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 ENGINEERINGA close-up of a sign

Description automatically generated

**LAB ASSESMENT - 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 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.
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.

1. 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.
2. **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:**



**Output:**

**A black screen with white text

Description automatically generated**

1. **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.**

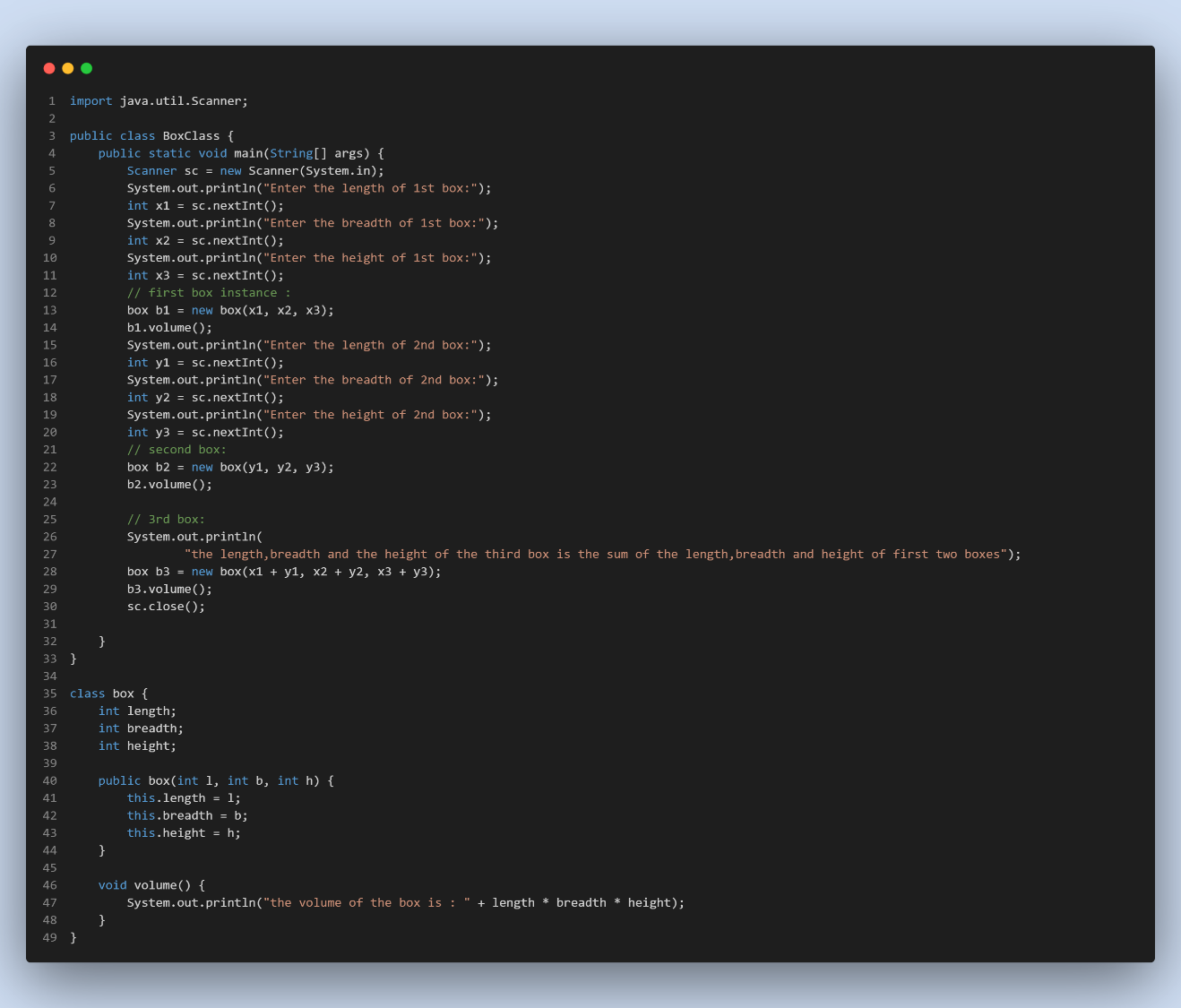
**Approach:**

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.

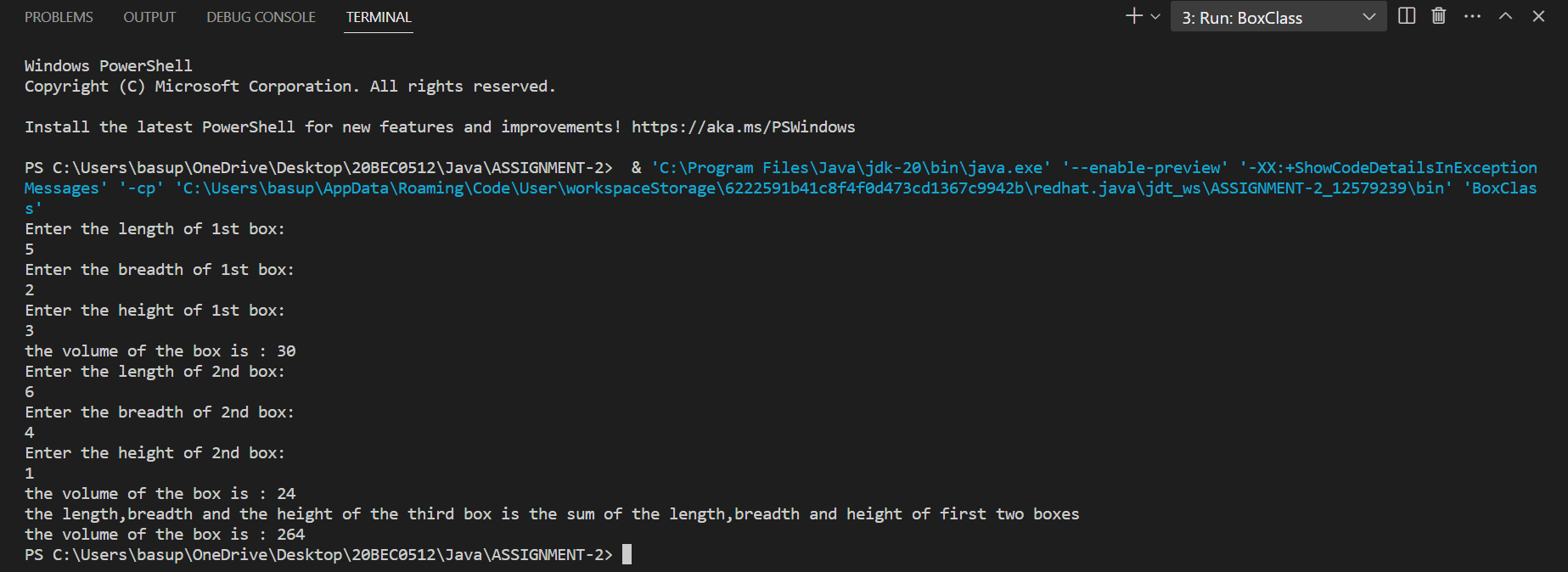
**Program code:**

|  |
| --- |
| 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:**



**Output Screenshot:**

****

1. **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;  }  } |

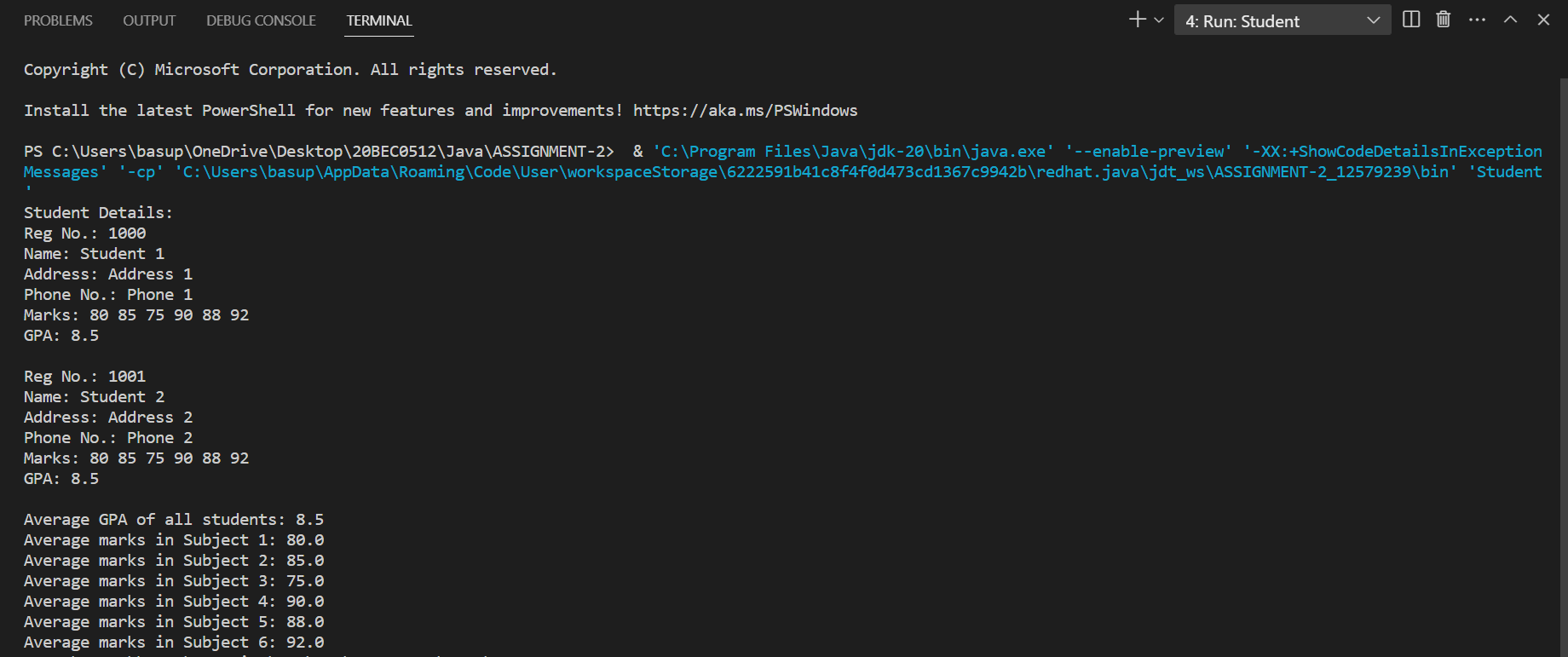
**Code Screenshot:**

A screen shot of a computer program

Description automatically generated



**Output Screenshot:**

****

1. **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:**

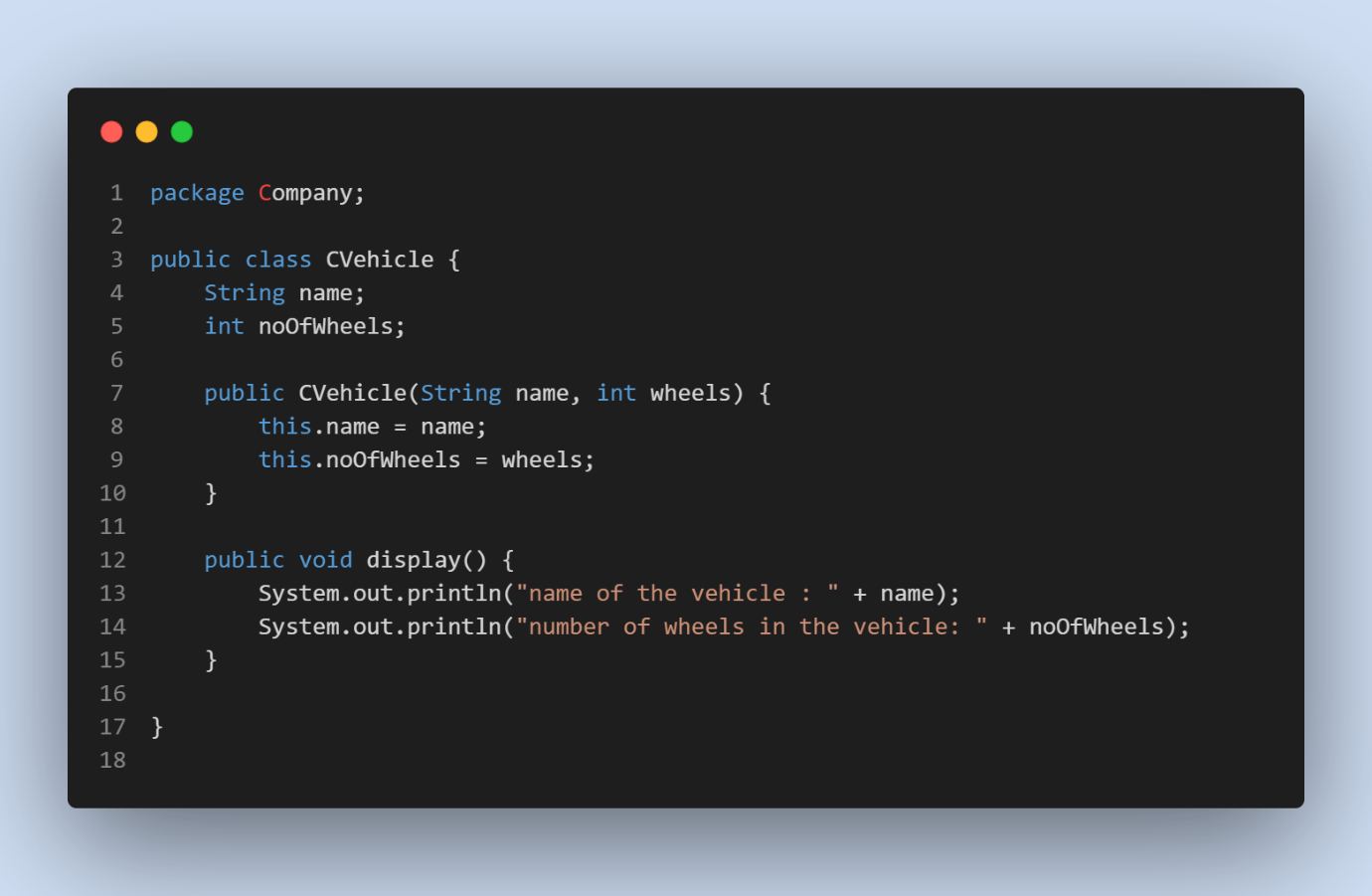
1. Define the Base Class CVehicle:
2. Define the CHeavyMotor Class (Subclass of CVehicle):
3. Override the display method in CHeavyMotor to include specialized attributes.
4. Define the CLightMotor Class (Subclass of CVehicle):
5. Create a subclass CLightMotor that inherits from CVehicle.
6. Add a specialized 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:**

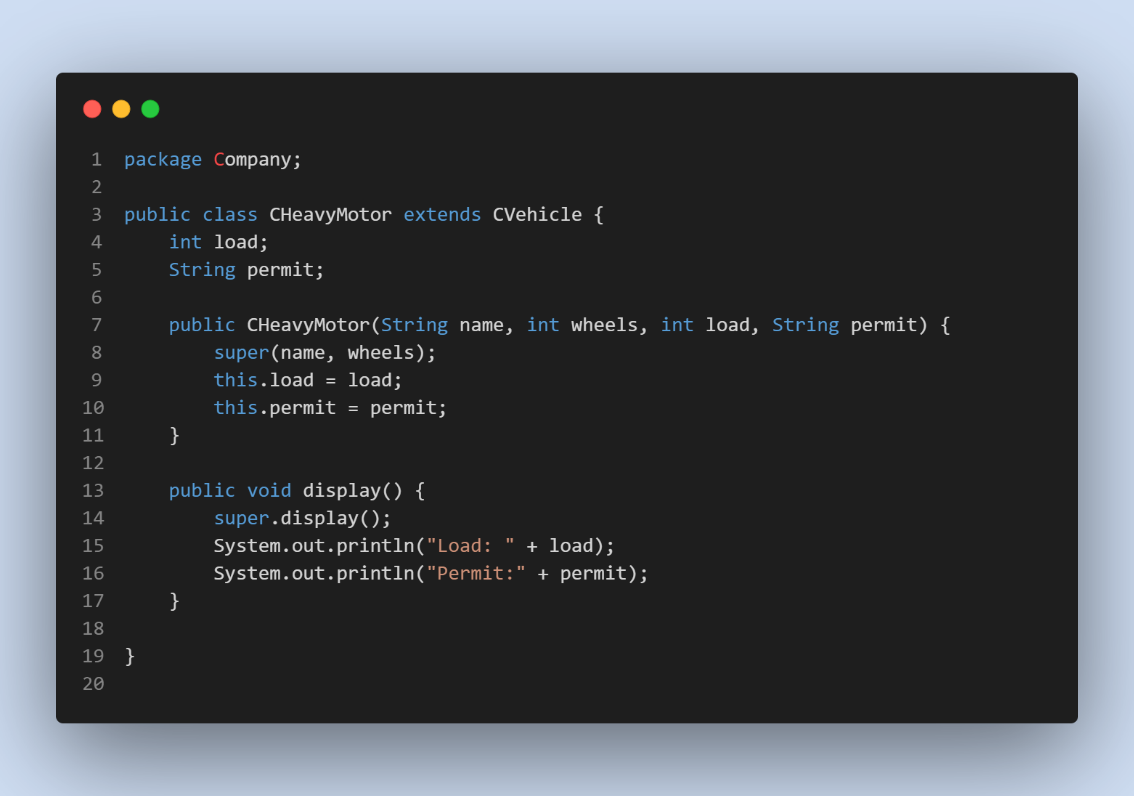
|  |
| --- |
| **Vehicle class:**  package Company;  public class CVehicle {  String name;  int noOfWheels;  public CVehicle(String name, int wheels) {  this.name = name;  this.noOfWheels = wheels;  }  public void display() {  System.out.println("name of the vehicle : " + name);  System.out.println("number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



**CHeavyMotor class:**



**CLightMotor class:**

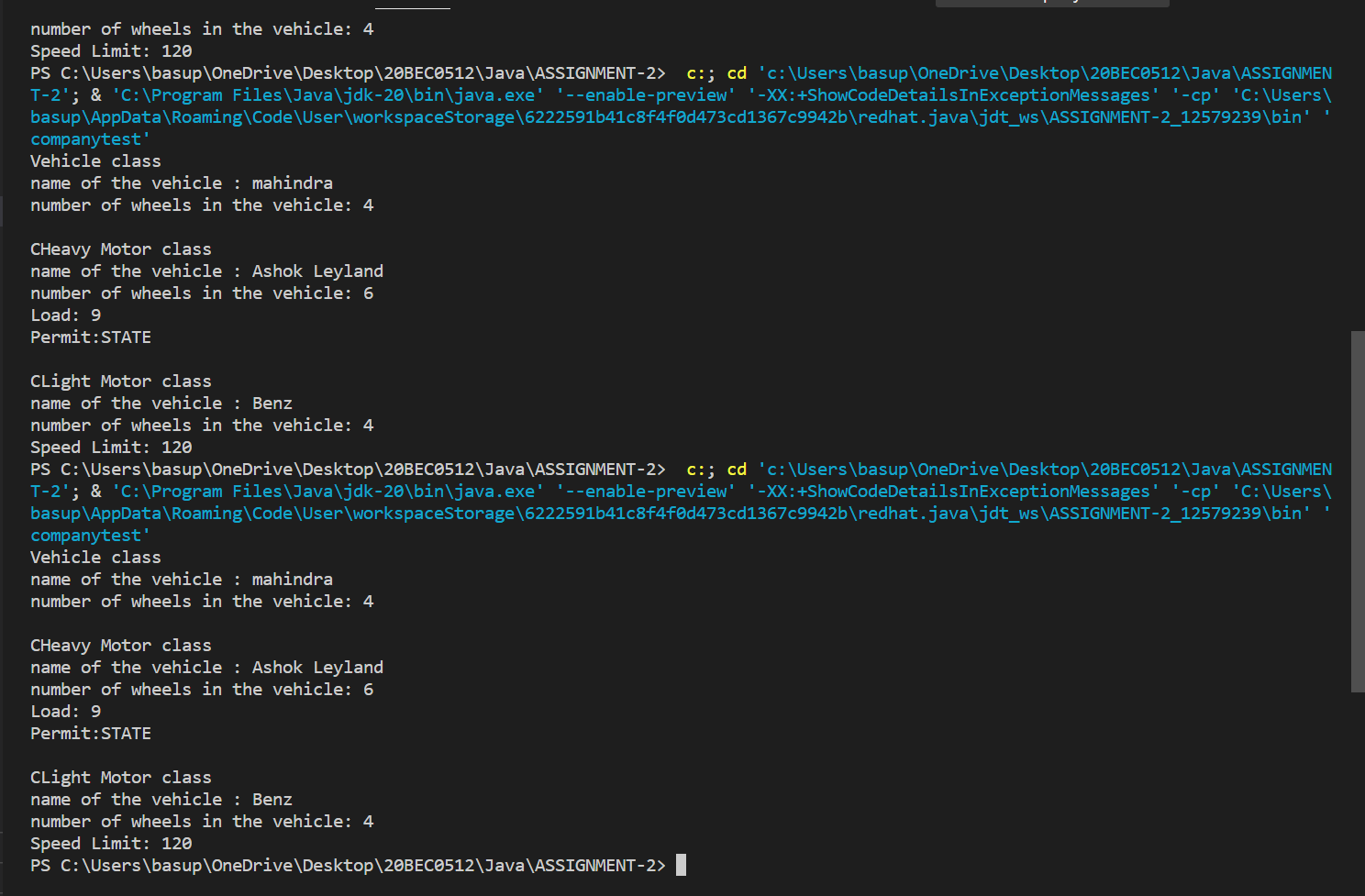
A screen shot of a computer program

Description automatically generated

**Test Application code:**



**Output Screenshot:**

****

1. **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.

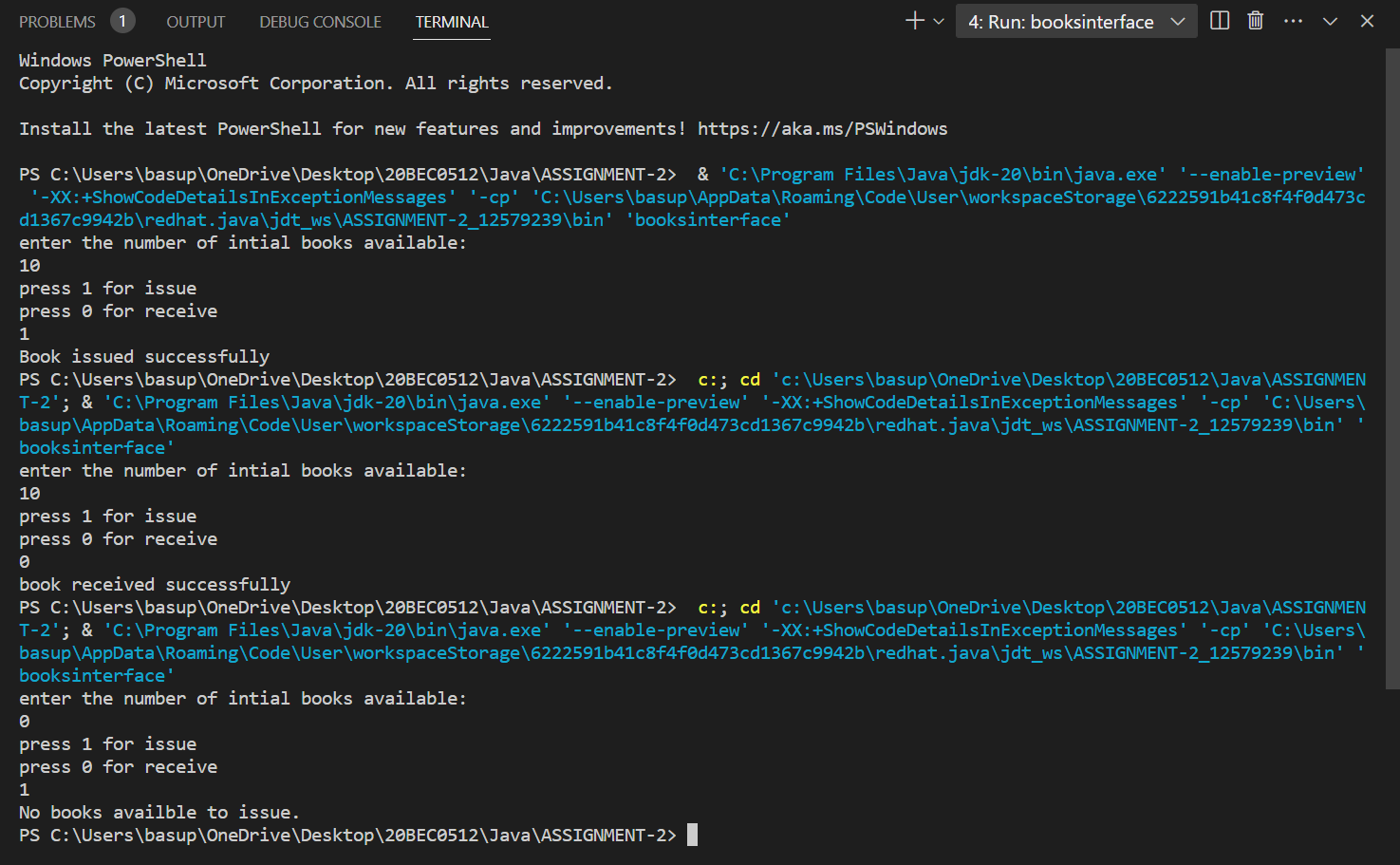
**Program code:**

|  |
| --- |
| 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:**



**Output Screenshot:**

****

1. **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:**

**Program code:**

**Code Screenshot:**

**Output Screenshot:**