

CSE1007: JAVA PROGRAMMING

PROGRAMME: B.TECH BRANCH: ECE

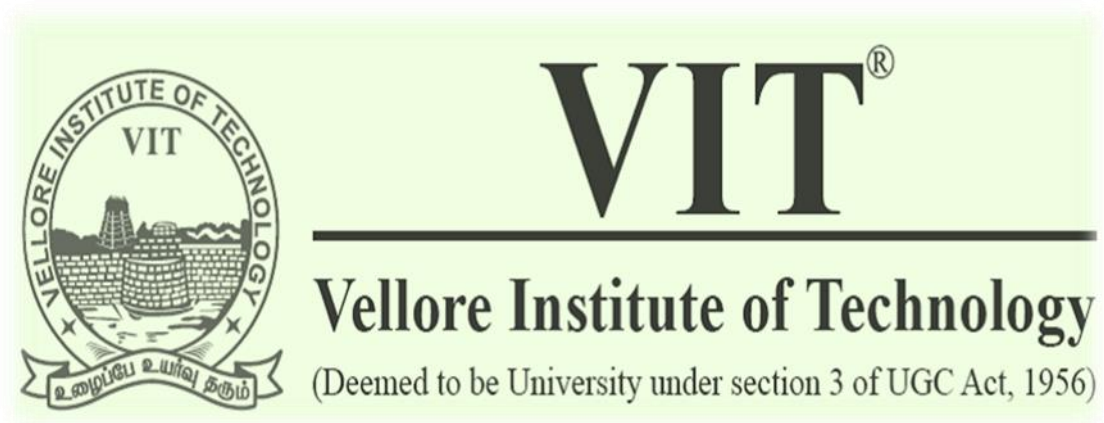
CLASS NUMBER: VL2023240105377

SLOT: L39+L40 DATE: 05-09-2023 TIME: 3:50-5:30 VENUE: SJTG17

REG NO: 20BEC0512

NAME: B. KARTHIK REDDY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING



LAB ASSESSMENT - 2

1. Overloads the methods to compute area and volume of different figures (at least 3 different figures)
2. Create a class box with the data members length, breadth, and height. The volume of the box can be calculated as $\text{length} \times \text{breadth} \times \text{height}$. Create three boxes, where the length, breadth, and height of the third box is the sum of the length, breadth and height of first two boxes. Calculate the volume of the three boxes. Use appropriate constructors and methods.
3. Create a class Student with data members Reg No., Name, Address, Phone No, Marks in six subjects and gpa. Implement methods to find the average of the marks and GPA of the students. Display the details of 'n' students along with their marks and gpa. Whenever a student object is created, a unique running Reg No. has to be automatically created. Overload the constructors.

4. Create a package Company. Define a class CVehicle . Include data members such as name and no. of wheels. Inherit two classes CHeavyMotor and CLightMotor. CHeavyMotor includes specialized attributes such as load and type of permit(state/ country/ international) and CLightMotor includes specialized attributes such as speed limit. Include parameterized constructors in each of the classes and also methods to read and print members in all classes. To initialize base class members call base class constructor from the sub class constructor. Write an application to test it.
5. An interface called IIssRec represents issue and return as follows:
interface IIssRec
{
int issue();
int receive();
}
Create a book class which implements this interface wherein issue decreases the no.of copies by 1 and receive increases the no.of copies by 1. Write an application to test it at interface level.
6. An item must be shipped from a source to destination. If it is a box, the cost is based on the weight and the distance (between source and destination). Create a user defined exception to handle If the weight is less than 1Kg throw an Exception. On the other hand, if it is a document it is based on the number of pages and the distance(between source and destination). Write an abstract class for Item and design the abstract method computeCost(). Have subclasses for Document and Box. Write an application class which tests the abstract method through dynamic binding.

1. Overloads the methods to compute area and volume of different figures (at least 3 different figures)

Approach:

1. Create a main method in the overloading class to serve as the entry point for the program.
2. Calculate and display the area and volume of a cube: a. Prompt the user to enter the side length of the cube. b. Create an instance of the shape class. c. Call the area and volume methods with the side length as an argument.
3. Calculate and display the area and volume of a cylinder: a. Prompt the user to enter the radius and height of the cylinder. b. Call the area and volume methods with the radius and height as arguments.
4. Calculate and display the area and volume of a cuboid: a. Prompt the user to enter the length, breadth, and height of the cuboid. b. Call the area and volume methods with the length, breadth, and height as arguments.
5. Close the Scanner to release system resources.

The shape class contains overloaded methods for calculating the area and volume of each figure (cube, cuboid, and cylinder) based on the provided arguments. These methods are

differentiated by their parameter lists, allowing the appropriate method to be called based on the number and types of arguments passed.

Program code:

```
import java.util.Scanner;
public class overloading {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("this program calculates the area and volume of cube,cubiod and cylinder");
        // Cube figure:
        System.out.println("enter the side of the cube : ");
        int x = sc.nextInt();
        // shape class which contains different methods
        shape s = new shape();
        s.area(x);
        s.volume(x);
        // cylinder figure:
        System.out.println("enter the radius of the cylinder :");
        int y1 = sc.nextInt();
        System.out.println("enter the height of the cylinder :");
        int y2 = sc.nextInt();
        s.area(y1, y2);
        s.volume(y1, y2);
        // cubiod figure:
        System.out.println("enter the length of the cube : ");
        int x1 = sc.nextInt();
        System.out.println("enter the breadth of the cube : ");
        int x2 = sc.nextInt();
        System.out.println("enter the height of the cube : ");
        int x3 = sc.nextInt();
        s.area(x1, x2, x3);
        s.volume(x1, x2, x3);
        sc.close();
    }
}
class shape {
    public void area(int side) {
        System.out.println("the surface area of the cube is " + side * side);
    }
    public void area(int length, int breadth, int height) {
        System.out.println("the surface area of the cuboid is "
            + 2 * ((length * breadth) + (breadth * height) + (length * height)));
    }
    public void area(int radius, int height) {
        System.out.println("the surface area of the cylinder is "
            + (2 * (3.14) * radius * height + 2 * (3.14) * (radius * radius)));
    }
    public void volume(int side) {
        System.out.println("the volume of the cube is " + side * side * side);
    }
    public void volume(int length, int breadth, int height) {
        System.out.println("the volume of the cuboid is " + length * breadth * height);
    }
    public void volume(int radius, int height) {
        System.out.println("the volume of the cylinder is " + (3.14) * (radius * radius) * height);
    }
}
```

Code screenshot:

```
1 import java.util.Scanner;
2
3 public class overloading {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         System.out.println("this program calculates the area and volume of cube,cuboid and cylinder");
7         // Cube figure:
8         System.out.println("enter the side of the cube : ");
9         int x = sc.nextInt();
10        // shape class which contains different methods
11        shape s = new shape();
12        s.area(x);
13        s.volume(x);
14        // cylinder figure:
15        System.out.println("enter the radius of the cylinder :");
16        int y1 = sc.nextInt();
17        System.out.println("enter the height of the cylinder :");
18        int y2 = sc.nextInt();
19        s.area(y1, y2);
20        s.volume(y1, y2);
21        // cuboid figure:
22        System.out.println("enter the length of the cube : ");
23        int x1 = sc.nextInt();
24        System.out.println("enter the breadth of the cube : ");
25        int x2 = sc.nextInt();
26        System.out.println("enter the height of the cube : ");
27        int x3 = sc.nextInt();
28        s.area(x1, x2, x3);
29        s.volume(x1, x2, x3);
30        sc.close();
31    }
32 }
33
34
35 class shape {
36     public void area(int side) {
37         System.out.println("the surface area of the cube is " + side * side);
38     }
39
40     public void area(int length, int breadth, int height) {
41         System.out.println("the surface area of the cuboid is "
42             + 2 * ((length * breadth) + (breadth * height) + (length * height)));
43     }
44
45     public void area(int radius, int height) {
46         System.out.println("the surface area of the cylinder is "
47             + (2 * (3.14) * radius * height) + 2 * (3.14) * (radius * radius));
48     }
49
50     public void volume(int side) {
51         System.out.println("the volume of the cube is " + side * side * side);
52     }
53
54     public void volume(int length, int breadth, int height) {
55         System.out.println("the volume of the cuboid is " + length * breadth * height);
56     }
57
58     public void volume(int radius, int height) {
59         System.out.println("the volume of the cylinder is " + (3.14) * (radius * radius) * height);
60     }
61 }
62 }
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  + - 2: Run: overloading  [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
ding'
this program calculates the area and volume of cube,cuboid and cylinder
enter the side of the cube :
5
the surface area of the cube is 25
the volume of the cube is 125
enter the radius of the cylinder :
4
enter the height of the cylinder :
2
the surface area of the cylinder is 150.72
the volume of the cylinder is 100.48
enter the length of the cube :
6
enter the breadth of the cube :
5
enter the height of the cube :
4
the surface area of the cuboid is 148
the volume of the cuboid is 120
PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMENT-2>
```

- 2. Create a class box with the data members length, breadth, and height. The volume of the box can be calculated as length*breadth*height. Create three boxes, where the length, breadth, and height of the third box is the sum of the length, breadth and height of first two boxes. Calculate the volume of the three boxes. Use appropriate constructors and methods.**

Approach:

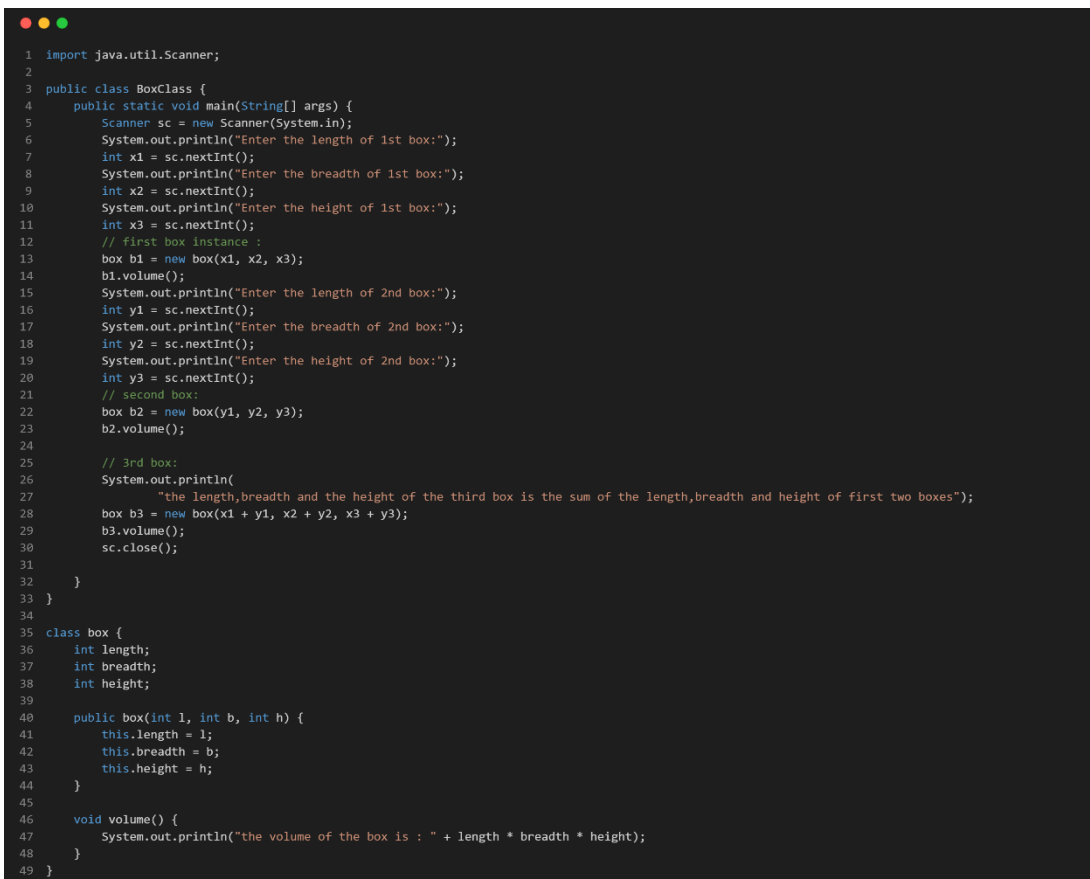
1. Prompt the user to enter the length, breadth, and height of the first box, and store these values in the variables x1, x2, and x3, respectively.
2. Create an instance of the box class (a class representing a box) named b1 with the dimensions provided by the user.
3. Call the volume method of b1 to calculate and display the volume of the first box.
4. Prompt the user to enter the length, breadth, and height of the second box, and store these values in the variables y1, y2, and y3, respectively.
5. Create an instance of the box class named b2 with the dimensions provided by the user.
6. Call the volume method of b2 to calculate and display the volume of the second box.
7. Calculate the dimensions of the third box by summing up the corresponding dimensions of the first and second boxes. Display a message explaining this.
8. Create an instance of the box class named b3 with the dimensions of the third box.
9. Call the volume method of b3 to calculate and display the volume of the third box.

Program code:

```
import java.util.Scanner;
public class BoxClass {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the length of 1st box:");
        int x1 = sc.nextInt();
        System.out.println("Enter the breadth of 1st box:");
        int x2 = sc.nextInt();
        System.out.println("Enter the height of 1st box:");
        int x3 = sc.nextInt();
        // first box instance :
        box b1 = new box(x1, x2, x3);
        b1.volume();
        System.out.println("Enter the length of 2nd box:");
        int y1 = sc.nextInt();
        System.out.println("Enter the breadth of 2nd box:");
        int y2 = sc.nextInt();
        System.out.println("Enter the height of 2nd box:");
        int y3 = sc.nextInt();
        // second box:
        box b2 = new box(y1, y2, y3);
        b2.volume();
        // 3rd box:
        System.out.println(
            "the length,breadth and the height of the third box is the sum of the length,breadth and height of first two boxes");
        box b3 = new box(x1 + y1, x2 + y2, x3 + y3);
        b3.volume();
        sc.close();
    }
}
class box {
    int length;
```

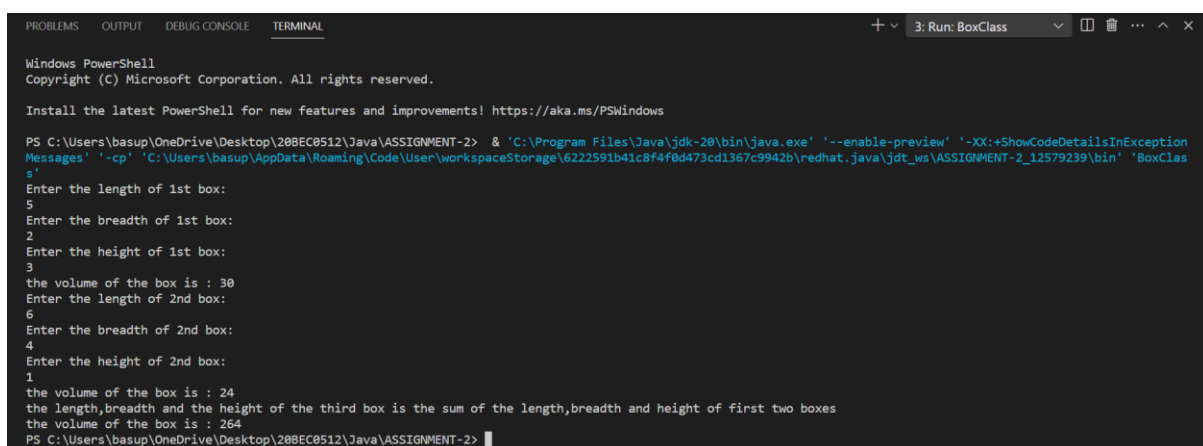
```
int breadth;  
int height;  
public box(int l, int b, int h) {  
    this.length = l;  
    this.breadth = b;  
    this.height = h;  
}  
void volume() {  
    System.out.println("the volume of the box is : " + length * breadth * height);  
}  
}
```

Code Screenshot:



```
1 import java.util.Scanner;  
2  
3 public class BoxClass {  
4     public static void main(String[] args) {  
5         Scanner sc = new Scanner(System.in);  
6         System.out.println("Enter the length of 1st box:");  
7         int x1 = sc.nextInt();  
8         System.out.println("Enter the breadth of 1st box:");  
9         int x2 = sc.nextInt();  
10        System.out.println("Enter the height of 1st box:");  
11        int x3 = sc.nextInt();  
12        // first box instance :  
13        box b1 = new box(x1, x2, x3);  
14        b1.volume();  
15        System.out.println("Enter the length of 2nd box:");  
16        int y1 = sc.nextInt();  
17        System.out.println("Enter the breadth of 2nd box:");  
18        int y2 = sc.nextInt();  
19        System.out.println("Enter the height of 2nd box:");  
20        int y3 = sc.nextInt();  
21        // second box:  
22        box b2 = new box(y1, y2, y3);  
23        b2.volume();  
24  
25        // 3rd box:  
26        System.out.println(  
27            "the length,breadth and the height of the third box is the sum of the length,breadth and height of first two boxes");  
28        box b3 = new box(x1 + y1, x2 + y2, x3 + y3);  
29        b3.volume();  
30        sc.close();  
31  
32    }  
33 }  
34  
35 class box {  
36     int length;  
37     int breadth;  
38     int height;  
39  
40     public box(int l, int b, int h) {  
41         this.length = l;  
42         this.breadth = b;  
43         this.height = h;  
44     }  
45  
46     void volume() {  
47         System.out.println("the volume of the box is : " + length * breadth * height);  
48     }  
49 }
```

Output Screenshot:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  
+ 3: Run: BoxClass  
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows  
  
PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMENT-2> & 'C:\Program Files\Java\jdk-20\bin\java.exe' '--enable-preview' '-XX:ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\basup\AppData\Roaming\Code\User\workspaceStorage\6222591b41c8f4f0d473cd1367c9942b\redhat.java\jdt_ws\ASSIGNMENT-2_12579239\bin' 'BoxClass'  
Enter the length of 1st box:  
5  
Enter the breadth of 1st box:  
2  
Enter the height of 1st box:  
3  
the volume of the box is : 30  
Enter the length of 2nd box:  
6  
Enter the breadth of 2nd box:  
4  
Enter the height of 2nd box:  
1  
the volume of the box is : 24  
the length,breadth and the height of the third box is the sum of the length,breadth and height of first two boxes  
the volume of the box is : 264  
PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMENT-2> |
```

- 3. Create a class Student with data members Reg No., Name, Address, Phone No, Marks in six subjects and gpa. Implement methods to find the average of the marks and GPA of the students. Display the details of 'n' students along with their marks and gpa. Whenever a student object is created, a unique running Reg No. has to be automatically created. Overload the constructors.**

Approach:

1. Import necessary classes, including ArrayList and List, to work with collections.
 2. Create a main method in the Student class to serve as the entry point for the program.
 3. Create an ArrayList named students to store StudentInfo objects.
 4. Set the variable n to the number of students (in this case, 2).
 5. Use a for loop to iterate through the students and create student objects with name, address, phone number, and an array of marks.
 6. Calculate and set the GPA for each student using the calculateGPA method, which averages the marks based on a total of 60 marks.
 7. Display the details of each student using the displayDetails method.
 8. Calculate the average GPA of all students using the findAverageGPA method and store it in the averageGPA variable.
 9. Calculate the average marks for each subject across all students using the findAverageMarksForAllSubjects method and store the results in the averageMarksSubjects array.
 10. Display the average GPA and average marks for each subject.
- The calculateGPA method calculates the GPA for a student based on their marks.
 - The displayDetails method displays all the information about a student.
 - The findAverageGPA method calculates the average GPA of a list of students.
 - The findAverageMarksForAllSubjects method calculates the average marks for each subject across all students.

Program code:

```
import java.util.ArrayList;
import java.util.List;

public class Student {
    public static void main(String[] args) {
        List<StudentInfo> students = new ArrayList<>();
        int n = 2; // Number of students

        for (int i = 0; i < n; i++) {
            String name = "Student " + (i + 1);
            String address = "Address " + (i + 1);
            String phoneNo = "Phone " + (i + 1);
            int[] marks = { 80, 85, 75, 90, 88, 92 };
            StudentInfo student = new StudentInfo(name, address, phoneNo, marks);
            students.add(student);
        }

        System.out.println("Student Details:");
        for (StudentInfo student : students) {
            student.displayDetails();
        }
    }
}
```

```
        System.out.println();
    }

    double averageGPA = StudentInfo.findAverageGPA(students);
    double[] averageMarksSubjects = StudentInfo.findAverageMarksForAllSubjects(students);

    System.out.println("Average GPA of all students: " + averageGPA);
    for (int i = 0; i < averageMarksSubjects.length; i++) {
        System.out.println("Average marks in Subject " + (i + 1) + ": " + averageMarksSubjects[i]);
    }
}

class StudentInfo {
    static int runningReg = 1000;
    int regNum;
    String name;
    String address;
    String phone;
    int[] marks;
    double gpa;

    // Overloading the constructor:
    public StudentInfo(String name, String address, String phone, int[] marks) {
        this.regNum = runningReg++;
        this.name = name;
        this.address = address;
        this.phone = phone;
        this.marks = marks;
        calculateGPA();
    }

    public void calculateGPA() {
        int totalMarks = 0;
        for (int i = 0; i < marks.length; i++) {
            totalMarks += marks[i];
        }
        this.gpa = (double) totalMarks / 60;
    }

    public void displayDetails() {
        System.out.println("Reg No.: " + regNum);
        System.out.println("Name: " + name);
        System.out.println("Address: " + address);
        System.out.println("Phone No.: " + phone);
        System.out.print("Marks: ");
        for (int mark : marks) {
            System.out.print(mark + " ");
        }
        System.out.println("\nGPA: " + gpa);
    }

    public static double findAverageGPA(List<StudentInfo> students) {
        double totalGPA = 0;
        for (StudentInfo student : students) {
            totalGPA += student.gpa;
        }
        return totalGPA / students.size();
    }

    public static double[] findAverageMarksForAllSubjects(List<StudentInfo> students) {
        int numSubjects = students.get(0).marks.length;
        double[] averageMarks = new double[numSubjects];

        for (StudentInfo student : students) {
            for (int i = 0; i < numSubjects; i++) {
                averageMarks[i] += student.marks[i];
            }
        }
    }
}
```



```
    }  
    }  
  
    for (int i = 0; i < numSubjects; i++) {  
        averageMarks[i] /= students.size();  
    }  
  
    return averageMarks;  
}  
}
```

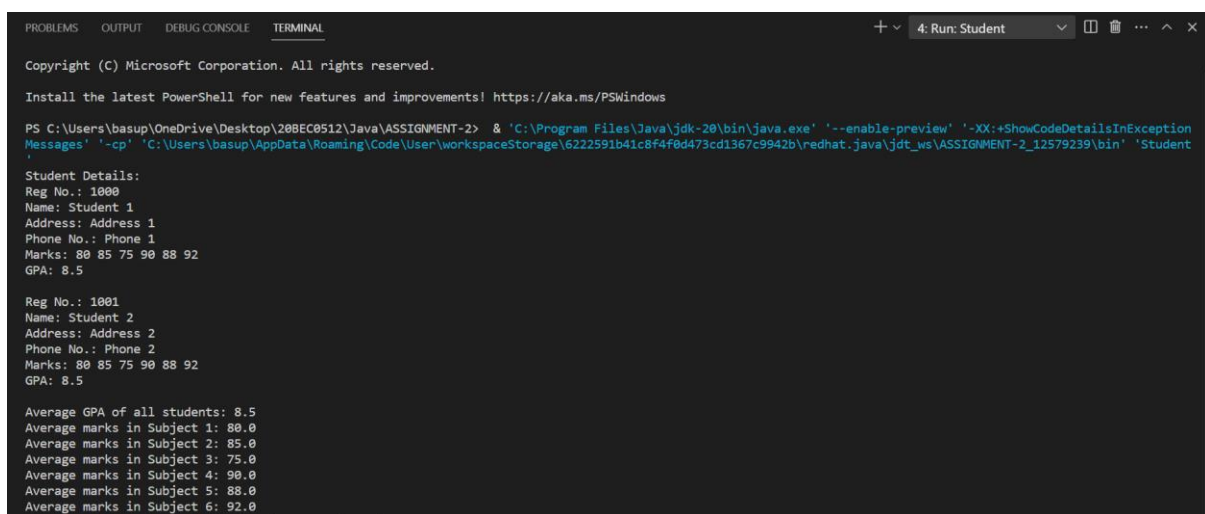
Code Screenshot:



```
1  import java.util.ArrayList;  
2  import java.util.List;  
3  
4  public class Student {  
5      public static void main(String[] args) {  
6          List<StudentInfo> students = new ArrayList<>();  
7          int n = 2; // Number of students  
8  
9          for (int i = 0; i < n; i++) {  
10             String name = "Student " + (i + 1);  
11             String address = "Address " + (i + 1);  
12             String phoneNo = "Phone " + (i + 1);  
13             int[] marks = { 80, 85, 75, 90, 88, 92 };  
14             StudentInfo student = new StudentInfo(name, address, phoneNo, marks);  
15             students.add(student);  
16         }  
17  
18         System.out.println("Student Details:");  
19         for (StudentInfo student : students) {  
20             student.displayDetails();  
21             System.out.println();  
22         }  
23  
24         double averageGPA = StudentInfo.findAverageGPA(student  
25         double[]); averageMarksSubjects = StudentInfo.findAverageMarksForAllSubjects(students);  
26  
27         System.out.println("Average GPA of all students: " + averageGPA);  
28         for (int i = 0; i < averageMarksSubjects.length; i++) {  
29             System.out.println("Average marks in Subject " + (i + 1) + ": "  
30             + averageMarksSubjects[i]);  
31         }  
32     }  
33  
34     class StudentInfo {  
35         static int runningReg = 1000;  
36         int regNum;  
37         String name;  
38         String address;  
39         String phone;  
40         int[] marks;  
41         double gpa;  
42  
43         // Overloading the constructor:  
44         public StudentInfo(String name, String address, String phone, int[] marks) {  
45             this.regNum = runningReg++;  
46             this.name = name;  
47             this.address = address;  
48             this.phone = phone;  
49             this.marks = marks;  
50             calculateGPA();  
51         }  
52     }  
53 }
```

```
52
53     public void calculateGPA() {
54         int totalMarks = 0;
55         for (int i = 0; i < marks.length; i++) {
56             totalMarks += marks[i];
57         }
58         this.gpa = (double) totalMarks / 60;
59     }
60
61     public void displayDetails() {
62         System.out.println("Reg No.: " + regNum);
63         System.out.println("Name: " + name);
64         System.out.println("Address: " + address);
65         System.out.println("Phone No.: " + phone);
66         System.out.print("Marks: ");
67         for (int mark : marks) {
68             System.out.print(mark + " ");
69         }
70         System.out.println("\nGPA: " + gpa);
71     }
72
73     public static double findAverageGPA(List<StudentInfo> students) {
74         double totalGPA = 0;
75         for (StudentInfo student : students) {
76             totalGPA += student.gpa;
77         }
78         return totalGPA / students.size();
79     }
80
81     public static double[] findAverageMarksForAllSubjects(List<StudentInfo> students) {
82         int numSubjects = students.get(0).marks.length;
83         double[] averageMarks = new double[numSubjects];
84
85         for (StudentInfo student : students) {
86             for (int i = 0; i < numSubjects; i++) {
87                 averageMarks[i] += student.marks[i];
88             }
89         }
90
91         for (int i = 0; i < numSubjects; i++) {
92             averageMarks[i] /= students.size();
93         }
94
95         return averageMarks;
96     }
97 }
```

Output Screenshot:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
+ 4: Run: Student

Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMENT-2> & 'C:\Program Files\Java\jdk-20\bin\java.exe' '-enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\basup\AppData\Roaming\Code\User\workspaceStorage\6222591b41c8f4f0d473cd1367c9942b\redhat.java\jdt_ws\ASSIGNMENT-2_12579239\bin' 'Student'

Student Details:
Reg No.: 1000
Name: Student 1
Address: Address 1
Phone No.: Phone 1
Marks: 80 85 75 90 88 92
GPA: 8.5

Reg No.: 1001
Name: Student 2
Address: Address 2
Phone No.: Phone 2
Marks: 80 85 75 90 88 92
GPA: 8.5

Average GPA of all students: 8.5
Average marks in Subject 1: 80.0
Average marks in Subject 2: 85.0
Average marks in Subject 3: 75.0
Average marks in Subject 4: 90.0
Average marks in Subject 5: 88.0
Average marks in Subject 6: 92.0
```

- 4. Create a package Company. Define a class CVehicle . Include data members such as name and no. of wheels. Inherit two classes CHeavyMotor and CLightMotor. CHeavyMotor includes specialized attributes such as load and type of permit(state/ country/ international) and CLightMotor includes specialized attributes such as speed limit. Include parameterized constructors in each of the classes and also methods to read and print members in all classes. To initialize base class members call base class constructor from the sub class constructor. Write an application to test it.**

Approach:

1. Define the Base Class CVehicle:
2. Define the CHeavyMotor Class (Subclass of CVehicle):
3. Override the display method in CHeavyMotor to include specialized attributes.
4. Define the CLightMotor Class (Subclass of CVehicle):
5. Create a subclass CLightMotor that inherits from CVehicle.
6. Add a specialized attribute speedLimit (int).
7. Implement a parameterized constructor for CLightMotor to initialize both inherited and specialized attributes.
8. Override the display method in CLightMotor to include the specialized attribute.
9. Write an Application to Test the Classes:

Program code:

Vehicle class:

```
package Company;
public class CVehicle {
    String name;
    int noOfWheels;
    public CVehicle(String name, int wheels) {
        this.name = name;
        this.noOfWheels = wheels;
    }
    public void display() {
        System.out.println("name of the vehicle : " + name);
        System.out.println("number of wheels in the vehicle: " + noOfWheels);
    }
}
```

CHeavyMotor class:

```
package Company;

public class CHeavyMotor extends CVehicle {
    int load;
    String permit;
    public CHeavyMotor(String name, int wheels, int load, String permit) {
        super(name, wheels);
        this.load = load;
        this.permit = permit;
    }
    public void display() {
        super.display();
        System.out.println("Load: " + load);
        System.out.println("Permit:" + permit);
    }
}
```

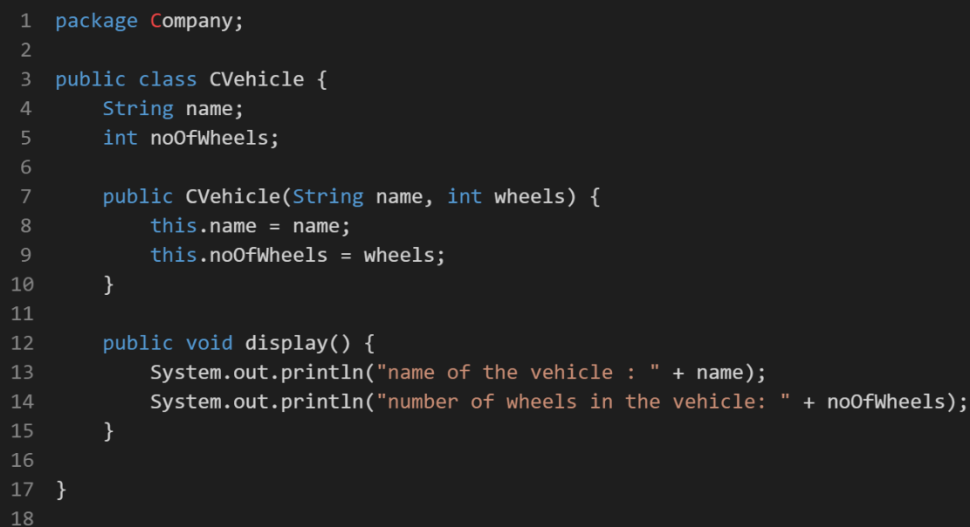
```
}
```

CLightMotor Class:

```
package Company;
public class CLightMotor extends CVehicle {
    int speedLimit;
    public CLightMotor(String name, int wheels, int speedLimit) {
        super(name, wheels);
        this.speedLimit = speedLimit;
    }
    public void display() {
        super.display();
        System.out.println("Speed Limit: " + speedLimit);
    }
}
```

Application test:

```
import Company.*;
public class companytest {
    public static void main(String[] args) {
        System.out.println("Vehicle class");
        CVehicle vehicle = new CVehicle("mahindra", 4);
        vehicle.display();
        System.out.println();
        System.out.println("CHeavy Motor class");
        CHeavyMotor heavy = new CHeavyMotor("Ashok Leyland", 6, 9, "STATE");
        heavy.display();
        System.out.println();
        System.out.println("CLight Motor class");
        CLightMotor light = new CLightMotor("Benz", 4, 120);
        light.display();
    }
}
```

Code Screenshot:**Vehicle class:**A screenshot of a code editor showing the implementation of the CVehicle class. The code is written in Java and includes package declarations, class definitions, and methods for initializing vehicle attributes and displaying them. The editor has a dark background with syntax highlighting and line numbers.

```
1 package Company;
2
3 public class CVehicle {
4     String name;
5     int noOfWheels;
6
7     public CVehicle(String name, int wheels) {
8         this.name = name;
9         this.noOfWheels = wheels;
10    }
11
12    public void display() {
13        System.out.println("name of the vehicle : " + name);
14        System.out.println("number of wheels in the vehicle: " + noOfWheels);
15    }
16
17 }
18
```

CHeavyMotor class:

```
1 package Company;
2
3 public class CHeavyMotor extends CVehicle {
4     int load;
5     String permit;
6
7     public CHeavyMotor(String name, int wheels, int load, String permit) {
8         super(name, wheels);
9         this.load = load;
10        this.permit = permit;
11    }
12
13    public void display() {
14        super.display();
15        System.out.println("Load: " + load);
16        System.out.println("Permit:" + permit);
17    }
18
19 }
20
```

CLightMotor class:

```
1 package Company;
2
3 public class CLightMotor extends CVehicle {
4     int speedLimit;
5
6     public CLightMotor(String name, int wheels, int speedLimit) {
7         super(name, wheels);
8         this.speedLimit = speedLimit;
9     }
10
11    public void display() {
12        super.display();
13        System.out.println("Speed Limit: " + speedLimit);
14    }
15
16 }
17
```

Test Application code:

```
1 import Company.*;
2
3 public class companytest {
4     public static void main(String[] args) {
5         System.out.println("Vehicle class");
6         CVehicle vehicle = new CVehicle("mahindra", 4);
7         vehicle.display();
8         System.out.println();
9         System.out.println("CHeavy Motor class");
10        CHeavyMotor heavy = new CHeavyMotor("Ashok Leyland", 6, 9, "STATE");
11        heavy.display();
12        System.out.println();
13        System.out.println("CLight Motor class");
14        CLightMotor light = new CLightMotor("Benz", 4, 120);
15        light.display();
16    }
17
18 }
```

Output Screenshot:

```

number of wheels in the vehicle: 4
Speed Limit: 120
PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMENT-2> c;; cd 'c:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMEN
T-2'; & 'C:\Program Files\Java\jdk-20\bin\java.exe' '-enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\
basup\AppData\Roaming\Code\User\workspaceStorage\6222591b41c8f4f0d473cd1367c9942b\redhat.java\jdt_ws\ASSIGNMENT-2_12579239\bin' '
companytest'
Vehicle class
name of the vehicle : mahindra
number of wheels in the vehicle: 4

CHeavy Motor class
name of the vehicle : Ashok Leyland
number of wheels in the vehicle: 6
Load: 9
Permit:STATE

CLight Motor class
name of the vehicle : Benz
number of wheels in the vehicle: 4
Speed Limit: 120
PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMENT-2> c;; cd 'c:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMEN
T-2'; & 'C:\Program Files\Java\jdk-20\bin\java.exe' '-enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\
basup\AppData\Roaming\Code\User\workspaceStorage\6222591b41c8f4f0d473cd1367c9942b\redhat.java\jdt_ws\ASSIGNMENT-2_12579239\bin' '
companytest'
Vehicle class
name of the vehicle : mahindra
number of wheels in the vehicle: 4

CHeavy Motor class
name of the vehicle : Ashok Leyland
number of wheels in the vehicle: 6
Load: 9
Permit:STATE

CLight Motor class
name of the vehicle : Benz
number of wheels in the vehicle: 4
Speed Limit: 120
PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMENT-2>

```

5. An interface called IIssRec represents issue and return as follows:
interface IIssRec

```

{
    int issue();
    int receive();
}

```

Create a book class which implements this interface wherein issue decreases the no.of copies by 1 and receive increases the no.of copies by 1. Write an application to test it at interface level.

Approach:

1. Import the necessary packages (java.util).
2. Create a Scanner object (sc) for user input.
3. Prompt the user to enter the number of initially available books (n) and store the input in the variable n.
4. Create an instance of the books class, passing the initial number of books (n) as a parameter to its constructor.
5. Display options to the user: 1 for issuing a book and 0 for receiving a book.
6. Read the user's choice (either 1 or 0) into the variable op.
7. If op is 1:
8. Call the issue() method on the books object.
9. If the issue() method returns 0, display "No books available to issue."
10. If the issue() method returns 1, display "Book issued successfully."

11. If op is 0:
12. Call the receive() method on the books object.
13. Display "Book received successfully."
14. Close the Scanner object (sc) to release system resources.

Program code:

```
import java.util.*;

public class booksinterface {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("enter the number of intial books available: ");
        int n = sc.nextInt();
        books b = new books(n);
        System.out.println("press 1 for issue");
        System.out.println("press 0 for receive");
        int op = sc.nextInt();
        if (op == 1) {
            int issue = b.issue();
            if (issue == 0) {
                System.out.println("No books availble to issue.");
            } else if (issue == 1) {
                System.out.println("Book issued successfully");
            }
        } else {
            int receive = b.receive();
            System.out.println("book received successfully");
        }
        sc.close();
    }
}

interface iisc {
    int issue();

    int receive();
}

class books implements iisc {
    int books;

    public books(int books) {
        this.books = books;
    }

    public int issue() {
        if (books > 0) {
            books--;
            return 1;
        } else {
            return 0;
        }
    }

    public int receive() {
        books++;
        return 1;
    }
}
```

Code Screenshot:

```
1  import java.util.*;
2
3  public class booksinterface {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          System.out.println("enter the number of intial books available: ");
7          int n = sc.nextInt();
8          books b = new books(n);
9          System.out.println("press 1 for issue");
10         System.out.println("press 0 for receive");
11         int op = sc.nextInt();
12         if (op == 1) {
13             int issue = b.issue();
14             if (issue == 0) {
15                 System.out.println("No books availble to issue.");
16             } else if (issue == 1) {
17                 System.out.println("Book issued successfully");
18             }
19         } else {
20             int receive = b.receive();
21             System.out.println("book received successfully");
22         }
23         sc.close();
24     }
25 }
26
27 interface iisc {
28     int issue();
29
30     int receive();
31 }
32
33
34 class books implements iisc {
35     int books;
36
37     public books(int books) {
38         this.books = books;
39     }
40
41     public int issue() {
42         if (books > 0) {
43             books--;
44             return 1;
45         } else {
46             return 0;
47         }
48     }
49
50     public int receive() {
51         books++;
52         return 1;
53     }
54 }
```


Output Screenshot:

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMENT-2> & 'C:\Program Files\Java\jdk-20\bin\java.exe' '--enable-preview'
'-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\basup\AppData\Roaming\Code\User\workspaceStorage\6222591b41c8f4f0d473cd1367c9942b\redhat.java\jdt_ws\ASSIGNMENT-2_12579239\bin' 'booksinterface'
enter the number of initial books available:
10
press 1 for issue
press 0 for receive
1
Book issued successfully
PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMENT-2> c;; cd 'c:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMEN
T-2'; & 'C:\Program Files\Java\jdk-20\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\
basup\AppData\Roaming\Code\User\workspaceStorage\6222591b41c8f4f0d473cd1367c9942b\redhat.java\jdt_ws\ASSIGNMENT-2_12579239\bin' '
booksinterface'
enter the number of initial books available:
10
press 1 for issue
press 0 for receive
0
book received successfully
PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMENT-2> c;; cd 'c:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMEN
T-2'; & 'C:\Program Files\Java\jdk-20\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\
basup\AppData\Roaming\Code\User\workspaceStorage\6222591b41c8f4f0d473cd1367c9942b\redhat.java\jdt_ws\ASSIGNMENT-2_12579239\bin' '
booksinterface'
enter the number of initial books available:
0
press 1 for issue
press 0 for receive
1
No books available to issue.
PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java\ASSIGNMENT-2>

```

- An item must be shipped from a source to destination. If it is a box, the cost is based on the weight and the distance (between source and destination). Create a user defined exception to handle If the weight is less than 1Kg throw an Exception. On the other hand, if it is a document it is based on the number of pages and the distance(between source and destination). Write an abstract class for Item and design the abstract method computeCost(). Have subclasses for Document and Box. Write an application class which tests the abstract method through dynamic binding.**

Approach:

Step 1: Define the WeightBelowMinimumException class as a custom exception that extends the built-in Exception class. This class will be used to handle cases where the weight of a box is less than 1Kg.

Step 2: Define the abstract class Item with a protected attribute distance. This class represents an item to be shipped and includes an abstract method computeCost().

Step 3: Create a subclass Document that extends Item. This subclass represents a document and includes an additional attribute numPages. Implement the computeCost() method in this subclass to calculate the shipping cost based on the number of pages and the distance.

Step 4: Create a subclass Box that extends Item. This subclass represents a box and includes an additional attribute weight. In the constructor of the Box class, check if the weight is below 1Kg, and if so, throw a WeightBelowMinimumException. Implement the computeCost() method in this subclass to calculate the shipping cost based on the weight and the distance.

Step 5: In the shippingtest class's main method:

Create an instance of Document named item1 with a distance of 100.0 and 20 pages.

Attempt to create an instance of Box named item2 with a distance of 200.0 and a weight of 0.5Kg. This will throw a WeightBelowMinimumException.

Handle the WeightBelowMinimumException in a catch block and print the exception message.

Step 6: Calculate and print the shipping cost for item1 (a Document) using its computeCost() method.

Step 7: If no exception is thrown during the execution, calculate and print the shipping cost for item2 (a Box) using its computeCost() method. However, this step will not be reached in the provided code due to the exception in step 5.

Program code:

```
public class shippingtest {
    public static void main(String[] args) {
        try {
            Item item1 = new Document(100.0, 20); // Document with 20 pages
            Item item2 = new Box(200.0, 0.5); // Box with 0.5 Kg weight (will throw an exception)

            // Compute and print the cost for the items
            System.out.println("Cost for Document: $" + item1.computeCost());
            System.out.println("Cost for Box: $" + item2.computeCost());
        } catch (WeightBelowMinimumException e) {
            System.out.println("WeightBelowMinimumException: " + e.getMessage());
        }
    }
}

// Custom exception for handling weight less than 1Kg
class WeightBelowMinimumException extends Exception {
    public WeightBelowMinimumException(String message) {
        super(message);
    }
}

// Abstract class Item
abstract class Item {
    protected double distance;

    public Item(double distance) {
        this.distance = distance;
    }

    // Abstract method to compute the cost
    public abstract double computeCost();
}

// Subclass for Document
class Document extends Item {
    private int numPages;
```

```
public Document(double distance, int numPages) {
    super(distance);
    this.numPages = numPages;
}

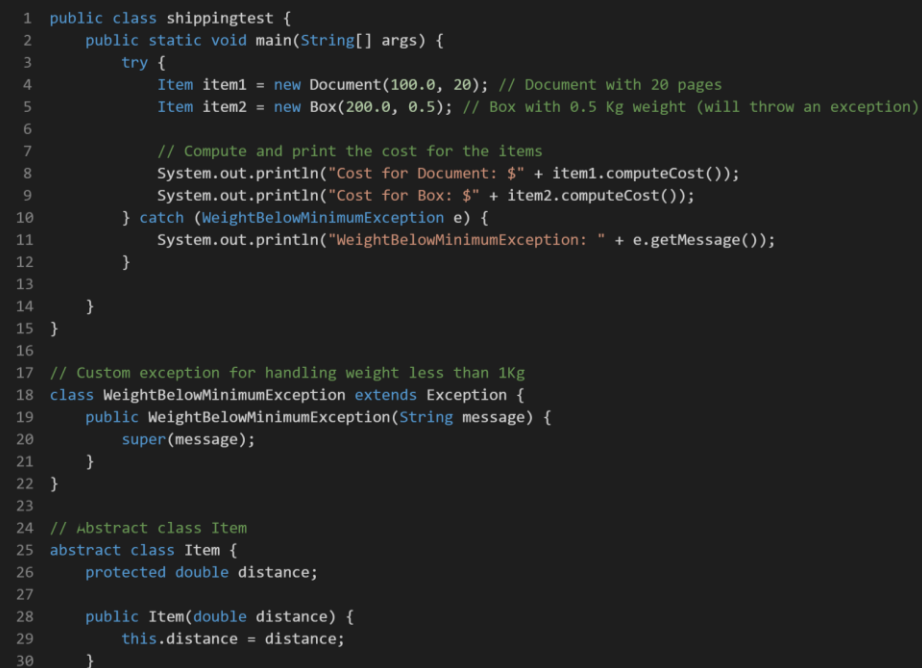
@Override
public double computeCost() {
    // Define the cost calculation for documents (e.g., based on numPages and
    // distance)
    double baseCost = 2.0; // Base cost per page
    double distanceFactor = 0.1; // Cost factor for distance
    return baseCost * numPages + distanceFactor * distance;
}

// Subclass for Box
class Box extends Item {
    private double weight;

    public Box(double distance, double weight) throws WeightBelowMinimumException {
        super(distance);
        if (weight < 1.0) {
            throw new WeightBelowMinimumException("Weight must be at least 1Kg");
        }
        this.weight = weight;
    }

    @Override
    public double computeCost() {
        // Define the cost calculation for boxes (e.g., based on weight and distance)
        double baseCost = 5.0; // Base cost per Kg
        double distanceFactor = 0.2; // Cost factor for distance
        return baseCost * weight + distanceFactor * distance;
    }
}
```

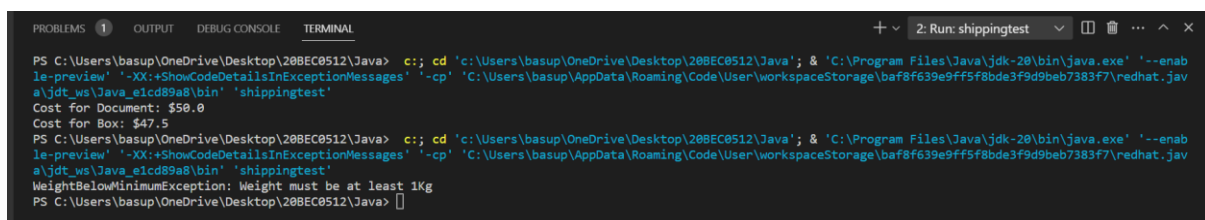
Code Screenshot:



```
1 public class shippingtest {
2     public static void main(String[] args) {
3         try {
4             Item item1 = new Document(100.0, 20); // Document with 20 pages
5             Item item2 = new Box(200.0, 0.5); // Box with 0.5 Kg weight (will throw an exception)
6
7             // Compute and print the cost for the items
8             System.out.println("Cost for Document: $" + item1.computeCost());
9             System.out.println("Cost for Box: $" + item2.computeCost());
10        } catch (WeightBelowMinimumException e) {
11            System.out.println("WeightBelowMinimumException: " + e.getMessage());
12        }
13    }
14 }
15
16 // Custom exception for handling weight less than 1Kg
17 class WeightBelowMinimumException extends Exception {
18     public WeightBelowMinimumException(String message) {
19         super(message);
20     }
21 }
22
23 // Abstract class Item
24 abstract class Item {
25     protected double distance;
26
27     public Item(double distance) {
28         this.distance = distance;
29     }
30 }
```

```
31
32 // Abstract method to compute the cost
33 public abstract double computeCost();
34 }
35
36 // Subclass for Document
37 class Document extends Item {
38     private int numPages;
39
40     public Document(double distance, int numPages) {
41         super(distance);
42         this.numPages = numPages;
43     }
44
45     @Override
46     public double computeCost() {
47         // Define the cost calculation for documents (e.g., based on numPages and
48         // distance)
49         double baseCost = 2.0; // Base cost per page
50         double distanceFactor = 0.1; // Cost factor for distance
51         return baseCost * numPages + distanceFactor * distance;
52     }
53 }
54
55 // Subclass for Box
56 class Box extends Item {
57     private double weight;
58
59     public Box(double distance, double weight) throws WeightBelowMinimumException {
60         super(distance);
61         if (weight < 1.0) {
62             throw new WeightBelowMinimumException("Weight must be at least 1Kg");
63         }
64         this.weight = weight;
65     }
66
67     @Override
68     public double computeCost() {
69         // Define the cost calculation for boxes (e.g., based on weight and distance)
70         double baseCost = 5.0; // Base cost per Kg
71         double distanceFactor = 0.2; // Cost factor for distance
72         return baseCost * weight + distanceFactor * distance;
73     }
74 }
75
```

Output Screenshot:



```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL + ~ 2: Run: shippingtest
PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java> c:\; cd 'C:\Users\basup\OneDrive\Desktop\20BEC0512\Java'; & 'C:\Program Files\Java\jdk-20\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\basup\AppData\Roaming\Code\User\workspaceStorage\baf8f639e9ff5f8bde3f9d9beb7383f7\redhat.java\jdt_ws\Java_e1cd89a8\bin' 'shippingtest'
Cost for Document: $50.0
Cost for Box: $47.5
PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java> c:\; cd 'C:\Users\basup\OneDrive\Desktop\20BEC0512\Java'; & 'C:\Program Files\Java\jdk-20\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\basup\AppData\Roaming\Code\User\workspaceStorage\baf8f639e9ff5f8bde3f9d9beb7383f7\redhat.java\jdt_ws\Java_e1cd89a8\bin' 'shippingtest'
WeightBelowMinimumException: Weight must be at least 1Kg
PS C:\Users\basup\OneDrive\Desktop\20BEC0512\Java>
```