**Assignment 2**

1. **The Basic Box**  
   Create a class named Box. It should have three data members: length, width, and height. Create a method calculateVolume() that prints the volume of the box (length \* width \* height). In a main method, create one Box object, assign values, and call its method.

CODE::

class Box {

double length, width, height;

void calculateVolume() {

double volume = length \* width \* height;

System.out.println("The volume of the box is: " + volume);

}

}

public class Main {

public static void main(String[] args) {

Box myBox = new Box();

myBox.length = 5.0;

myBox.width = 3.0;

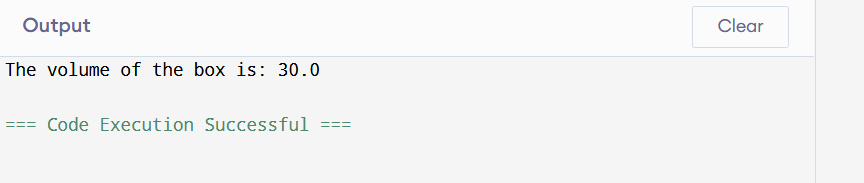
myBox.height = 2.0;

myBox.calculateVolume();

}

}

OUTPUT



1. **Overload the Constructor**  
   Modify your Box class from question 1. Create two constructors:  
   \* A **default constructor** that sets all dimensions to 1.  
   \* A **parameterized constructor** that takes l, w, h and assigns them.  
   In the main method, create two Box objects using each constructor and call calculateVolume() for both.

CODE:

class Box {

// Data members

double length, width, height;

// Default constructor (all dimensions = 1)

Box() {

length = 1;

width = 1;

height = 1;

}

// Parameterized constructor

Box(double l, double w, double h) {

length = l;

width = w;

height = h;

}

// Method to calculate volume

void calculateVolume() {

double volume = length \* width \* height;

System.out.println("The volume of the box is: " + volume);

}

}

public class Main {

public static void main(String[] args) {

Box box1 = new Box();

box1.calculateVolume();

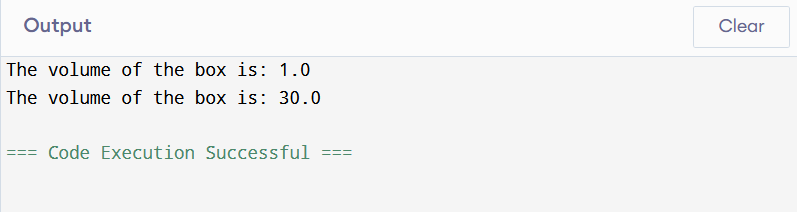
Box box2 = new Box(5.0, 3.0, 2.0);

box2.calculateVolume();

}

}

OUTPUT



1. **Simple Overloading (Area Calculator)**  
   Create a class Calculator. Overload a method named findArea:  
   \* findArea(int side) : calculates the area of a square (side \* side) and prints it.  
   \* findArea(int length, int breadth) : calculates the area of a rectangle (length \* breadth) and prints it.  
   In the main method, call both versions.

CODE:

class Calculator {

void findArea(int side) {

int area = side \* side;

System.out.println("Area of square: " + area);

}

void findArea(int length, int breadth) {

int area = length \* breadth;

System.out.println("Area of rectangle: " + area);

}

}

public class Main {

public static void main(String[] args) {

Calculator calc = new Calculator();

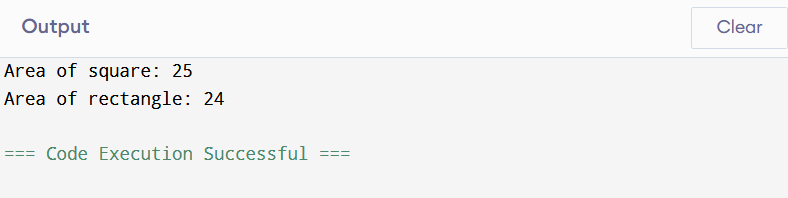
calc.findArea(5);

calc.findArea(4, 6);

}

}

OUTPUT



1. **The Bank Account**  
   Create a class BankAccount with data members accountNumber and balance. Create two methods:  
   \* deposit(int amount) : adds the amount to the balance.  
   \* withdraw(int amount) : subtracts the amount from the balance, but only if there are sufficient funds.  
   In main, create an account, deposit some money, withdraw some, and print the final balance.

CODE:

class BankAccount {

int accountNumber;

double balance;

BankAccount(int accNo, double initialBalance) {

accountNumber = accNo;

balance = initialBalance;

}

void deposit(int amount) {

balance += amount;

System.out.println("Deposited: " + amount);

}

void withdraw(int amount) {

if (amount <= balance) {

balance -= amount;

System.out.println("Withdrawn: " + amount);

} else {

System.out.println("Insufficient funds! Withdrawal denied.");

}

}

void displayBalance() {

System.out.println("Final Balance: " + balance);

}

}

public class Main {

public static void main(String[] args) {

BankAccount account = new BankAccount(1001, 5000);

account.deposit(2000);

account.withdraw(1500);

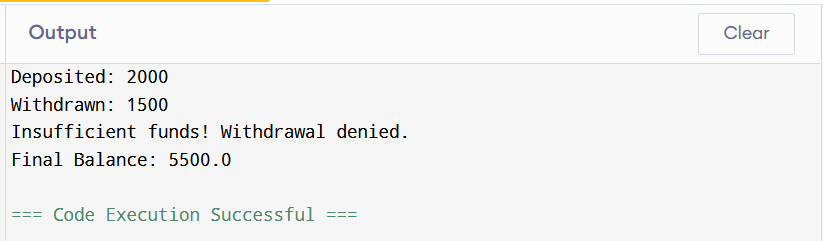
account.withdraw(6000); funds

account.displayBalance();

}

}

OUTPUT



1. **Basic Overriding (Animals)**  
   Create a parent class Animal with a method makeSound() that prints "Animal makes a sound". Create a child class Cat that **overrides** makeSound() to print "Meow!". In the main method, create objects of both Animal and Cat and call their makeSound() methods.

CODE:

class Animal {

void makeSound() {

System.out.println("Animal makes a sound");

}

class Cat extends Animal {

@Override

void makeSound() {

System.out.println("Meow!");

}

}

public class Main {

public static void main(String[] args) {

Animal a = new Animal();

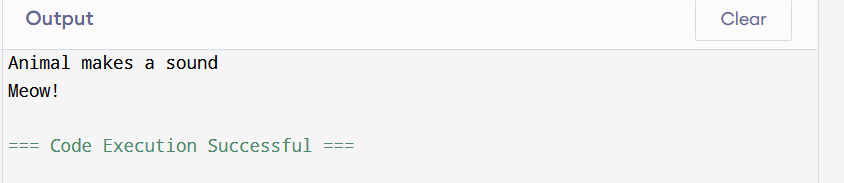
a.makeSound();

c.makeSound();

}

}

OUTPUT



1. **Using**super**in Overriding**  
   Create a class Vehicle with a method start() that prints "Vehicle is starting...". Create a subclass Car that overrides start(). The Car's start() method should first call the parent's start() method using super and then print "Car is starting with a key!".

CODE:

// Parent class

class Vehicle {

void start() {

System.out.println("Vehicle is starting...");

}

}

// Child class

class Car extends Vehicle {

@Override

void start() {

// Call parent class method

super.start();

// Add child-specific behavior

System.out.println("Car is starting with a key!");

}

}

public class Main {

public static void main(String[] args) {

// Create Car object

Car myCar = new Car();

// Call overridden start() method

myCar.start();

}

}

OUTPUT

