square and a circle. Create an abstract class 'Shape' with three abstract
 * methods namely 'RectangleArea' taking two parameters, 'SquareArea' and 'CircleArea' taking one parameter each. The parameters
 * of 'RectangleArea' are its length and breadth, that of 'SquareArea' is its side and that of 'CircicArea' is its radius.
 * Now create another class 'Area' containing all the three methods 'RectangleArea', 'SquareArea' and 'CircleArea' for printing
 * the area of rectangle, square and circle respectively. Create an object of class 'Area' and call all the three methods.
 */
import java.lang.Math;
public class problem6 {
    public static void main(String[] args) {
        Area objArea = new Area();

        objArea.RectangleArea(Math.random()*10, Math.random()*10);
        objArea.SquareArea(Math.random()*10);
        objArea.CircleArea(Math.random()*10);
    }
}
```

```
abstract class Shape {
    abstract void RectangleArea(double h, double w);
    abstract void SquareArea(double h);
    abstract void CircleArea(double r);
}

class Area extends Shape {
    @Override
    void RectangleArea(double h, double w) {
        System.out.println("Area of Rectangle is " + (h*w));
    }

    @Override
    void SquareArea(double h) {
        System.out.println("Area of square is " + (h*h));
    }

    @Override
    void CircleArea(double r) {
        System.out.println("Area of Circle is " + (Math.PI*r*r));
    }
}
```

```
/* 7. We have to calculate the area of a rectangle, a square and a circle. Create an abstract class 'Shape' with three abstract
 * methods namely 'RectangleArea' taking two parameters, 'SquareArea' and 'CircleArea' taking one parameter each. The parameters
 * of 'RectangleArea' are its length and breadth, that of 'SquareArea' is its side and that of 'CircicArea' is its radius.
 * Now create another class 'Area' containing all the three methods 'RectangleArea', 'SquareArea' and 'CircleArea' for printing
 * the area of rectangle, square and circle respectively. Create an object of class 'Area' and call all the three methods.
 *
 * Repeat the above question for 4 rectangles, 4 squares and 5 circles. Hint- Use array of objects.
 */
public class problem7 {
    public static void main(String[] args) {
        Area[] objArea = new Area[5];

        for (int i = 0; i < 4; i++) {
            objArea[i] = new Area();
            System.out.println("Shape Number: " + (i+1));
            objArea[i].RectangleArea(Math.random()*10, Math.random()*10);
            objArea[i].SquareArea(Math.random()*10);
            objArea[i].CircleArea(Math.random()*10);
            System.out.println("");
        }

        objArea[4] = new Area();
        System.out.println("Shape Number: " + 5);
        objArea[4].CircleArea(Math.random()*10);
        System.out.println("");
    }
}

abstract class Shape {
    abstract void RectangleArea(double h, double w);
    abstract void SquareArea(double h);
    abstract void CircleArea(double r);
}

class Area extends Shape {
    @Override
    void RectangleArea(double h, double w) {
        System.out.println("Area of Rectangle is " + (h*w));
    }

    @Override
    void SquareArea(double h) {
        System.out.println("Area of square is " + (h*h));
    }

    @Override
    void CircleArea(double r) {
        System.out.println("Area of Circle is " + (Math.PI*r*r));
    }
}
```

```
/* 8. Create an interface TVremote and use it to inherit another interface smart TVremote.
 * Create a class TV which implements TVremote interface.
 */
```

```
public class problem8 {
    public static void main(String[] args) {

    }
}

interface TVremote {
    //Properties of this interface
}

interface smartTVremote {
    //Properties of this interface
}

class TV implements TVremote {
    //Properties of this interface
}
```