2. RocketTrajectory.java

```
import java.util.Scanner;

class RocketTrajectory {
    private double a; // Acceleration due to gravity
    private double b; // Initial upward velocity
    private double c; // Initial height

    // Constructor to initialize rocket parameters
    public RocketTrajectory(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    // Method to calculate the time(s) when the rocket reaches the ground (h = 0)
    public double[] getGroundTimes() {
        // Quadratic equation: h(t) = a*t^2 + b*t + c = 0
        double discriminant = b * b - 4 * a * c;
        if (discriminant < 0) {
            // No real solutions: Rocket never reaches the ground
            return new double[0];
        } else if (discriminant == 0) {
            // One real solution
            double t = -b / (2 * a);
            if (t >= 0)
                return new double[]{t};
            else
                return new double[0];
        } else {
```

```
            // Two real solutions
            double sqrtDisc = Math.sqrt(discriminant);
            double t1 = (-b + sqrtDisc) / (2 * a);
            double t2 = (-b - sqrtDisc) / (2 * a);
            // Only consider non-negative times
            if (t1 >= 0 && t2 >= 0)
                return new double[]{t1, t2};
            else if (t1 >= 0)
                return new double[]{t1};
            else if (t2 >= 0)
                return new double[]{t2};
            else
                return new double[0];
        }
    }

    // Method to get height at any time t
    public double heightAtTime(double t) {
        return a * t * t + b * t + c;
    }
}

public class RocketTrajectorySimulation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input rocket parameters
        System.out.print("Enter acceleration due to gravity (a, negative): ");
        double a = scanner.nextDouble();

        System.out.print("Enter initial upward velocity (b): ");
```

```
        double b = scanner.nextDouble();

        System.out.print("Enter initial height (c): ");
        double c = scanner.nextDouble();

        // Create an instance using constructor
        RocketTrajectory rocket = new RocketTrajectory(a, b, c);

        // Get time(s) when rocket reaches the ground
        double[] groundTimes = rocket.getGroundTimes();

        if (groundTimes.length == 0) {
            System.out.println("The rocket never reaches the ground.");
        } else {
            System.out.println("Time(s) when rocket reaches the ground:");
            for (double t : groundTimes) {
                System.out.printf("t = %.2f seconds\n", t);
            }
        }
    }
}
```

Input:

- Acceleration due to gravity $a = -9.8\,m/s^2$
- Initial upward velocity $b = 50\,m/s$
- Initial height $c = 10\,m$

Output

The rocket reaches the ground at: t = 5.53 seconds

height = 0.00 meters t = 0.18 seconds

height = 0.00 meters

3. Heartrate Program

```
import java.util.Scanner;

public class HeartRateMonitor {
    private int[] heartRates;

    // Constructor
    public HeartRateMonitor(int[] heartRates) {
        this.heartRates = heartRates;
    }

    // Method to calculate the summary value based on readings count
    public int calculateSummary() {
        int n = heartRates.length;
        if (n == 0) return 0;
        if (n % 2 == 1) {
            // Odd number of readings: sum first, middle, and last
            int middleIndex = n / 2;
            return heartRates[0] + heartRates[middleIndex] + heartRates[n - 1];
        } else {
            // Even number of readings: sum first and last
            return heartRates[0] + heartRates[n - 1];
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter number of heart rate readings: ");
        int n = scanner.nextInt();
```

```
        int[] readings = new int[n];
        System.out.println("Enter heart rate readings:");
        for (int i = 0; i < n; i++) {
            readings[i] = scanner.nextInt();
        }

        HeartRateMonitor monitor = new HeartRateMonitor(readings);
        int summaryValue = monitor.calculateSummary();

        System.out.println("Summary value based on heart rate readings: " + summaryValue);

        scanner.close();
    }
}
```

Input and Output

```
Enter number of heart rate readings: 5
Enter heart rate readings:
72
75
78
74
70
Summary value based on heart rate readings: 220
```

4. Climate Simulation System

```
import java.util.Scanner;
```

```
class ClimateVariable {
    double realPart;      // Measurable value (e.g., temperature)
    double imaginaryPart; // Anomaly or uncertainty factor

    // Default constructor - no initial data
    public ClimateVariable() {
        this.realPart = 0.0;
        this.imaginaryPart = 0.0;
    }

    // Constructor with only real data, uncertainty assumed zero
    public ClimateVariable(double realPart) {
        this.realPart = realPart;
        this.imaginaryPart = 0.0;
    }

    // Constructor with real and imaginary parts
    public ClimateVariable(double realPart, double imaginaryPart) {
        this.realPart = realPart;
        this.imaginaryPart = imaginaryPart;
    }

    // Copy constructor for duplicating an existing ClimateVariable
    public ClimateVariable(ClimateVariable other) {
        this.realPart = other.realPart;
        this.imaginaryPart = other.imaginaryPart;
    }

    // Method to display the climate variable in standard complex format
    public String toString() {
        return String.format("%.2f + %.2fi", realPart, imaginaryPart);
```

```
    }

    // Method to add another ClimateVariable's values to this one (example manipulation)
    public ClimateVariable add(ClimateVariable other) {
        return new ClimateVariable(this.realPart + other.realPart,
                                   this.imaginaryPart + other.imaginaryPart);
    }
}

public class ClimateSimulation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter real part for partial data ClimateVariable:");
        double realOnly = scanner.nextDouble();

        System.out.println("Enter real and imaginary parts for full data ClimateVariable:");
        double realFull = scanner.nextDouble();
        double imagFull = scanner.nextDouble();

        ClimateVariable cvDefault = new ClimateVariable();
        ClimateVariable cvPartial = new ClimateVariable(realOnly);
        ClimateVariable cvFull = new ClimateVariable(realFull, imagFull);
        ClimateVariable cvCopy = new ClimateVariable(cvFull);

        System.out.println("\nClimate Variables:");
        System.out.println("Default (no initial data): " + cvDefault);
        System.out.println("Partial data (real only): " + cvPartial);
        System.out.println("Full data (real + imaginary): " + cvFull);
        System.out.println("Copy of full data variable: " + cvCopy);
```

```
        ClimateVariable sumVariable = cvPartial.add(cvFull);
        System.out.println("\nSum of partial and full climate variables: " + sumVariable);

        scanner.close();
    }
}
```

Input:

Enter real part for partial data ClimateVariable:

15.3

Enter real and imaginary parts for full data ClimateVariable:

22.5

3.8

Output

Climate Variables:

Default (no initial data): 0.00 + 0.00i

Partial data (real only): 15.30 + 0.00i

Full data (real + imaginary): 22.50 + 3.80i

Copy of full data variable: 22.50 + 3.80i

Sum of partial and full climate variables: 37.80 + 3.80i

6. Box Program using the packages

Class1:

```
package logistics.box;

public class Box {
```

```
    public double width, height, depth;

    public Box() {
        width = height = depth = 0;
    }

    public Box(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
    }

    public Box(Box ob) {
        width = ob.width;
        height = ob.height;
        depth = ob.depth;
    }

    public double volume() {
        return width * height * depth;
    }
}
```

Class 2

```
package logistics.boxweight;

import logistics.box.Box;

public class BoxWeight extends Box {
    public double weight;
```

```
    public BoxWeight() {
        super();
        weight = 0;
    }

    public BoxWeight(double w, double h, double d, double m) {
        super(w, h, d);
        weight = m;
    }

    public BoxWeight(BoxWeight ob) {
        super(ob);
        weight = ob.weight;
    }
}
```

Class 3

```
package logistics.shipment;

import logistics.boxweight.BoxWeight;

public class Shipment extends BoxWeight {
    public double cost;

    public Shipment() {
        super();
        cost = 0;
    }

    public Shipment(double w, double h, double d, double m, double c) {
```

```
        super(w, h, d, m);
        cost = c;
    }

    public Shipment(Shipment ob) {
        super(ob);
        cost = ob.cost;
    }
}
```

Class 4

```
package logistics.app;

import logistics.shipment.Shipment;
import java.util.Scanner;

public class LogisticsApp {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.println("Enter width, height, depth, weight and cost for Shipment 1:");
        double w1 = input.nextDouble();
        double h1 = input.nextDouble();
        double d1 = input.nextDouble();
        double weight1 = input.nextDouble();
        double cost1 = input.nextDouble();

        Shipment shipment1 = new Shipment(w1, h1, d1, weight1, cost1);

        System.out.println("Enter width, height, depth, weight and cost for Shipment 2:");
        double w2 = input.nextDouble();
```

```
        double h2 = input.nextDouble();
        double d2 = input.nextDouble();
        double weight2 = input.nextDouble();
        double cost2 = input.nextDouble();

        Shipment shipment2 = new Shipment(w2, h2, d2, weight2, cost2);

        Shipment shipment3 = new Shipment(shipment1);

        System.out.println("\nShipment 1 details:");
        displayShipmentDetails(shipment1);

        System.out.println("\nShipment 2 details:");
        displayShipmentDetails(shipment2);

        System.out.println("\nShipment 3 (copy of Shipment 1) details:");
        displayShipmentDetails(shipment3);

        input.close();
    }

    static void displayShipmentDetails(Shipment s) {
        System.out.printf("Dimensions (WxHxD): %.2f x %.2f x %.2f\n", s.width, s.height,
s.depth);
        System.out.printf("Volume: %.2f\n", s.volume());
        System.out.printf("Weight: %.2f\n", s.weight);
        System.out.printf("Cost: $%.2f\n", s.cost);
    }
}
```

Input

10 20 15 34.5 250.75

5 10 8 12.0 100.50

Output:

Shipment 1 details:

Dimensions (WxHxD): 10.00 x 20.00 x 15.00

Volume: 3000.00

Weight: 34.50

Cost: $250.75

Shipment 2 details:

Dimensions (WxHxD): 5.00 x 10.00 x 8.00

Volume: 400.00

Weight: 12.00

Cost: $100.50

Shipment 3 (copy of Shipment 1) details:

Dimensions (WxHxD): 10.00 x 20.00 x 15.00

Volume: 3000.00

Weight: 34.50

Cost: $250.75

7. banking application

```
import java.util.Scanner;

// Interface for fixed deposit
interface FixedDeposit {
    double calculateMaturityAmount(double principal, double rate, int years);
}
```

```
// Bank A implementation
class BankA implements FixedDeposit {
    @Override
    public double calculateMaturityAmount(double principal, double rate, int years) {
        // Compound interest formula: A = P(1 + r/100)^t
        return principal * Math.pow(1 + rate/100, years);
    }
}

// Bank B implementation with slightly different compound interest (quarterly compounding)
class BankB implements FixedDeposit {
    @Override
    public double calculateMaturityAmount(double principal, double rate, int years) {
        int n = 4; // compounded quarterly
        return principal * Math.pow(1 + (rate/(n*100)), n * years);
    }
}

public class FixedDepositSimulation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter principal amount:");
        double principal = scanner.nextDouble();

        System.out.println("Enter annual interest rate (in %):");
        double rate = scanner.nextDouble();

        System.out.println("Enter number of years:");
        int years = scanner.nextInt();
```

```
        FixedDeposit bankA = new BankA();
        FixedDeposit bankB = new BankB();

        double maturityA = bankA.calculateMaturityAmount(principal, rate, years);
        double maturityB = bankB.calculateMaturityAmount(principal, rate, years);

        System.out.printf("Maturity amount in Bank A: %.2f\n", maturityA);
        System.out.printf("Maturity amount in Bank B (quarterly compounding): %.2f\n",
maturityB);

        scanner.close();
    }
}
```

Input

Enter principal amount:

10000

Enter annual interest rate (in %):

5

Enter number of years:

2

Output:

Maturity amount in Bank A: 11025.00

Maturity amount in Bank B (quarterly compounding): 11038.14

8. Civil Engineering

Code:

```
import java.util.Scanner;
```

```
// Abstract class Solid
abstract class Solid {
    double radius;
    double height;

    // Constructor to initialize radius and height (height may be 0 for Sphere)
    Solid(double radius, double height) {
        this.radius = radius;
        this.height = height;
    }

    // Abstract method to calculate surface area (for coating)
    abstract double surfaceArea();

    // Abstract method to calculate volume (for filling materials)
    abstract double volume();
}

// Cylinder class
class Cylinder extends Solid {

    Cylinder(double radius, double height) {
        super(radius, height);
    }

    @Override
    double surfaceArea() {
        // Surface Area of Cylinder = 2πr(h + r)
        return 2 * Math.PI * radius * (height + radius);
    }
```

```
    @Override
    double volume() {
        // Volume of Cylinder = πr²h
        return Math.PI * radius * radius * height;
    }
}

// Cone class
class Cone extends Solid {

    Cone(double radius, double height) {
        super(radius, height);
    }

    @Override
    double surfaceArea() {
        // Surface Area of Cone = πr(r + l), where l = slant height = sqrt(r² + h²)
        double slantHeight = Math.sqrt(radius * radius + height * height);
        return Math.PI * radius * (radius + slantHeight);
    }

    @Override
    double volume() {
        // Volume of Cone = (1/3)πr²h
        return (1.0/3) * Math.PI * radius * radius * height;
    }
}

// Sphere class
class Sphere extends Solid {
```

```
    Sphere(double radius) {
        super(radius, 0);
    }

    @Override
    double surfaceArea() {
        // Surface Area of Sphere = 4πr²
        return 4 * Math.PI * radius * radius;
    }

    @Override
    double volume() {
        // Volume of Sphere = (4/3)πr³
        return (4.0/3) * Math.PI * radius * radius * radius;
    }
}

public class CivilEngineeringTool {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Cylinder input
        System.out.println("Enter radius and height of the Cylinder:");
        double cylRadius = scanner.nextDouble();
        double cylHeight = scanner.nextDouble();
        Cylinder cylinder = new Cylinder(cylRadius, cylHeight);

        // Cone input
        System.out.println("Enter radius and height of the Cone:");
        double coneRadius = scanner.nextDouble();
```

```
        double coneHeight = scanner.nextDouble();
        Cone cone = new Cone(coneRadius, coneHeight);

        // Sphere input
        System.out.println("Enter radius of the Sphere:");
        double sphereRadius = scanner.nextDouble();
        Sphere sphere = new Sphere(sphereRadius);

        System.out.printf("\nCylinder Surface Area: %.2f\n", cylinder.surfaceArea());
        System.out.printf("Cylinder Volume: %.2f\n", cylinder.volume());

        System.out.printf("\nCone Surface Area: %.2f\n", cone.surfaceArea());
        System.out.printf("Cone Volume: %.2f\n", cone.volume());

        System.out.printf("\nSphere Surface Area: %.2f\n", sphere.surfaceArea());
        System.out.printf("Sphere Volume: %.2f\n", sphere.volume());

        scanner.close();
    }
}
```

Input:

Enter radius and height of the Cylinder:

3

5

Enter radius and height of the Cone:

4

6

Enter radius of the Sphere:

2.5

Output

Cylinder Surface Area: 150.80

Cylinder Volume: 141.37

Cone Surface Area: 157.08

Cone Volume: 100.53

Sphere Surface Area: 78.54

Sphere Volume: 65.45