**ID: hrajranj**

**Day 8 – 5th June 2025**

**Task 18:**

**Now create one more program named Task018.java**

public class Main {

public static void main(String[] args) {

Person myObj = new Person();

myObj.setName("John");

System.out.println(myObj.getName());

}

}

Now —--------------think what is the output of the above code—--------------

**Ans:**



//Attaching Multiple values

public enum Element {

    H("Hydrogen", 1, 1.008f),

    HE("Helium", 2, 4.0026f),

    // ...

    NE("Neon", 10, 20.180f);

    private static final Map<String, Element> BY\_LABEL = new HashMap<>();

    private static final Map<Integer, Element> BY\_ATOMIC\_NUMBER = new HashMap<>();

    private static final Map<Float, Element> BY\_ATOMIC\_WEIGHT = new HashMap<>();

    static {

        for (Element e : values()) {    //for each loop

            BY\_LABEL.put(e.label, e);

            BY\_ATOMIC\_NUMBER.put(e.atomicNumber, e);

            BY\_ATOMIC\_WEIGHT.put(e.atomicWeight, e);

        }

    }

    public final String label;

    public final int atomicNumber;

    public final float atomicWeight;

    private Element(String label, int atomicNumber, float atomicWeight) {

        this.label = label;

        this.atomicNumber = atomicNumber;

        this.atomicWeight = atomicWeight;

    }

    public static Element valueOfLabel(String label) {

        return BY\_LABEL.get(label);

    }

    public static Element valueOfAtomicNumber(int number) {

        return BY\_ATOMIC\_NUMBER.get(number);

    }

    public static Element valueOfAtomicWeight(float weight) {

        return BY\_ATOMIC\_WEIGHT.get(weight);

    }

}

**Task 19:**

**Wap to display the content of the above enum.. (main  needs to be added)**

**Ans:**

import java.util.HashMap;

import java.util.Map;

public enum Element {

    H("Hydrogen", 1, 1.008f),

    HE("Helium", 2, 4.0026f),

    LI("Lithium", 3, 6.94f),

    BE("Beryllium", 4, 9.0122f),

    B("Boron", 5, 10.81f),

    C("Carbon", 6, 12.011f),

    N("Nitrogen", 7, 14.007f),

    O("Oxygen", 8, 15.999f),

    F("Fluorine", 9, 18.998f),

    NE("Neon", 10, 20.180f);

    private static final Map<String, Element> BY\_LABEL = new HashMap<>();

    private static final Map<Integer, Element> BY\_ATOMIC\_NUMBER = new HashMap<>();

    private static final Map<Float, Element> BY\_ATOMIC\_WEIGHT = new HashMap<>();

    static {

        for (Element e : values()) {

            BY\_LABEL.put(e.label, e);

            BY\_ATOMIC\_NUMBER.put(e.atomicNumber, e);

            BY\_ATOMIC\_WEIGHT.put(e.atomicWeight, e);

        }

    }

    public final String label;

    public final int atomicNumber;

    public final float atomicWeight;

    private Element(String label, int atomicNumber, float atomicWeight) {

        this.label = label;

        this.atomicNumber = atomicNumber;

        this.atomicWeight = atomicWeight;

    }

    public static Element valueOfLabel(String label) {

        return BY\_LABEL.get(label);

    }

    public static Element valueOfAtomicNumber(int number) {

        return BY\_ATOMIC\_NUMBER.get(number);

    }

    public static Element valueOfAtomicWeight(float weight) {

        return BY\_ATOMIC\_WEIGHT.get(weight);

    }

    // ✅ Main method to display all enum values

    public static void main(String[] args) {

        System.out.println("All Elements:");

        for (Element e : Element.values()) {

            System.out.println("Symbol: " + e.name() +

                               ", Label: " + e.label +

                               ", Atomic Number: " + e.atomicNumber +

                               ", Atomic Weight: " + e.atomicWeight);

        }

        // 🔍 Test specific lookups

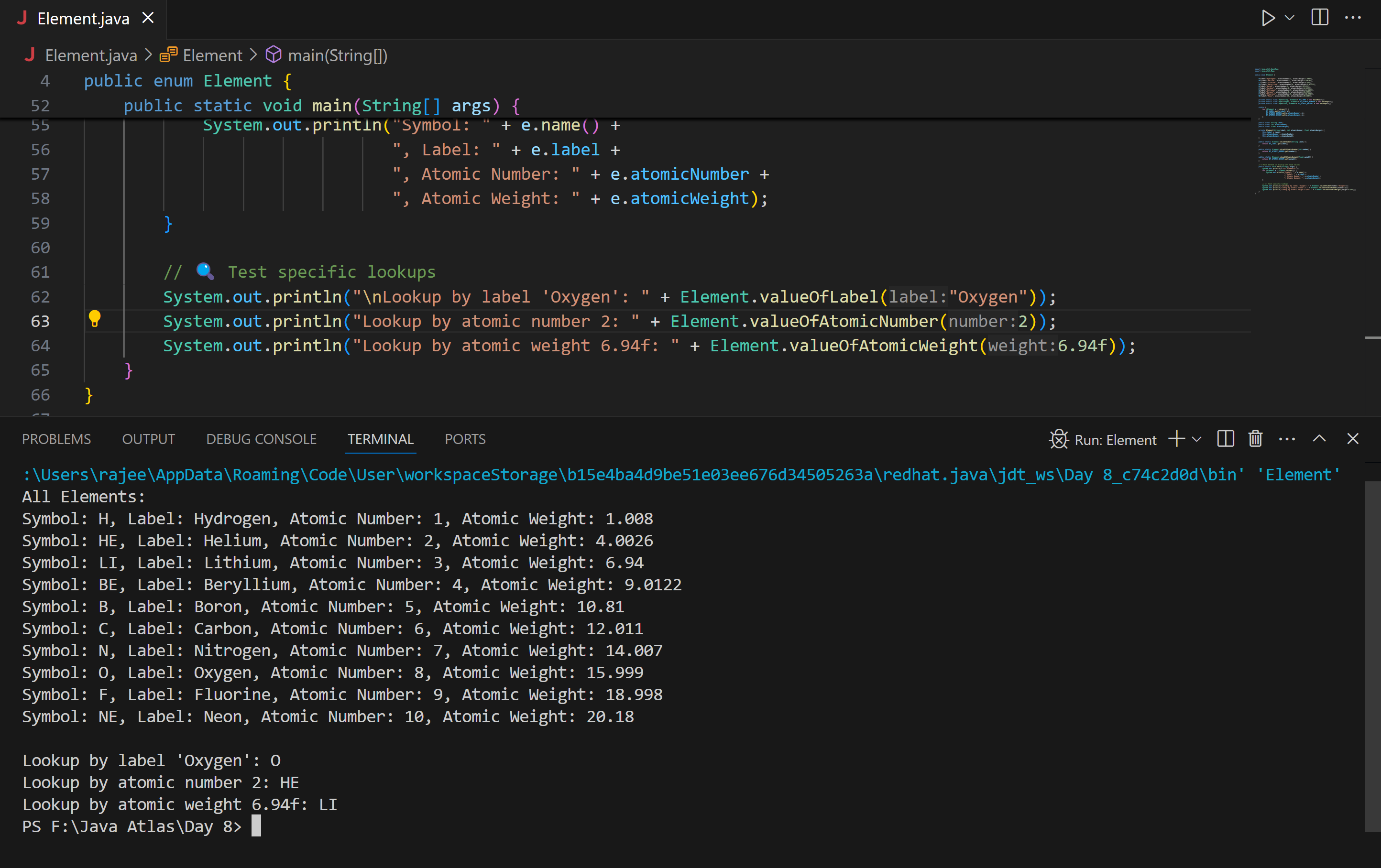
        System.out.println("\nLookup by label 'Oxygen': " + Element.valueOfLabel("Oxygen"));

        System.out.println("Lookup by atomic number 2: " + Element.valueOfAtomicNumber(2));

        System.out.println("Lookup by atomic weight 6.94f: " + Element.valueOfAtomicWeight(6.94f));

    }

}



**Task 020:**

**Create an array of your name**

Hint : use

Char[] Name = {‘P’, “r’, ….}; // initializing an array

sout(Name);

Int n = Name.length; // size of your name

sout(“there are “+ n +”letters in my name”);

Use for loop to display each letter..

HInt: use ghe below code snippet…

// traversing array

        for (int i = 0; i < n; i++)

            System.out.print(Name[i] + " ");

**Ans:**

public class Task20 {

    public static void main(String[] args) {

        // Initialize name as a character array

        char[] Name = {'R',  'a', 'j', 'e', 'e', 'v', };

        // Print full array as a string

        System.out.println(Name);  // prints the full name

        // Get length of the array

        int n = Name.length;

        // Print number of letters

        System.out.println("There are " + n + " letters in my name.");

        // Traverse and print each letter

        System.out.print("Letters in my name: ");

        for (int i = 0; i < n; i++) {

            System.out.print(Name[i] + " ");

        }

    }

}



Task 21: Home Task

**Example:** This example demonstrates how to initialize an array and traverse it using a for loop to print each element.

public class Main {

   public static void main(String[] args)

   {

​

       // initializing array

       int[] arr = { 1, 2, 3, 4, 5 };

​

       // size of array

       int n = arr.length;

​

       // traversing array

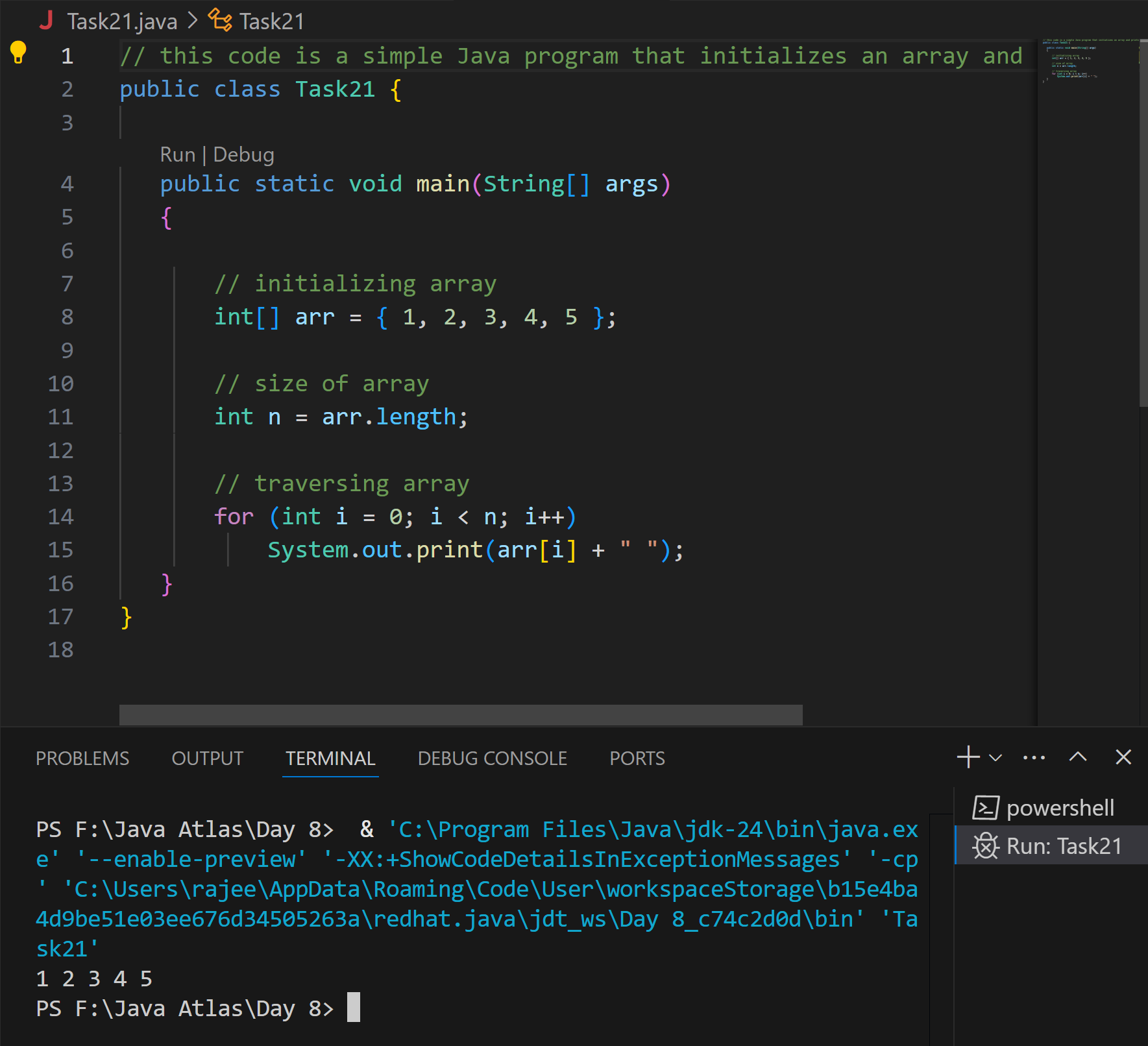
       for (int i = 0; i < n; i++)

           System.out.print(arr[i] + " ");

   }

}

**Ans:**



**Task 22 - home task**

**Implementation:**

// Java program to illustrate creating an array

// of integers,  puts some values in the array,

// and prints each value to standard output.

​

class GFG {

   public static void main(String[] args)

   {

       // declares an Array of integers.

       int[] arr;

​

       // allocating memory for 5 integers.

       arr = new int[5];

​

       // initialize the elements of the array

       // first to last(fifth) element

      arr[0] = 10;

      arr[1] = 20;

       arr[2] = 30;

       arr[3] = 40;

       arr[4] = 50;

​       // accessing the elements of the specified array

       for (int i = 0; i < arr.length; i++)

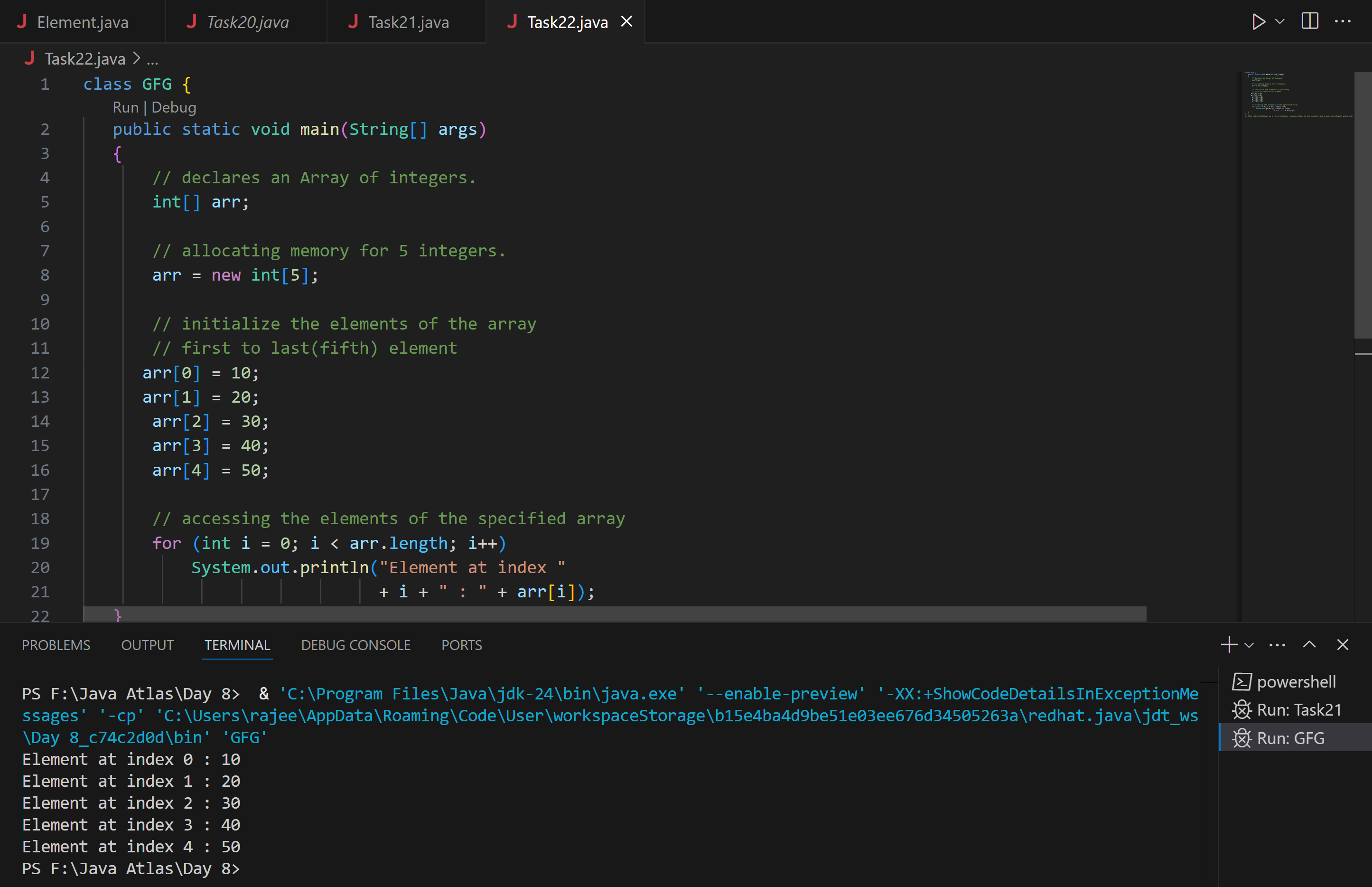
           System.out.println("Element at index "

                              + i + " : " + arr[i]);

   }

}

**Ans:**



**Task 23 - home task**

**Example:** Here we are taking a student class and creating an array of Student with five Student objects stored in the array. The Student objects have to be instantiated using the constructor of the Student class, and their references should be assigned to the array elements.

**Ans:​**

class Student {

public int roll\_no;

public String name;

Student(int roll\_no, String name){

this.roll\_no = roll\_no;

this.name = name;

}

}

public class Task23 {

public static void main(String[] args){

// declares an Array of Student

Student[] arr;

// allocating memory for 5 objects of type Student.

arr = new Student[5];

// initialize the elements of the array

arr[0] = new Student(1, "aman");

arr[1] = new Student(2, "vaibhav");

arr[2] = new Student(3, "shikar");

arr[3] = new Student(4, "dharmesh");

arr[4] = new Student(5, "mohit");

// accessing the elements of the specified array

for (int i = 0; i < arr.length; i++)

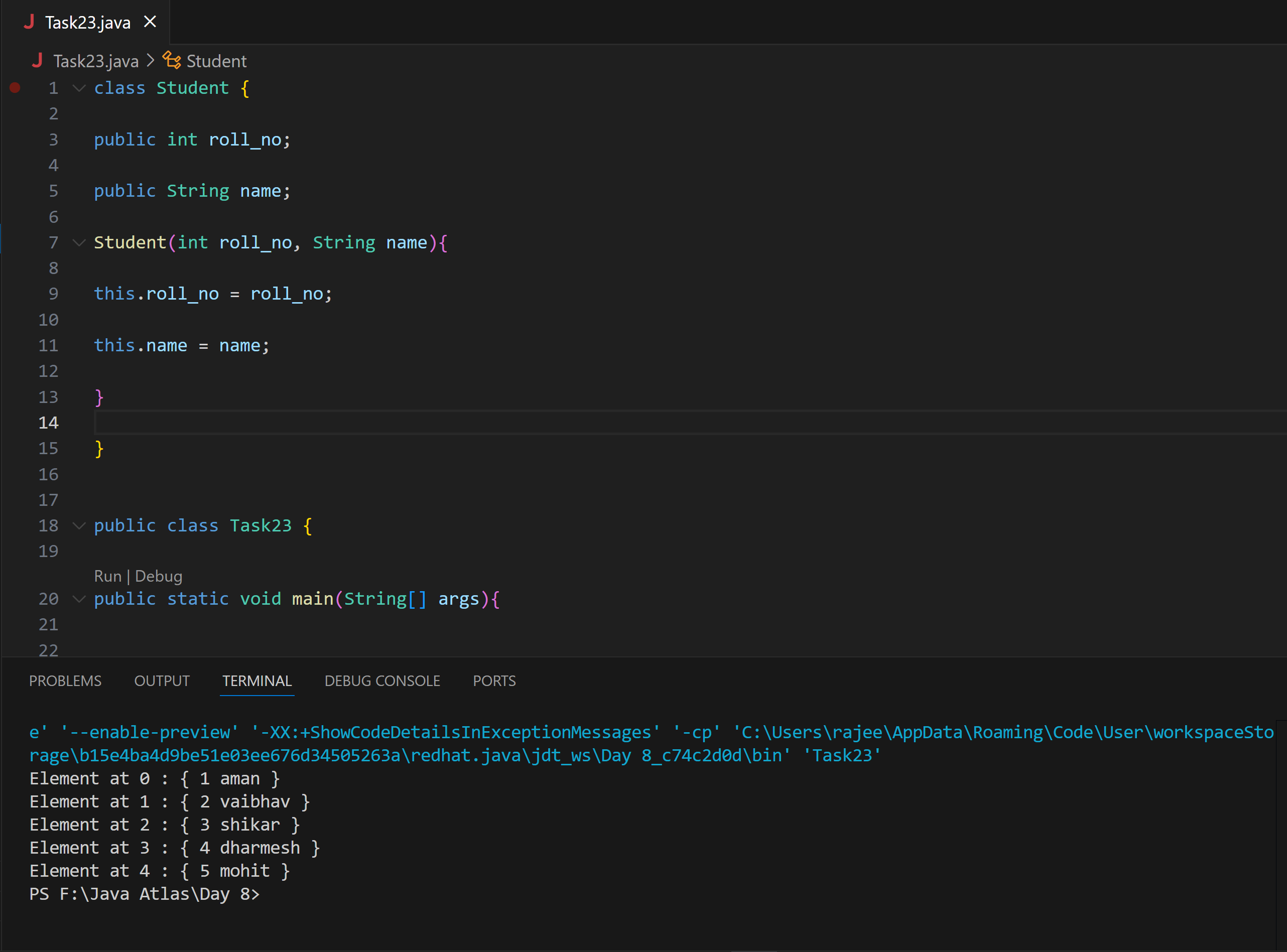
System.out.println("Element at " + i + " : { "

+ arr[i].roll\_no + " "

+ arr[i].name+" }");

}

}



**Task 024 Home task**

**Example: An array of objects is also created like**

**Ans:**

// Task 24: Create a class named Student with a field for the student's name.

class Student{

public String name;

Student(String name){

this.name = name;

}

@Override

public String toString(){

return name;

}

}

public class Task24{

public static void main (String[] args){

// declares an Array and initializing the

// elements of the array

Student[] myStudents = new Student[]{

new Student("Dharma"),new Student("sanvi"),

new Student("Rupa"),new Student("Ajay")

};

// accessing the elements of the specified array

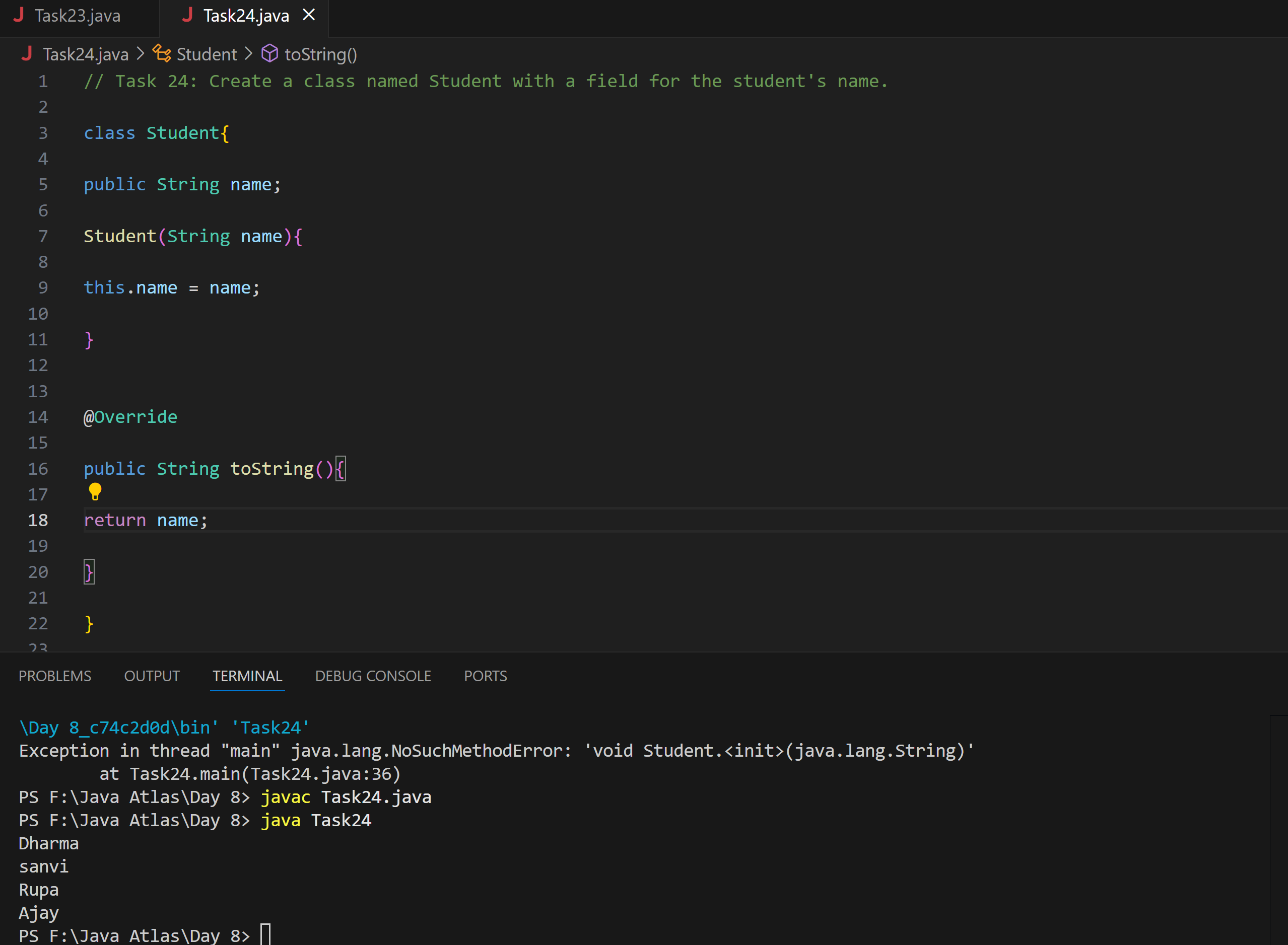
for(Student m:myStudents){

System.out.println(m);

}

}

}



**Task 025 - home task**

**Example: Let us start with basic two dimensional Array declared and initialized.**

**Ans:**

import java.io.\*;

class GFG {

    public static void main(String[] args) {

        // Declaring and initializing a 3x3 2D array

        int[][] arr = new int[3][3];

        // Getting number of rows

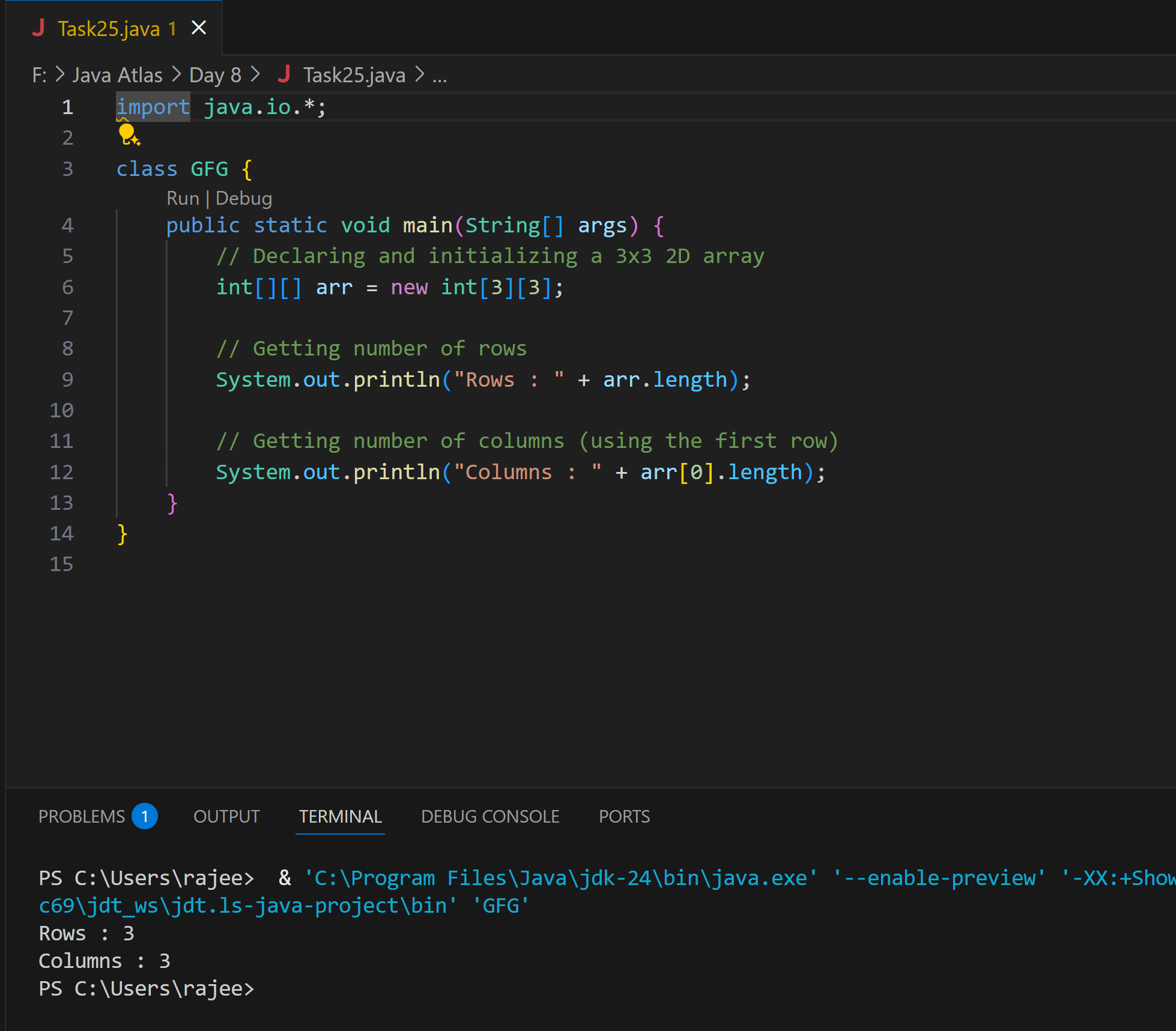
        System.out.println("Rows : " + arr.length);

        // Getting number of columns (using the first row)

        System.out.println("Columns : " + arr[0].length);

    }

}



**Task 026 - Home Task**

**Example: Now, after declaring and initializing the array we will check how to Traverse the Multidimensional Array using for loop.**

**Ans:**

// Java Program to Multidimensional Array

// Driver Class

public class Task26 {

    // main function

    public static void main(String args[]) {

        // declaring and initializing 2D array

        int arr[][] = { { 2, 7, 9 }, { 3, 6, 1 }, { 7, 4, 2 } };

        // printing 2D array

        for (int i = 0; i < 3; i++) {

            for (int j = 0; j < 3; j++)

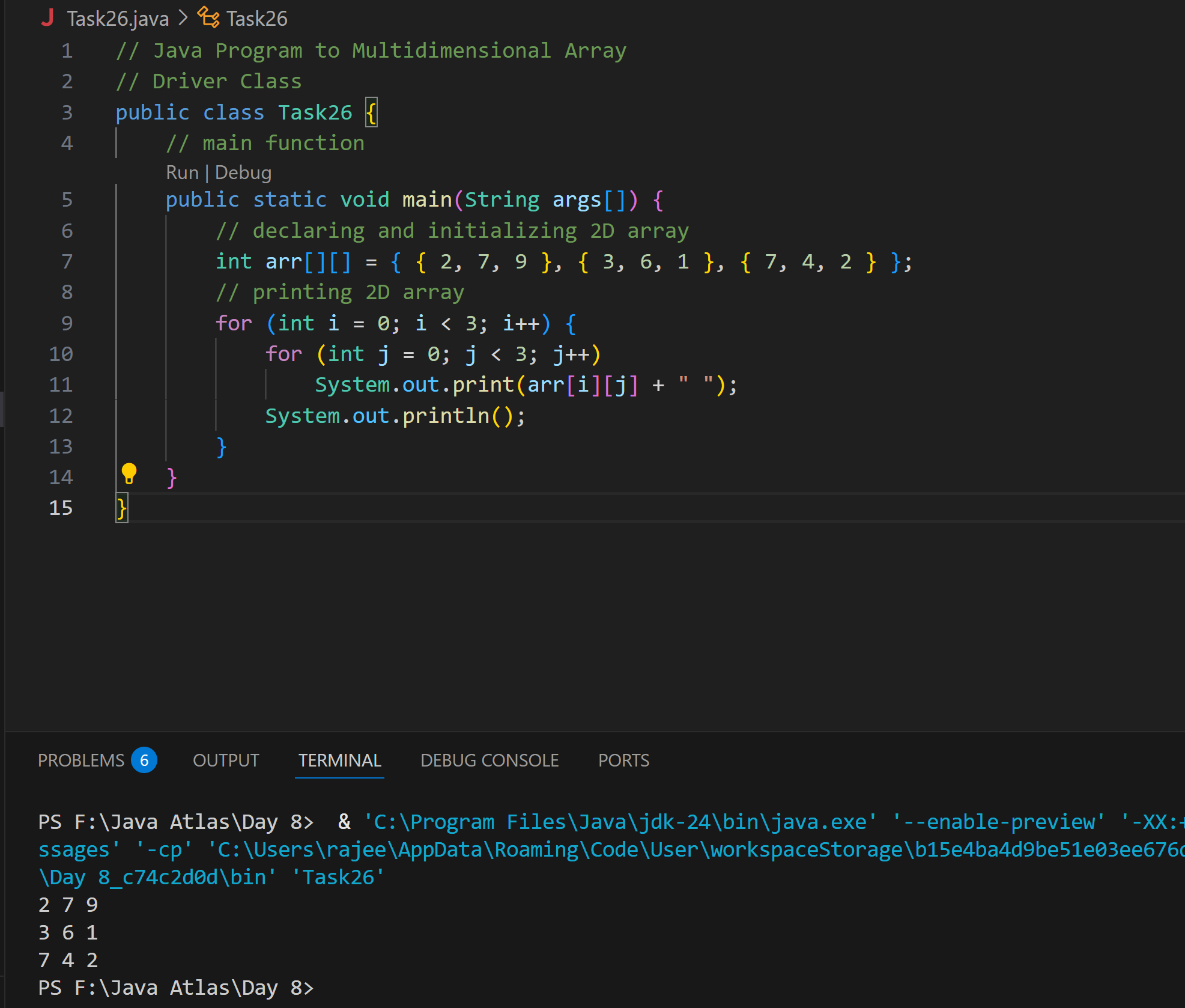
                System.out.print(arr[i][j] + " ");

            System.out.println();

        }

    }

}



**Task 27 - Home task**

**// Java program to demonstrate passing of array to method**

**Ans:**

public class Task27 {

// Driver method

public static void main(String args[])

{

int arr[] = { 3, 1, 2, 5, 4 };

// passing array to method m1

sum(arr);

}

public static void sum(int[] arr)

{

// getting sum of array values

int sum = 0;

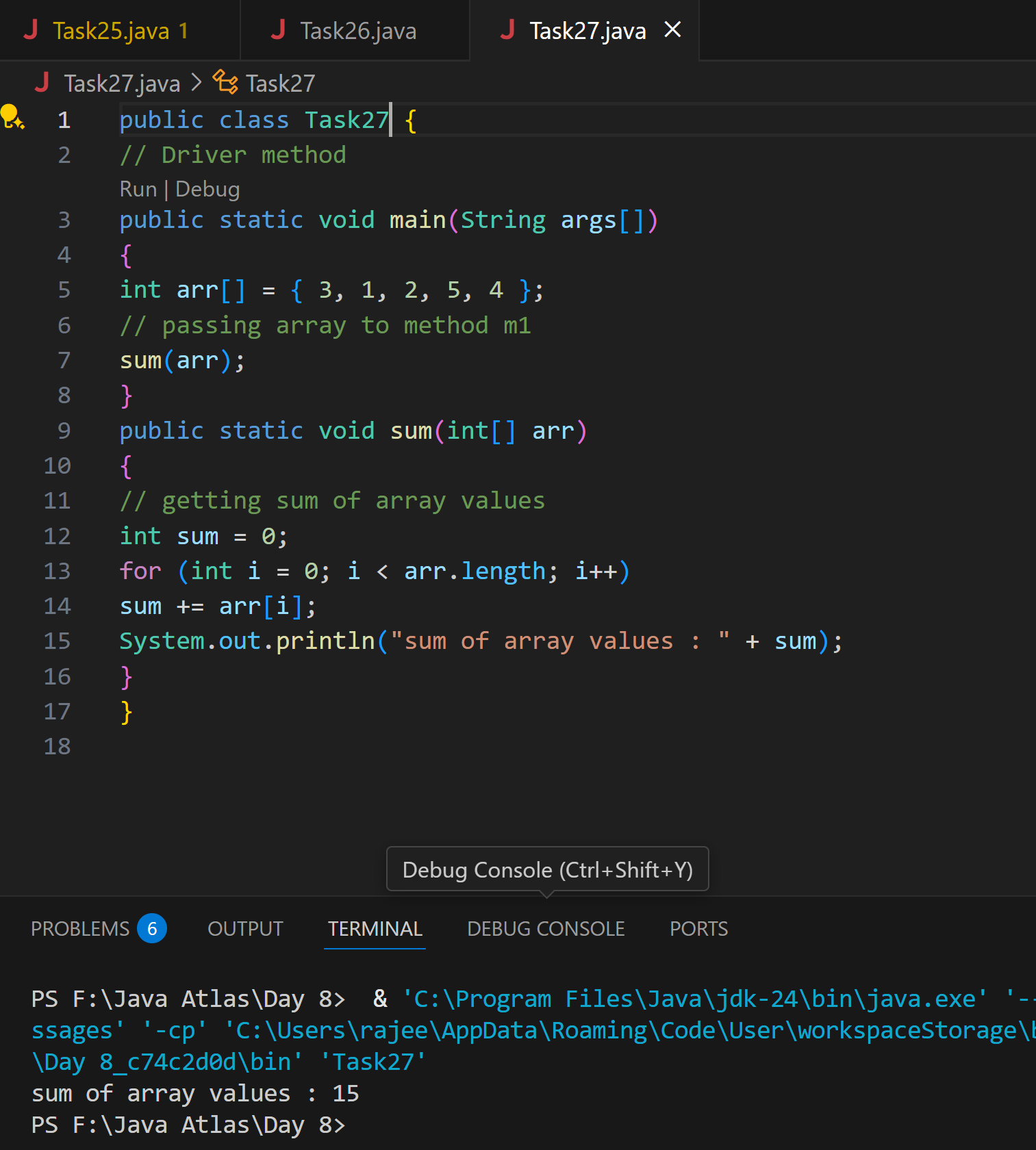
for (int i = 0; i < arr.length; i++)

sum += arr[i];

System.out.println("sum of array values : " + sum);

}

}



**Task 28 - Home Task**

**// Java program to demonstrate // return of array from method**

**Ans:**

class Test {

    // Driver method

    public static void main(String args[]) {

        int arr[] = m1();

        for (int i = 0; i < arr.length; i++)

            System.out.print(arr[i] + " ");

    }

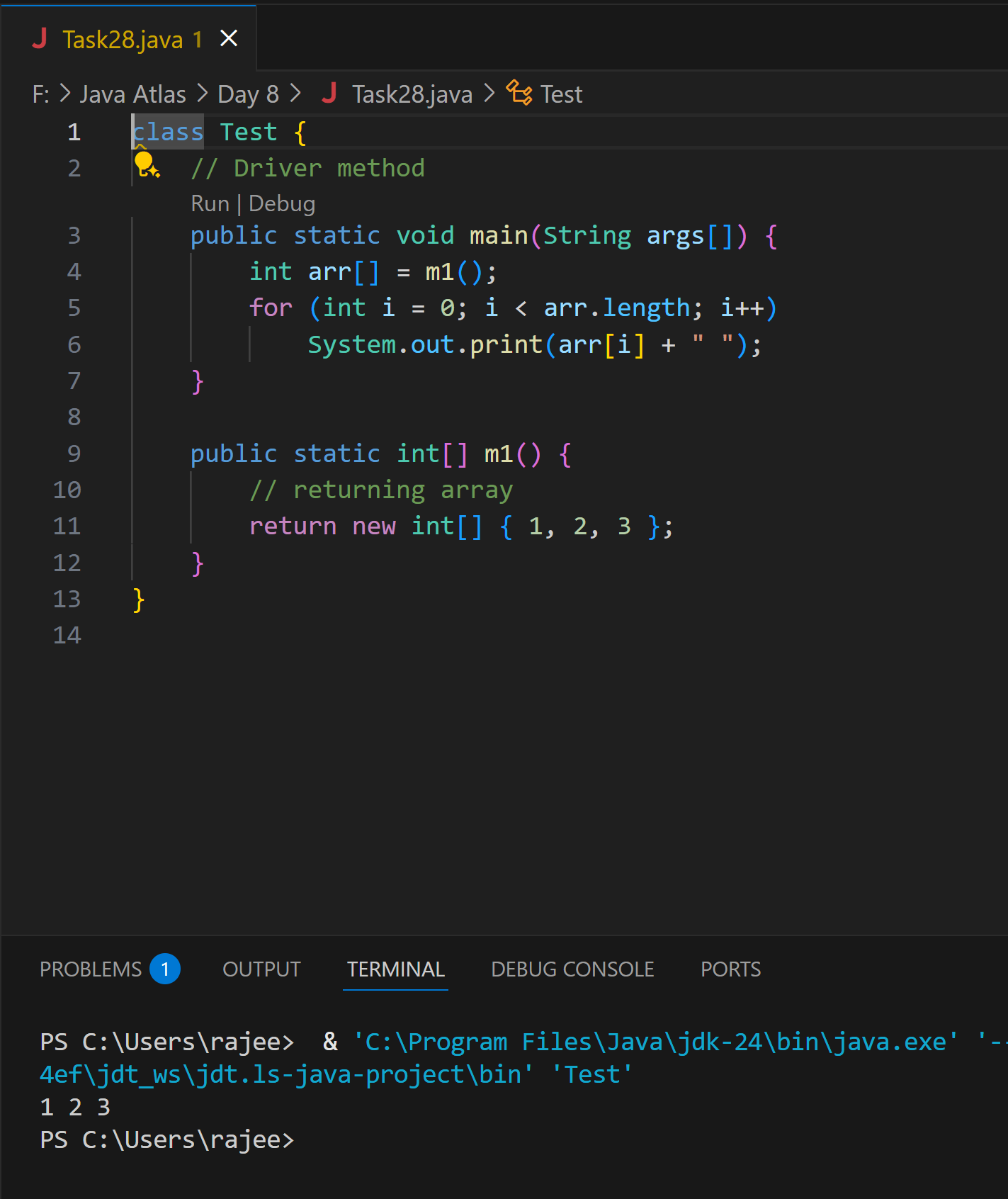
    public static int[] m1() {

        // returning array

        return new int[] { 1, 2, 3 };

    }

}



**Task 29 home Task**

**// Java program to demonstrate // cloning of one-dimensional arrays**

Ans:

class Test {

    public static void main(String args[]) {

        int intArray[] = { 1, 2, 3 };

        int cloneArray[] = intArray.clone();

        // Will print false as a shallow copy is created

        System.out.println(intArray == cloneArray);

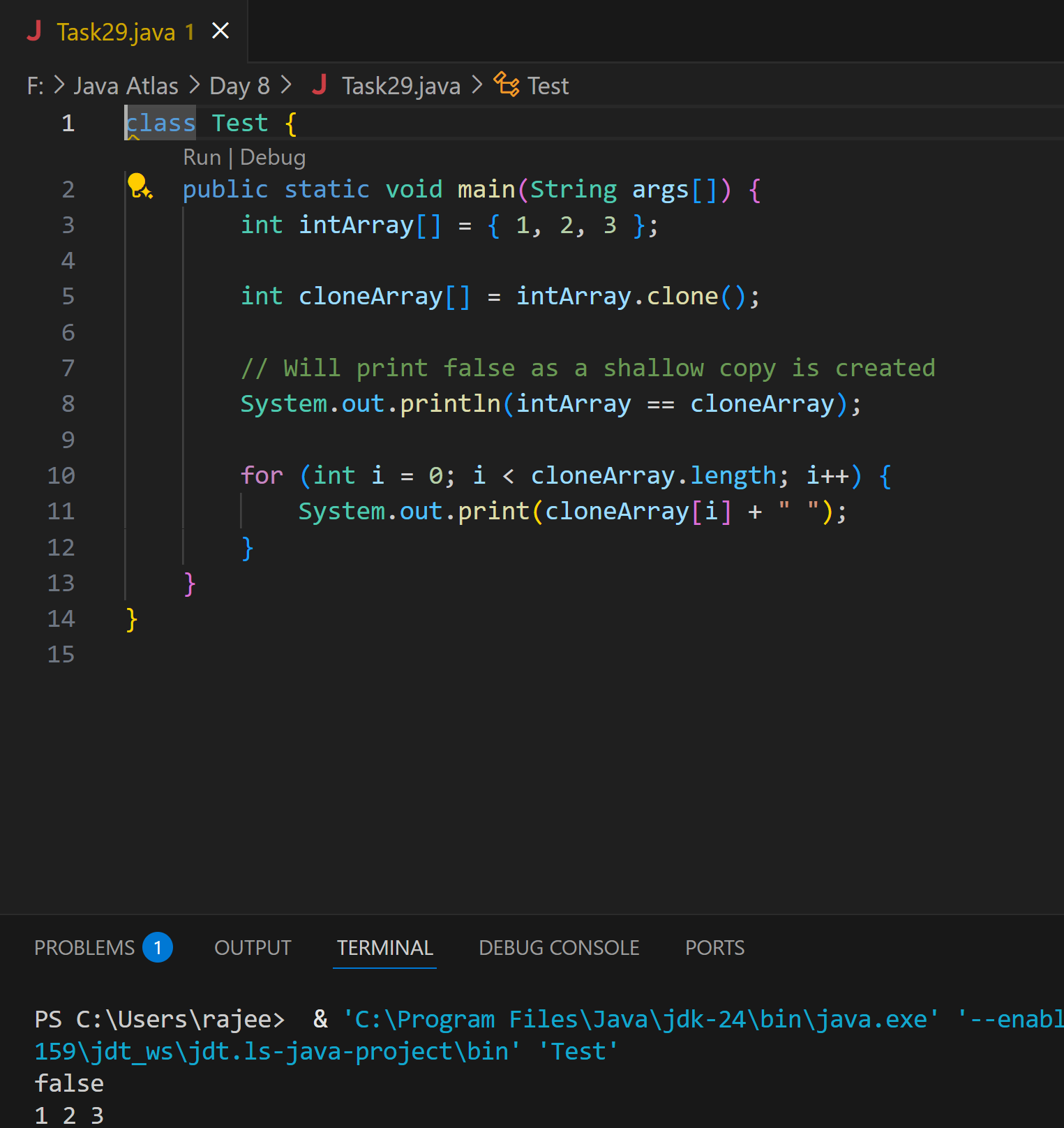
        for (int i = 0; i < cloneArray.length; i++) {

            System.out.print(cloneArray[i] + " ");

        }

    }

}



**Task 30 Home Task**

**// Java program to demonstrate**

**// cloning of multi-dimensional arrays**

**Ans:**

class Test {

    public static void main(String args[]) {

        int intArray[][] = { { 1, 2, 3 }, { 4, 5 } };

        int cloneArray[][] = intArray.clone();

        // will print false

        System.out.println(intArray == cloneArray);

        // will print true as shallow copy is created

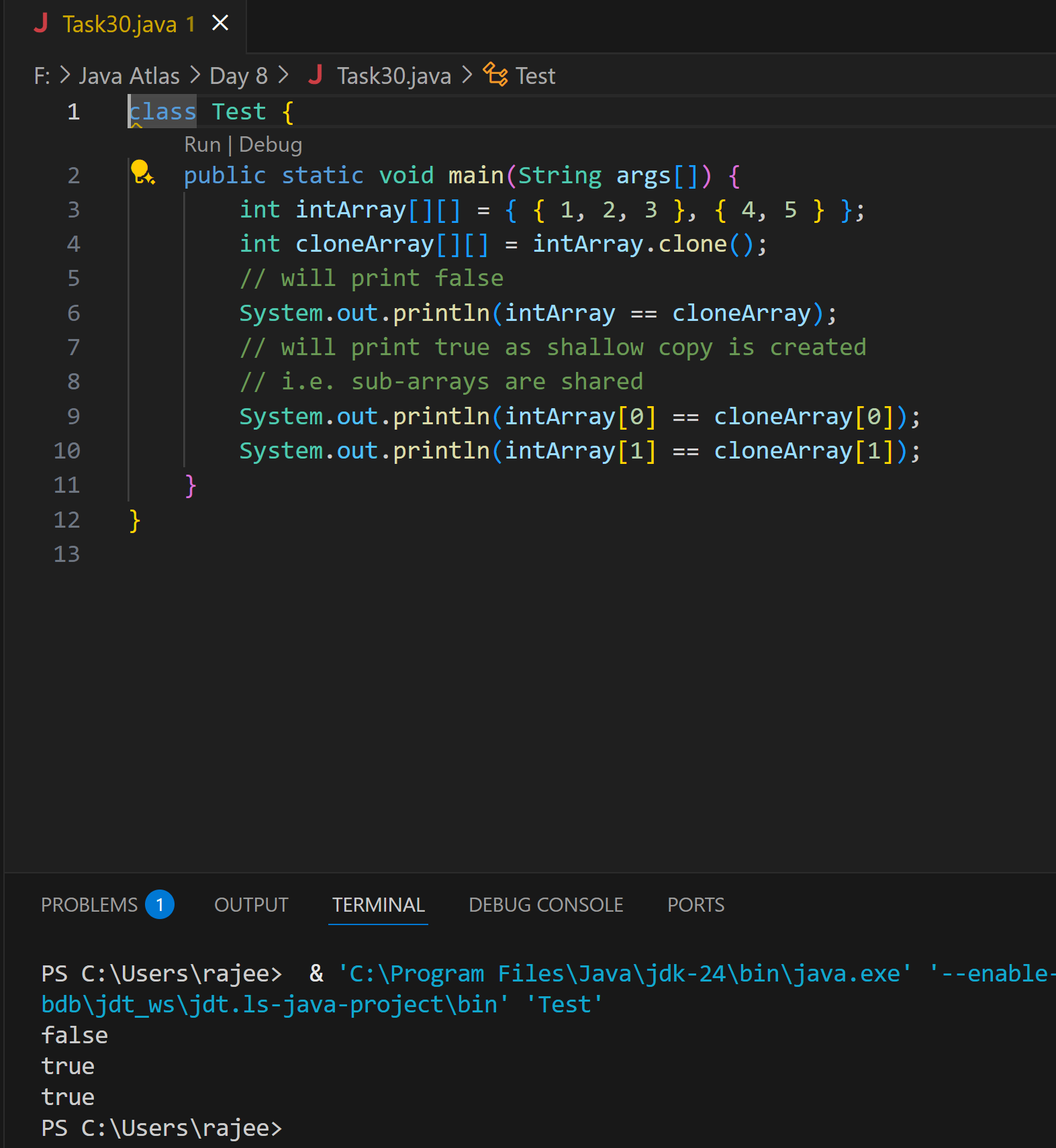
        // i.e. sub-arrays are shared

        System.out.println(intArray[0] == cloneArray[0]);

        System.out.println(intArray[1] == cloneArray[1]);

    }

}



**Task 31**

class Calculation {

int z;

public void addition(int x, int y) {

z = x + y;

System.out.println("The sum of the given numbers:"+z);

}

public void Subtraction(int x, int y) {

z = x - y;

System.out.println("The difference between the given numbers:"+z);

}

}

public class My\_Calculation extends Calculation {

public void multiplication(int x, int y) {

z = x \* y;

System.out.println("The product of the given numbers:"+z);

}

public static void main(String args[]) {

int a = 20, b = 10;

My\_Calculation demo = new My\_Calculation();

demo.addition(a, b);

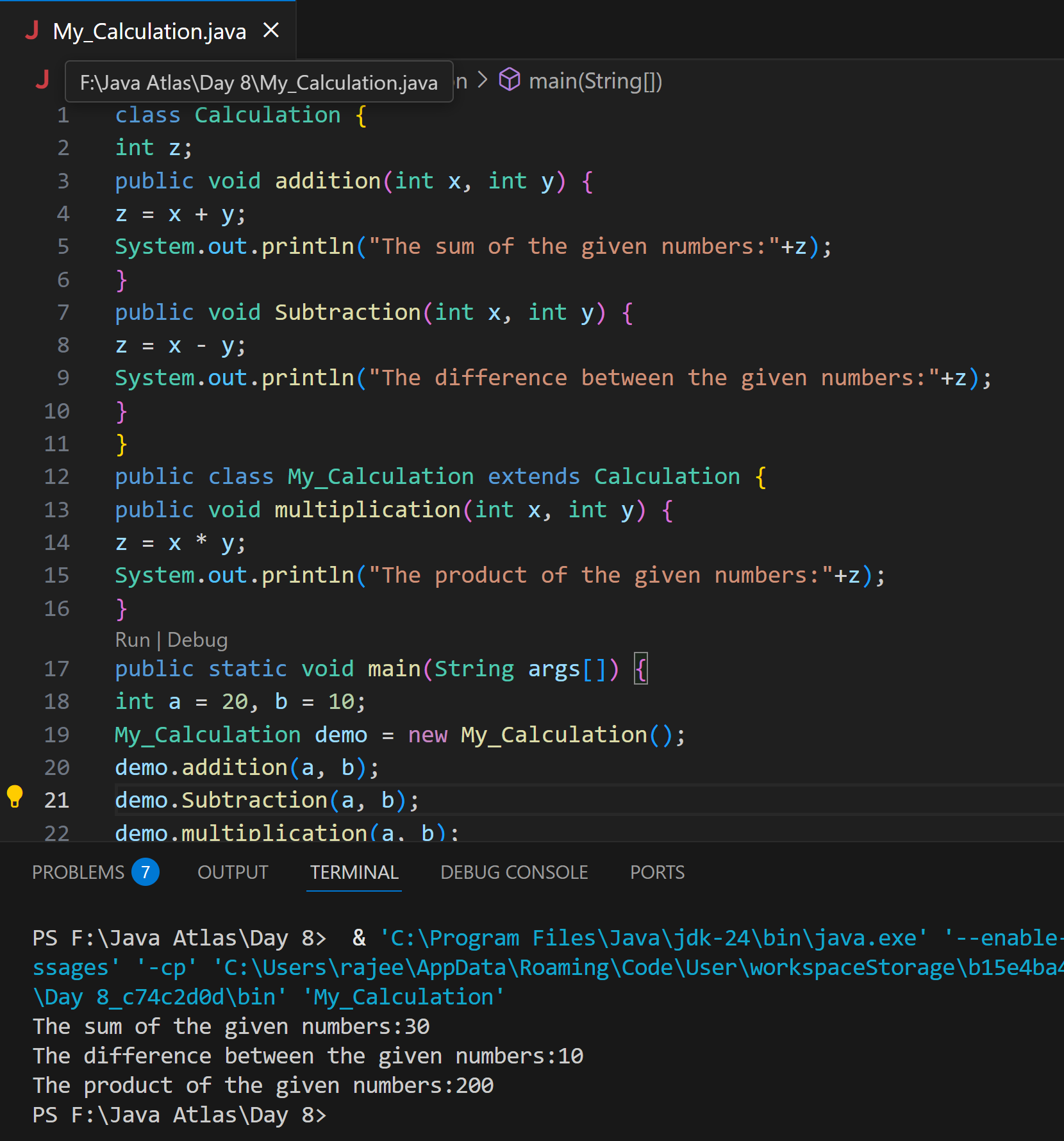
demo.Subtraction(a, b);

demo.multiplication(a, b);

}

}

**Ans:**



**Task 32**

In the above code add a class clock — and try to extend calculation and clock in the my calculation class.. Is it possible ???? give reason.

class clock {

—--

—--

}

class my\_calculation extends calculation , clock{ // multiple inheritance

// —---------------------------------- ???????????????????????

**Ans:**

No, we cannot do this. Java does not support multiple inheritance through classes. In java, a class cannot extend more than one class. However, we can do this by interfaces.

**Task 33**

class Customer {

Void purchage\_list{

Int cos = 40t;

String items = “Tomatoes”;

}

}

public class Mart extends Customer {

Void billing(){

String items = “onions”;

Int cost = 30;

}

Psvm (String[] args) {

Super.items = “Potatoes”

Super.cost = 50;

Sout(items);

sout(cost);

sout”(%%%%%%%%%%%%%%”);

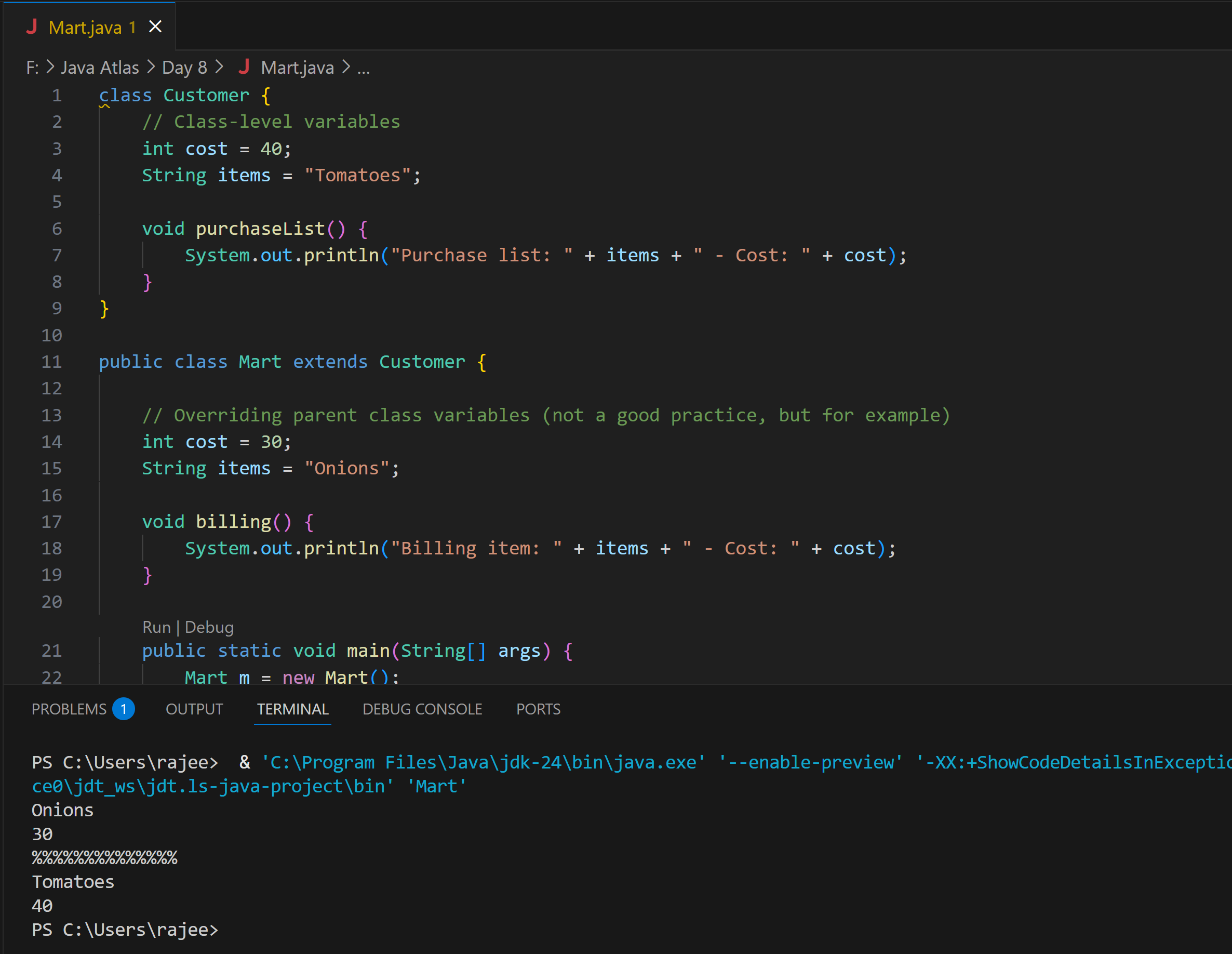
Sout(super.items);

sout(suer.cost);

}

}

**Ans:**



**Task 34**

Void add(int x, int y){

Sout —> x and y values

}

Void add(int x, int y, int z){

Sout —-> x, y, z values

}

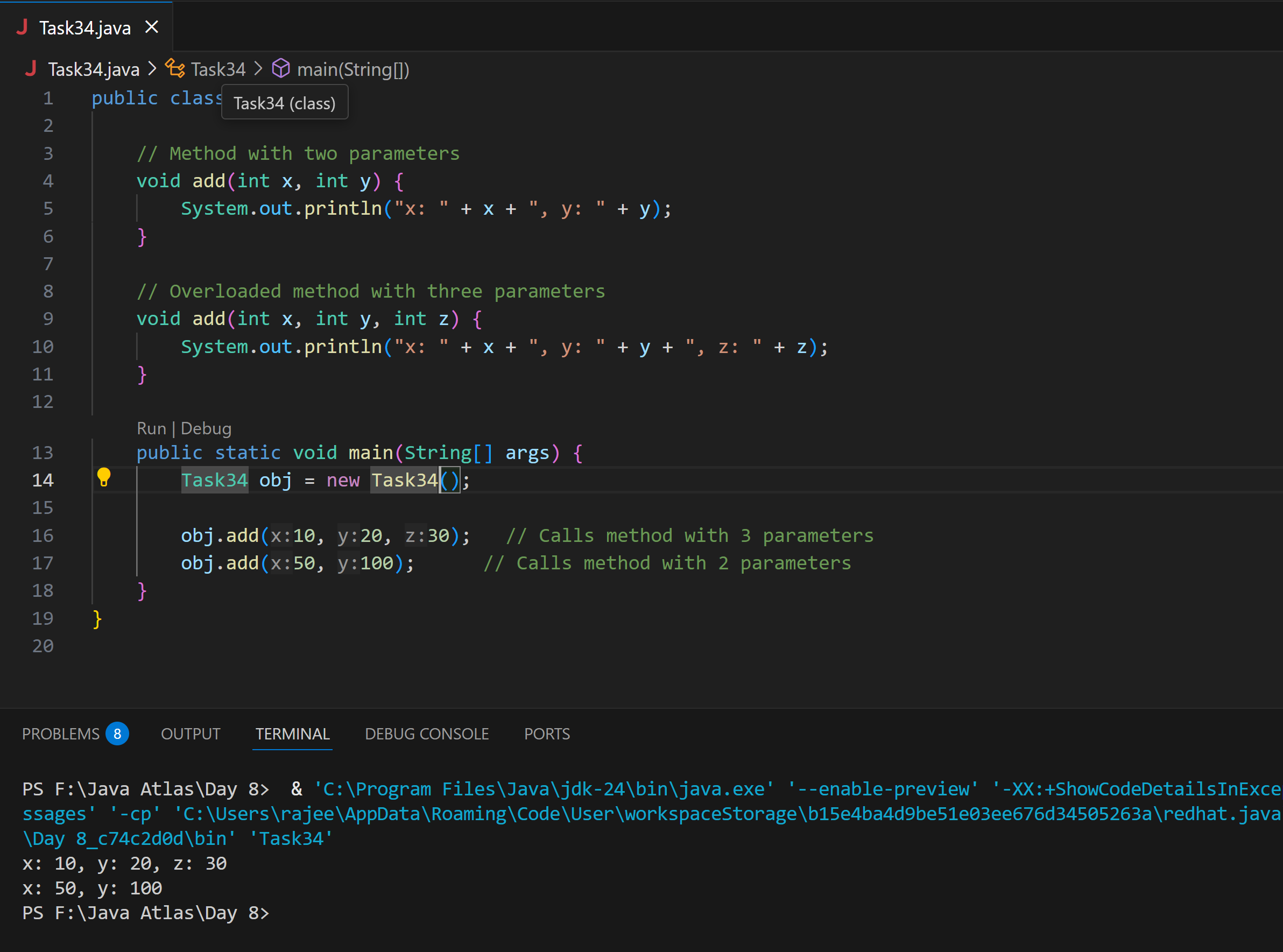
psvm(){

add(10,20,30);

add(50,100);

}

**Ans:**



**Task 35**

Void add(char x, char y){

Sout —-> x, y values

}

Void add(int x, int y) {

Sout —> x, y values

}

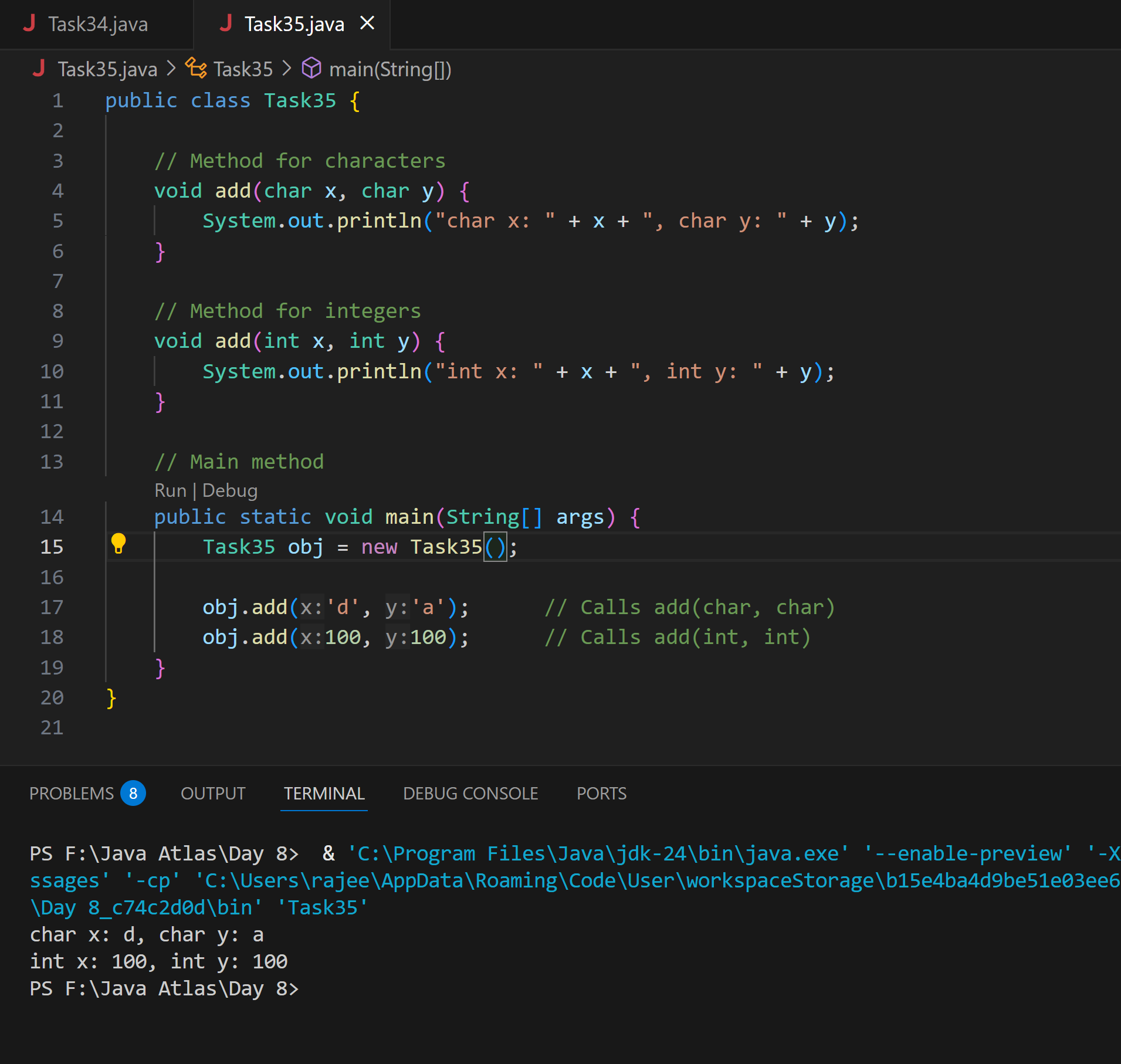
psvm(){

add(‘d’, ‘a’);

add(100, 100);

}

**Ans:**



**Task 36**

Void add(int x, float y){

Sout → x, y values

}

Void add(float x, int y){

Sout → x, y

}

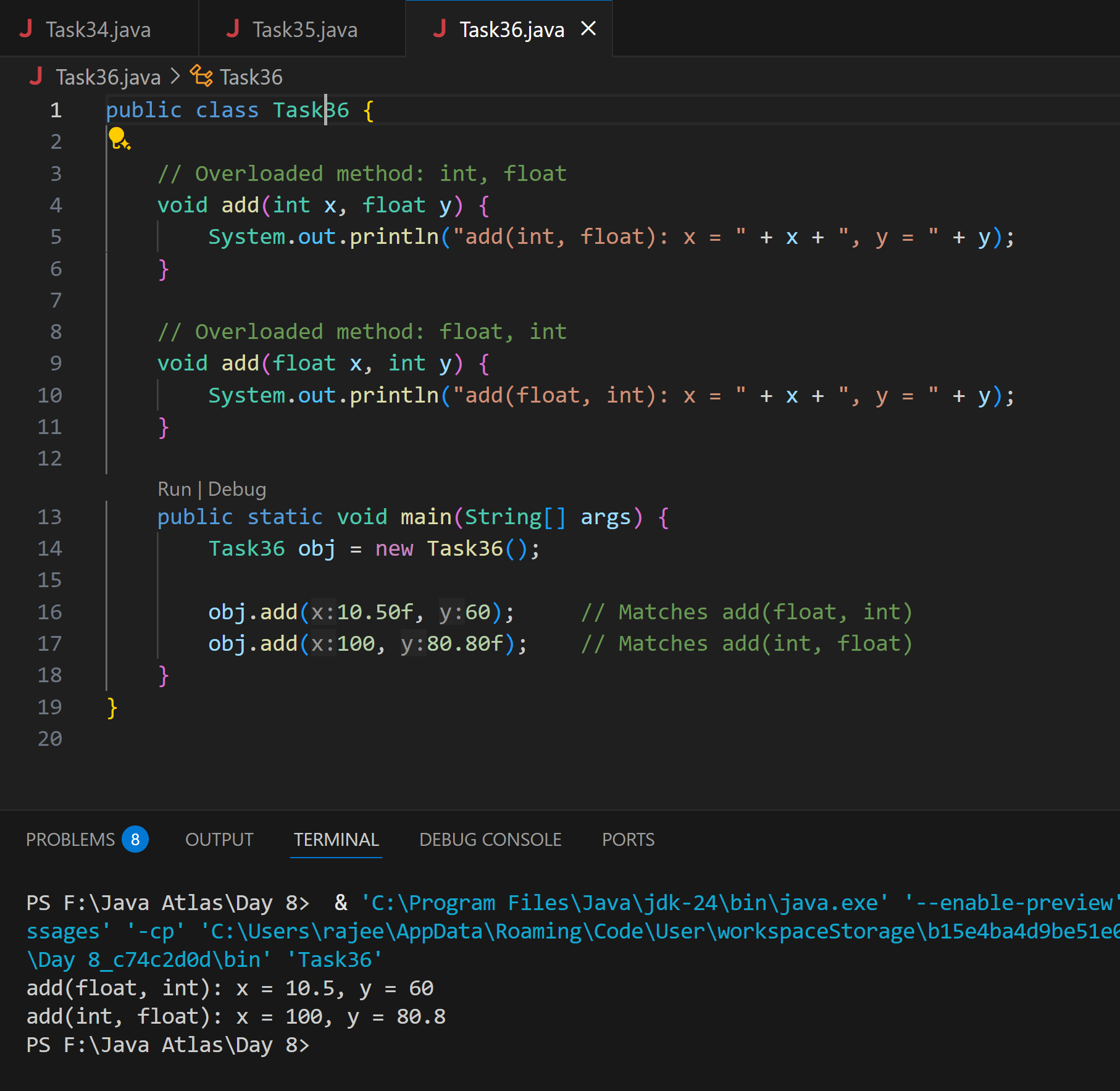
psvm(){

add(10.50f, 60);

add(100, 80.80f)

}

**Ans:**



**Task 37:**

Class Employee{

Private int pwd;

Protected int Salary;

Public int empid:

employee(){ // constructors are methods having same name as class name (we have in c++)

}

~employee(){// destructors used in c++ but not in java

}

}

Class Hr extends Employee {

super.pwd = 1254; //===============> ??????

super.Salary = 50000; //==================> ?

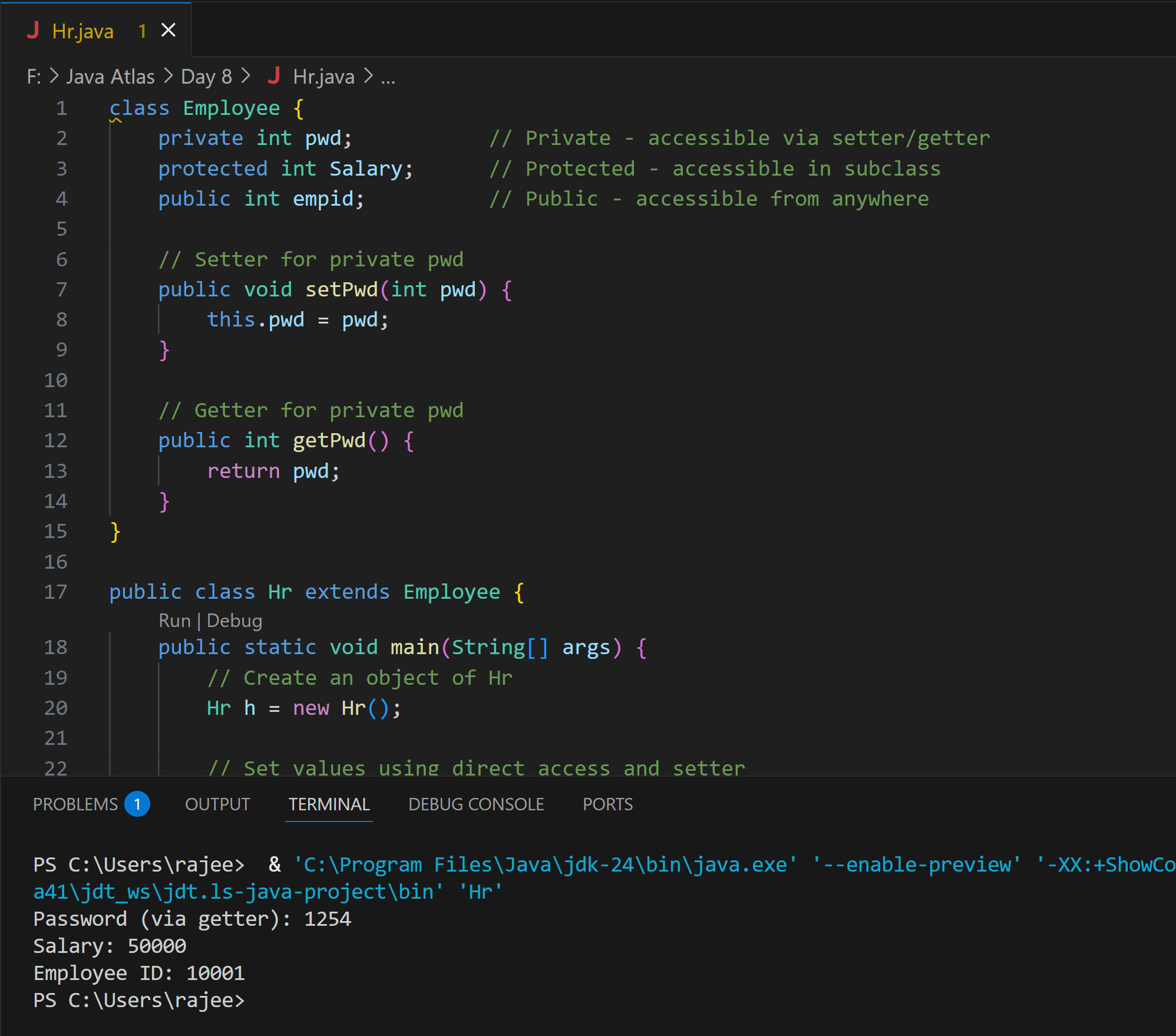
Super.empid = 10001; // ======================>?

psvm(){

}

}

**Ans:**



**Task 38**

/\* File name : AbstractDemo.java \*/

Public class AbstractDemo {

public static void main(String [] args) {

/\* Following is not allowed and would raise error \*/

Employee e = new Employee("George W.", "Houston, TX", 43);

System.out.println("\n Call mailCheck using Employee reference--");

e.mailCheck();

}

}

abstract class Employee {

private String name;

private String address;

private int number;

public Employee(String name, String address, int number) {

System.out.println("Constructing an Employee");

this.name = name;

this.address = address;

this.number = number;

}

public double computePay() {

System.out.println("Inside Employee computePay");

return 0.0;

}

public void mailCheck() {

System.out.println("Mailing a check to " + this.name + " " + this.address);

}

public String toString() {

return name + " " + address + " " + number;

}

public String getName() {

return name;

}

public String getAddress() {

return address;

}

public void setAddress(String newAddress) {

address = newAddress;

}

public int getNumber() {

return number;

}

}

**Ans:** It throws an error because abstract class cannot be instantiated directly in Java. Rather we can create a subclass and then extend it.

**Task 39**

Rewrite the above code to give the output without errors..

**Ans:**

abstract class Employee {

    private String name;

    private String address;

    private int number;

    public Employee(String name, String address, int number) {

        System.out.println("Constructing an Employee");

        this.name = name;

        this.address = address;

        this.number = number;

    }

    public double computePay() {

        System.out.println("Inside Employee computePay");

        return 0.0;

    }

    public void mailCheck() {

        System.out.println("Mailing a check to " + this.name + " " + this.address);

    }

    public String toString() {

        return name + " " + address + " " + number;

    }

    public String getName() {

        return name;

    }

    public String getAddress() {

        return address;

    }

    public void setAddress(String newAddress) {

        address = newAddress;

    }

    public int getNumber() {

        return number;

    }

}

// A concrete subclass of Employee

class SalariedEmployee extends Employee {

    private double salary; // Annual salary

    public SalariedEmployee(String name, String address, int number, double salary) {

        super(name, address, number);

        this.salary = salary;

    }

    @Override

    public double computePay() {

        System.out.println("Computing salary pay for " + getName());

        return salary / 52;

    }

    @Override

    public void mailCheck() {

        System.out.println("Within mailCheck of SalariedEmployee class");

        System.out.println("Mailing check to " + getName() + " with salary " + salary);

    }

}

public class AbstractDemo {

    public static void main(String[] args) {

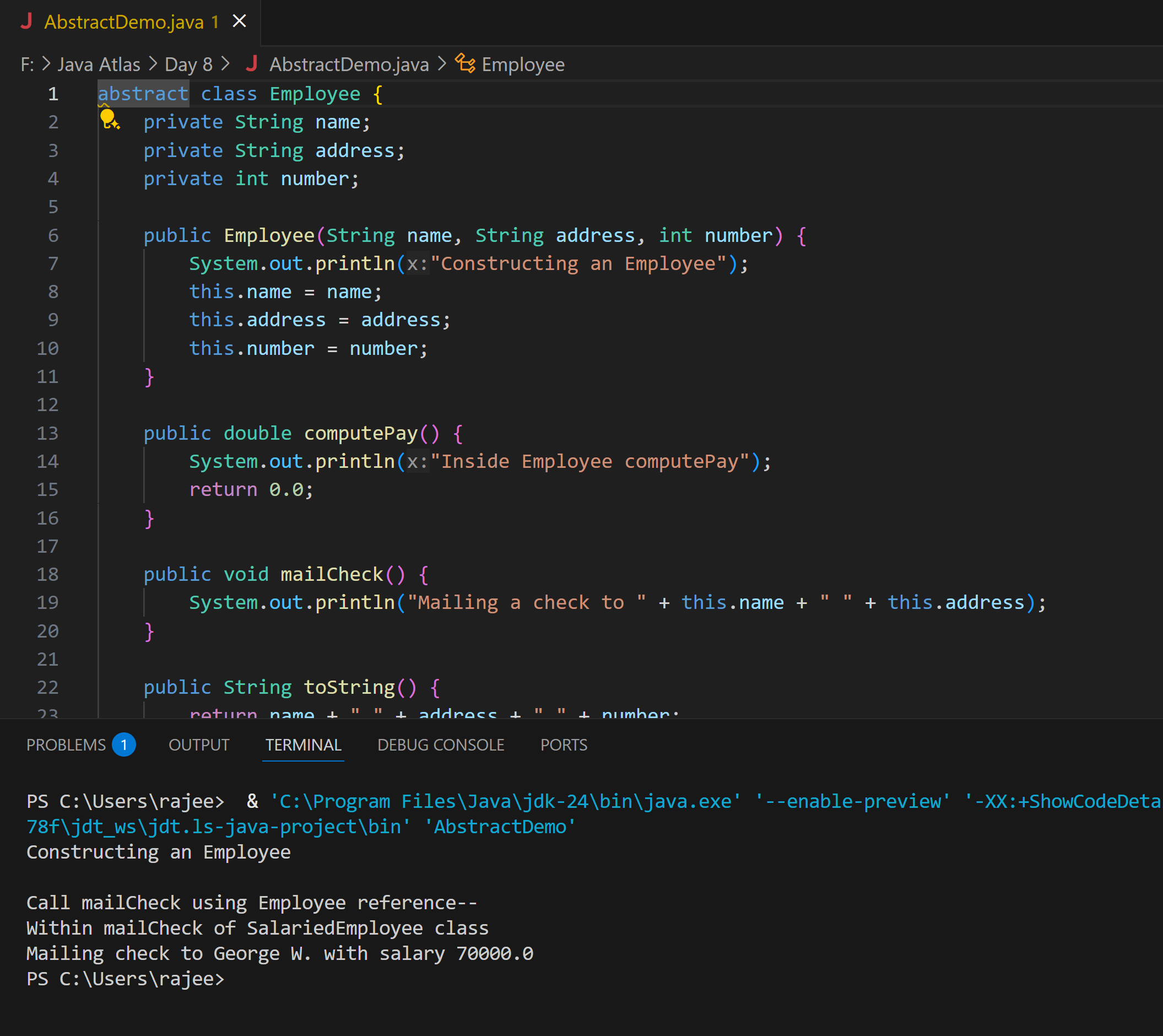
        Employee e = new SalariedEmployee("George W.", "Houston, TX", 43, 70000.00);

        System.out.println("\nCall mailCheck using Employee reference--");

        e.mailCheck();

    }

}



**Task 40**

// Working of Abstraction in Java

abstract class Gadgets {

abstract void turnOn();

abstract void turnOff();

}

// Concrete class implementing the abstract methods

class TVRemote extends Gadgets {

@Override

void turnOn() {

System.out.println("TV is turned ON.");

}

@Override

void turnOff() {

System.out.println("TV is turned OFF.");

}

}

// Main class to demonstrate abstraction

public class Main {

public static void main(String[] args) {

Gadgets remote = new TVRemote();

remote.turnOn();

remote.turnOff();

}

}

**Ans:**

