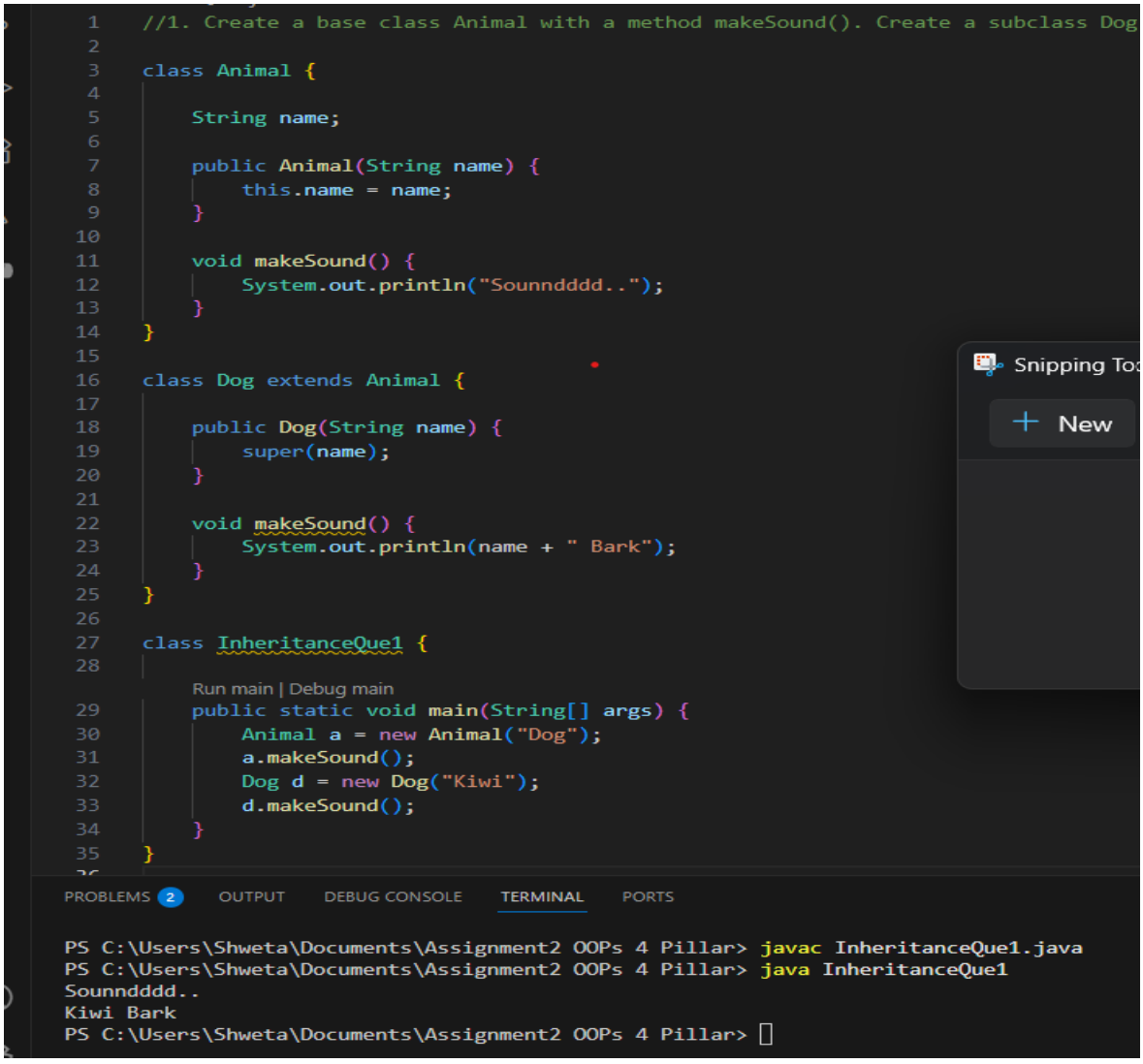
Assignment No. 2

OOPS 4 Pillar Based

1. Inheritance (5 Questions)

1. **Create a base class `Animal`** with a method `makeSound()`. Create a subclass `Dog` that overrides `makeSound()` to print `"Bark"`.

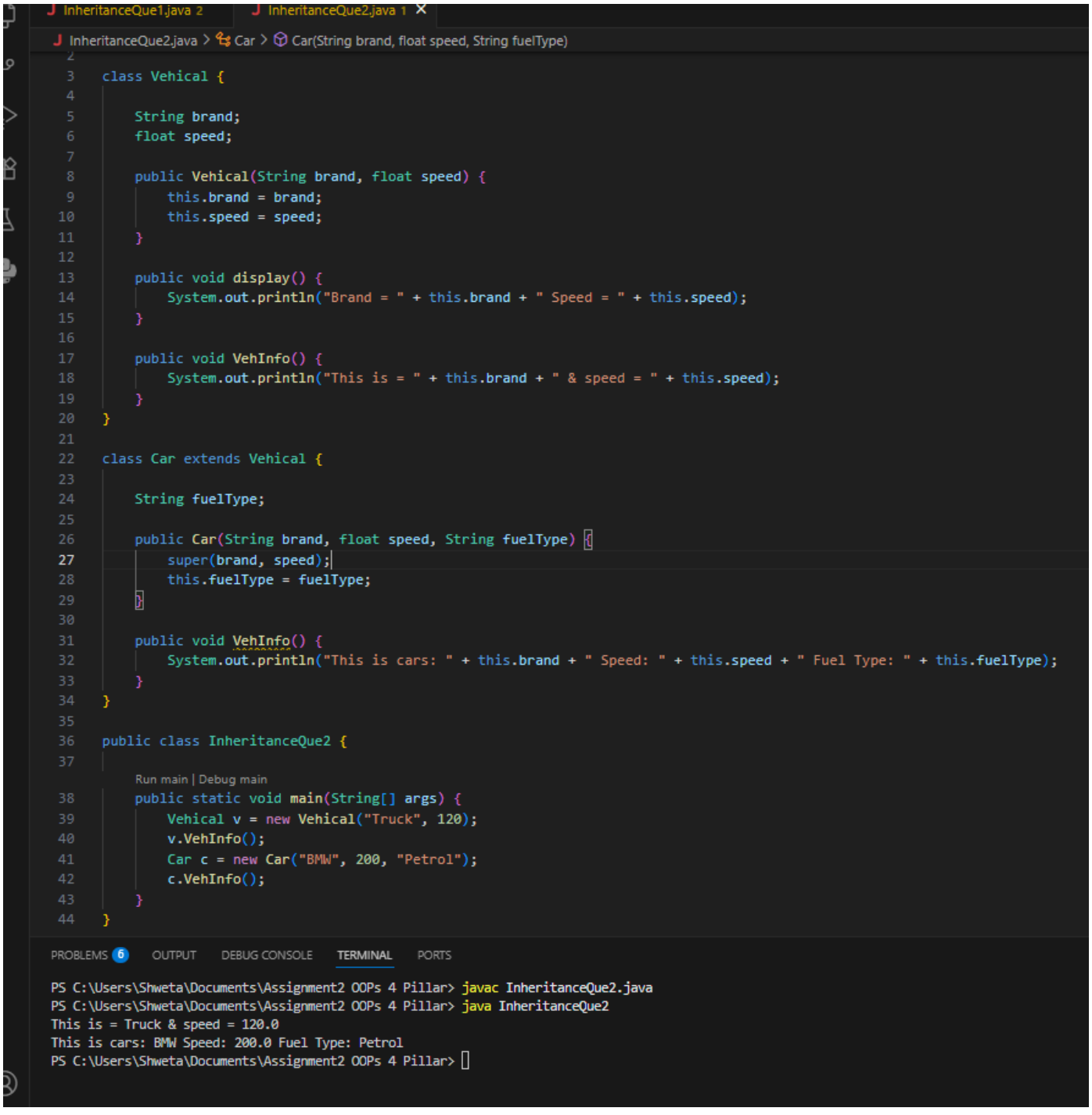


```java
//1. Create a base class Animal with a method makeSound(). Create a subclass Dog

class Animal {

    String name;

    public Animal(String name) {
        this.name = name;
    }

    void makeSound() {
        System.out.println("Sounndddd..");
    }
}

class Dog extends Animal {

    public Dog(String name) {
        super(name);
    }

    void makeSound() {
        System.out.println(name + " Bark");
    }
}

class InheritanceQue1 {

    Run main | Debug main
    public static void main(String[] args) {
        Animal a = new Animal("Dog");
        a.makeSound();
        Dog d = new Dog("Kiwi");
        d.makeSound();
    }
}
```

```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac InheritanceQue1.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java InheritanceQue1
Sounndddd..
Kiwi Bark
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

Snipping To
New

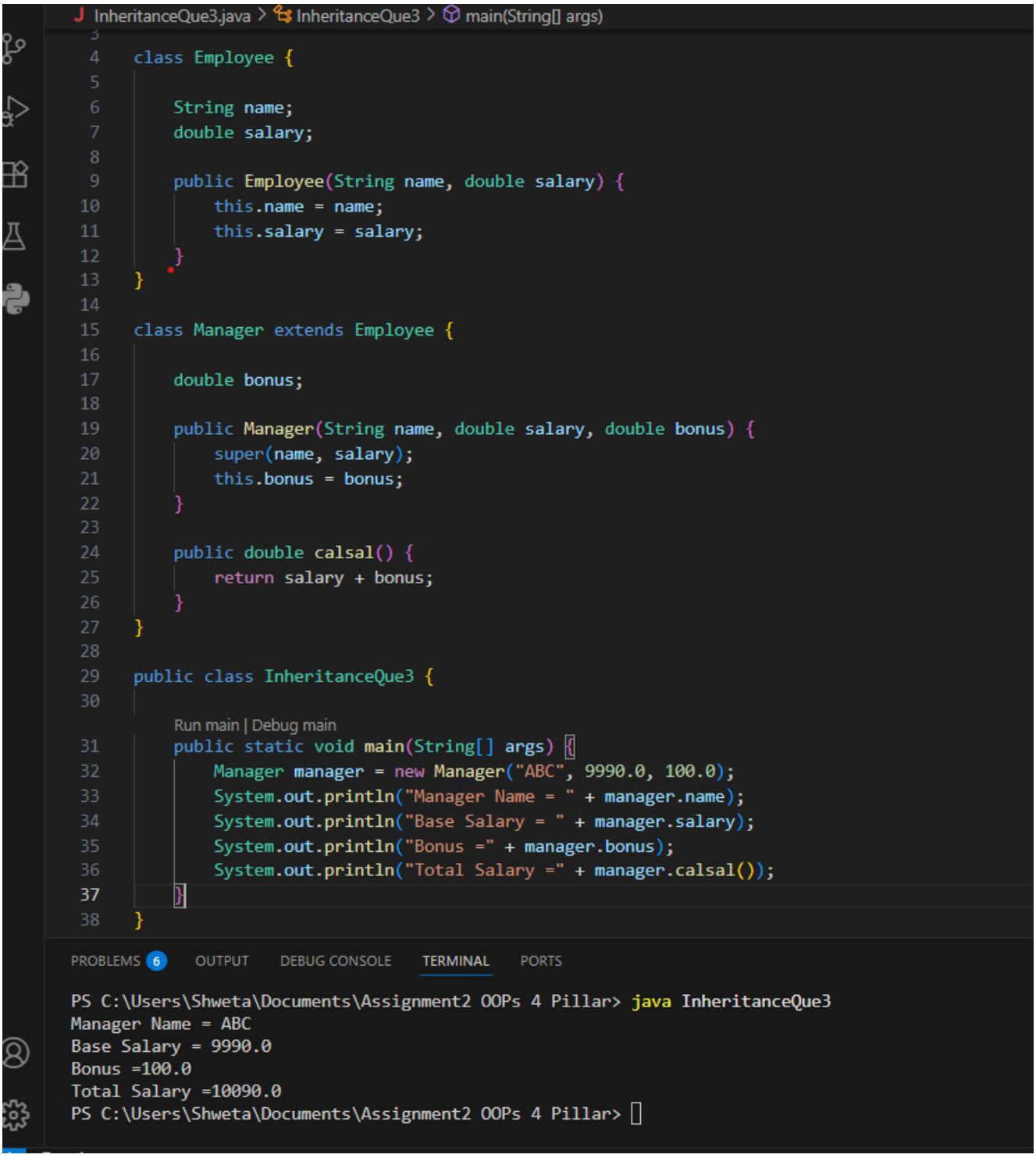
2. **Create a base class `Vehicle` with properties `brand` and `speed`. Create a subclass `Car`that adds `fuelType` and a method `displayCarDetails()`.**



```java
class Vehical {

    String brand;
    float speed;

    public Vehical(String brand, float speed) {
        this.brand = brand;
        this.speed = speed;
    }

    public void display() {
        System.out.println("Brand = " + this.brand + " Speed = " + this.speed);
    }

    public void VehInfo() {
        System.out.println("This is = " + this.brand + " & speed = " + this.speed);
    }
}

class Car extends Vehical {

    String fuelType;

    public Car(String brand, float speed, String fuelType) {
        super(brand, speed);
        this.fuelType = fuelType;
    }

    public void VehInfo() {
        System.out.println("This is cars: " + this.brand + " Speed: " + this.speed + " Fuel Type: " + this.fuelType);
    }
}

public class InheritanceQue2 {

    Run main | Debug main
    public static void main(String[] args) {
        Vehical v = new Vehical("Truck", 120);
        v.VehInfo();
        Car c = new Car("BMW", 200, "Petrol");
        c.VehInfo();
    }
}
```

```
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac InheritanceQue2.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java InheritanceQue2
This is = Truck & speed = 120.0
This is cars: BMW Speed: 200.0 Fuel Type: Petrol
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

3. **Create a base class `Employee`** with attributes `name` and `salary`. Create a subclass `Manager` that adds `bonus`. Write a method to calculate the total salary.


InheritanceQue3.java > InheritanceQue3 > main(String[] args)
class Employee {
String name;
double salary;
public Employee(String name, double salary) {
this.name = name;
this.salary = salary;
}
}
class Manager extends Employee {
double bonus;
public Manager(String name, double salary, double bonus) {
super(name, salary);
this.bonus = bonus;
}
public double calsal() {
return salary + bonus;
}
}
public class InheritanceQue3 {
Run main | Debug main
public static void main(String[] args) {
Manager manager = new Manager("ABC", 9990.0, 100.0);
System.out.println("Manager Name = " + manager.name);
System.out.println("Base Salary = " + manager.salary);
System.out.println("Bonus =" + manager.bonus);
System.out.println("Total Salary =" + manager.calsal());
}
}
PROBLEMS 6
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java InheritanceQue3
Manager Name = ABC
Base Salary = 9990.0
Bonus =100.0
Total Salary =10090.0
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>

4. **Write a Java program** where a `Person` class has `name` and `age`. Create a subclass `Student` that adds `rollNumber` and `marks`.

```java
class Person {
    String name;
    int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
class Student extends Person {
    int rollNumber;
    double marks;
    public Student(String name, int age, int rollNumber, double marks) {
        super(name, age);
        this.rollNumber = rollNumber;
        this.marks = marks;
    }
    public void displayStudentDetails() {
        System.out.println("Name " + name);
        System.out.println("Age " + age);
        System.out.println("Roll Number " + rollNumber);
        System.out.println("Marks " + marks);
    }
}
public class InheritanceQue4 {
    Run main | Debug main
    public static void main(String[] args) {
        Student student1 = new Student("Stuuuu", 22, 101, 98);
        System.out.println("Student Info  = ");
        student1.displayStudentDetails();
    }
}

```

```
PROBLEMS 6  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac InheritanceQue4.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java InheritanceQue4
Student Info  =
Name Stuuuu
Age 22
Roll Number 101
Marks 98.0
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

5. **Create a base class `Shape`** with a method `area()`. Create subclasses `Circle` and `Rectangle` that override `area()` to calculate their respective areas.

InheritanceQue5.java > InheritanceQue5 > main(String[] args)

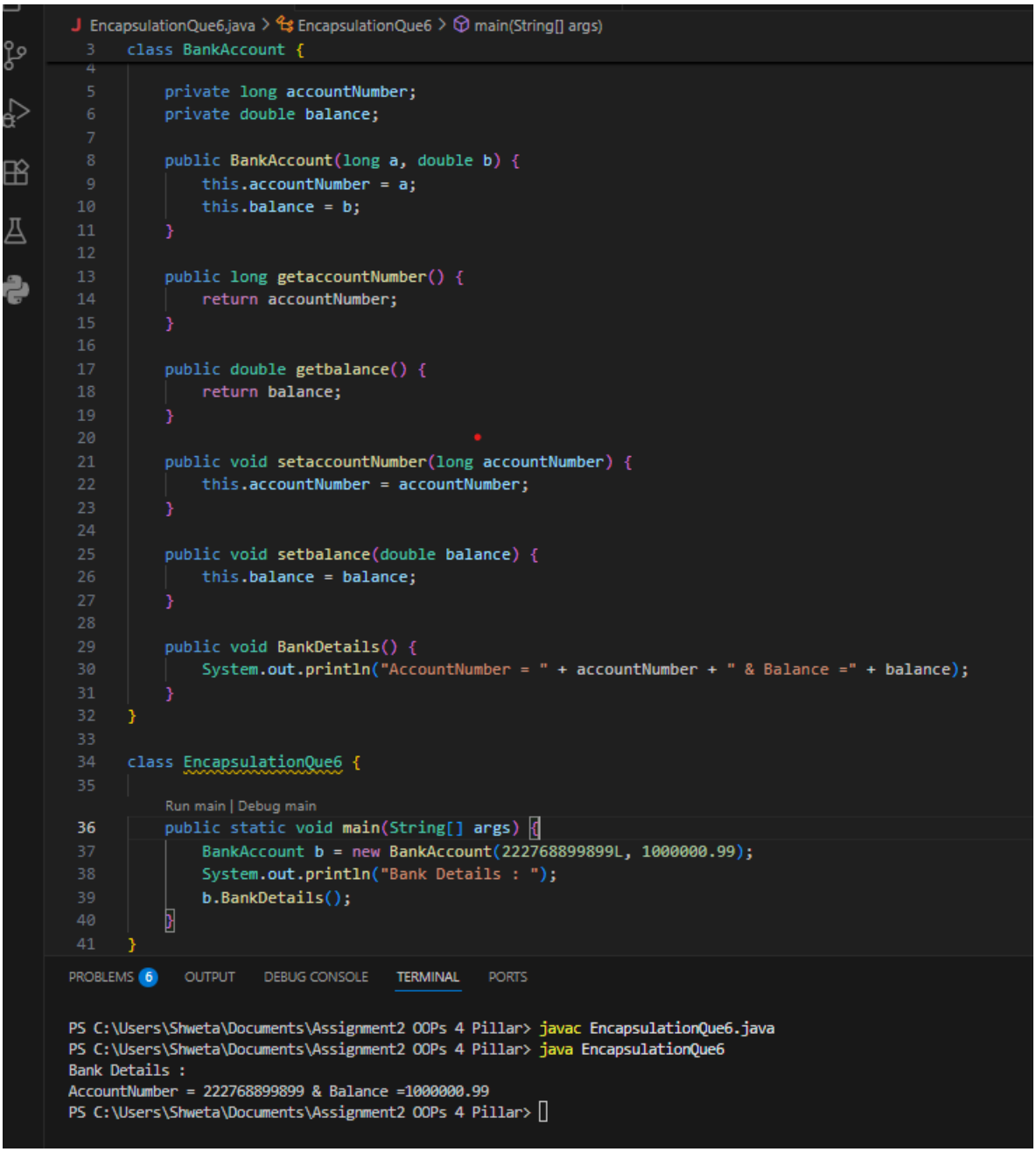
```java
//5. Create a base class Shape with a method area(). Create subclasses Circle and  Rectangle

class Shape {
    public double area() {
        return 0;
    }
}
class Circle extends Shape {
    double r;
    public Circle(double r) {
        this.r = r;
    }
    public double area() {
        return 3.14 * r * r;
    }
}
class Rectangle extends Shape {
    double length;
    double width;
    public Rectangle(double l, double w) {
        this.length = l;
        this.width = w;
    }
    public double area() {
        return length * width;
    }
}
public class InheritanceQue5 {
    Run main | Debug main
    public static void main(String[] args) {
        Circle circle = new Circle(70);
        System.out.println(circle.area());
        Rectangle rectangle = new Rectangle(9.0, 8.0);
        System.out.println(rectangle.area());
    }
}
```

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac InheritanceQue5.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java InheritanceQue5
15386.0
72.0
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

2. Encapsulation (5 Questions)

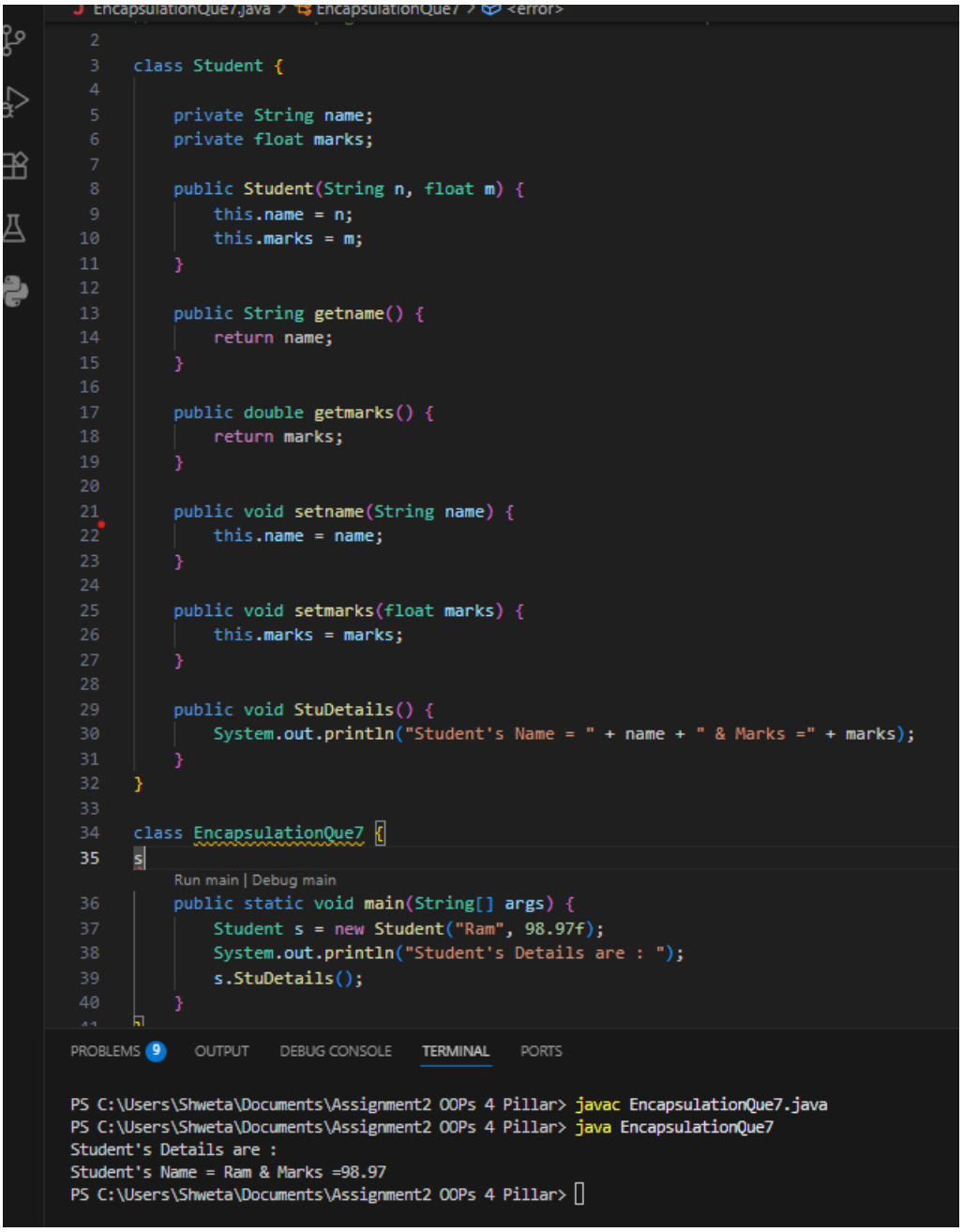
6. **Create a class `BankAccount`** with private attributes `accountNumber` and `balance`. Use getters and setters to access and modify them.



```java
class BankAccount {

    private long accountNumber;
    private double balance;

    public BankAccount(long a, double b) {
        this.accountNumber = a;
        this.balance = b;
    }

    public long getaccountNumber() {
        return accountNumber;
    }

    public double getbalance() {
        return balance;
    }

    public void setaccountNumber(long accountNumber) {
        this.accountNumber = accountNumber;
    }

    public void setbalance(double balance) {
        this.balance = balance;
    }

    public void BankDetails() {
        System.out.println("AccountNumber = " + accountNumber + " & Balance =" + balance);
    }
}

class EncapsulationQue6 {

    Run main | Debug main
    public static void main(String[] args) {
        BankAccount b = new BankAccount(222768899899L, 1000000.99);
        System.out.println("Bank Details : ");
        b.BankDetails();
    }
}
```

```
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac EncapsulationQue6.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java EncapsulationQue6
Bank Details :
AccountNumber = 222768899899 & Balance =1000000.99
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

7. **Write a Java program** to create a `Student` class with private variables `name` and `marks`. Use getters to retrieve and setters to modify the values.


EncapsulationQue7.java > EncapsulationQue7 > <error>
class Student {
private String name;
private float marks;
public Student(String n, float m) {
this.name = n;
this.marks = m;
}
public String getname() {
return name;
}
public double getmarks() {
return marks;
}
public void setname(String name) {
this.name = name;
}
public void setmarks(float marks) {
this.marks = marks;
}
public void StuDetails() {
System.out.println("Student's Name = " + name + " & Marks =" + marks);
}
}
class EncapsulationQue7 {
s
Run main | Debug main
public static void main(String[] args) {
Student s = new Student("Ram", 98.97f);
System.out.println("Student's Details are : ");
s.StuDetails();
}
PROBLEMS 9
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac EncapsulationQue7.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java EncapsulationQue7
Student's Details are :
Student's Name = Ram & Marks =98.97
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>

8. **Create a class `Car`** with private variables `model`, `year`, and `price`. Provide public methods to get and set values while ensuring `year` is not negative.

```java
class Car {

    private String model;
    private int year;
    private float price;

    public Car(String m, int y, float p) {
        this.model = m;
        this.year = y;
        this.price = p;
    }

    public String getmodel() {
        return model;
    }

    public int getyear() {
        return year;
    }

    public float getprice() {
        return price;
    }

    public void setmodel(String model) {
        this.model = model;
    }

    public void setyear(int year) {
        this.year = year;
    }

    public void setprice(float price) {
        this.price = price;
    }

    public void CarDetails() {
        System.out.println("Car's Model = " + model + " , year  =" + year + " & price = " + price);
    }
}

class EncapsulationQue8 {

    Run main | Debug main
    public static void main(String[] args) {
        Car c = new Car("BMW", 2021, 9800000.97f);
        System.out.println("Car's Details are : ");
        c.CarDetails();
    }
}
```

```
PROBLEMS 8   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac EncapsulationQue8.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java EncapsulationQue8
Car's Details are :
Car's Model = BMW , year  =2021 & price = 9800001.0
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

9. **Write a Java program** for a `Laptop` class with private attributes `brand` and `price`. Ensure price cannot be set below zero using validation inside the setter method.

EncapsulationQue9.java > EncapsulationQue9 > main(String[] args)

```
3   class Laptop {
5       private String brand;
6       private float price;
7
8       public Laptop(String b, float price) {
9           this.brand = b;
10          setprice(price);
11      }
12
13      public String getbrand() {
14          return brand;
15      }
16
17      public float getprice() {
18          return price;
19      }
20
21      public void setbrand(String brand) {
22          this.brand = brand;
23      }
24
25      public void setprice(float price) {
26          if (price >= 0) {
27              this.price = price;
28          } else {
29              System.err.println("Price cannot be set to negative value");
30          }
31      }
32
33      public void LaptopDetails() {
34          System.out.println("Laptop's Brand = " + getbrand());
35          System.out.println("& Price =" + getprice());
36      }
37  }
38
39  class EncapsulationQue9 {
40
        Run main | Debug main
41      public static void main(String[] args) {
42          Laptop l = new Laptop("HP", -89);
43          System.out.println("Laptop's Details are : ");
44          l.LaptopDetails();
45      }
46  }
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac EncapsulationQue9.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java EncapsulationQue9
Price cannot be set to negative value
Laptop's Details are :
Laptop's Brand = HP
& Price =0.0
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

10. **Create a `Patient` class** with private attributes `id`, `name`, and `disease`. Provide methods to set and get details and restrict modification of `id` once assigned.

```java
//10. Create a Patient class with private attributes id, name, and disease. Provide methods  to set a

class Patient {

    private final int id;
    private String name;
    private String disease;

    public Patient(int i, String n, String d) {
        this.id = i;
        this.name = n;
        this.disease = d;
    }

    public int getid() {
        return id;
    }

    public String getname() {
        return name;
    }

    public String getdisease() {
        return disease;
    }

    public void setname(String name) {
        this.name = name;
    }

    public void setdisease(String disease) {
        this.disease = disease;
    }

    public void PatientDetails() {
        System.out.println("id= " + id + " name = " + name + " disease = " + disease);
    }
}

class EncapsulationQue10 {

    Run main | Debug main
    public static void main(String[] args) {
        Patient p = new Patient(101, "XYZ", "Cancer");
        System.out.println("Patient Details are : ");
        p.PatientDetails();
    }
}
```

```
PROBLEMS 25  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac EncapsulationQue10.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java EncapsulationQue10
Patient Details are :
id= 101 name = XYZ disease = Cancer
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

3. Polymorphism (5 Questions)

(A) Compile-Time Polymorphism (Method Overloading)

11. **Create a `MathOperations` class** with overloaded `add()` methods: one for two integers, another for three integers, and one for two double values.

```java
class MathOperations {

    int add(int a, int b) {
        return a + b;
    }

    int add(int a, int b, int c) {
        return a + b + c;
    }

    double add(double a, double b) {
        return a + b;
    }

}

public class PolymorphismOverloadingQue11 {

    Run main | Debug main
    public static void main(String[] args) {
        MathOperations op = new MathOperations();
        System.out.println("Add 2 ints: " + op.add(99, 88));
        System.out.println("Add 3 ints: " + op.add(100, 200, 300));
        System.out.println("Add 2 doubles: " + op.add(9.9, 8.9));

    }
}
```

PROBLEMS 25 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac PolymorphismOverloadingQue11.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java PolymorphismOverloadingQue11
Add 2 ints: 187
Add 3 ints: 600
Add 2 doubles: 18.8
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

12. **Write a Java program** to create a class `Printer` that has multiple overloaded `print()` methods for `String`, `int`, and `double` values.

```java
class Printer {

    public void print(String s) {
        System.out.println("String =" + s);
    }

    public void print(int i) {
        System.out.println("Integer =" + i);
    }

    public void print(double d) {
        System.out.println("Double =" + d);
    }
}

public class PolymorphismOverloadingQue12 {

    Run main | Debug main
    public static void main(String[] args) {
        Printer p = new Printer();
        p.print("XYZ");
        p.print(99);
        p.print(999.9);
    }
}
```

PROBLEMS 25 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac PolymorphismOverloadingQue12.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java PolymorphismOverloadingQue12
String =XYZ
Integer =99
Double =999.9
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

13. **Create a `Calculator` class** with overloaded `multiply()` methods to accept integers, doubles, and a mix of both.

```java
class Calculator {

    int multiple(int a, int b) {
        return a * b;
    }

    double multiply(double a, double b) {
        return a * b;
    }

    double multiply(int a, double b) {
        return a * b;
    }

}

public class PolymorphismOverloadingQue13 {

    Run main | Debug main
    public static void main(String[] args) {
        Calculator cal = new Calculator();
        System.out.println(" Multiplication of Integers =" + cal.multiple(8, 9));
        System.out.println(" Multiplication of Double =" + cal.multiply(8.8, 9.9));
        System.out.println(" Multiplication of Mix of both =" + cal.multiply(8, 9.0));

    }
}
```

```
PROBLEMS 25   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

Double =999.9
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac PolymorphismOverloadingQue13.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java PolymorphismOverloadingQue13
 Multiplication of Integers =72
 Multiplication of Double =87.12
 Multiplication of Mix of both =72.0
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

Indexing completed.

14. **Write a Java program** where a `Shape` class has overloaded `draw()` methods, accepting different numbers of parameters to draw different shapes.

```java
class Shape {

    void draw(int r) {
        System.out.println("Print Circle");
    }

    void draw(float l, float w) {
        System.out.println("Print Rectangle");
    }

    void draw(int s1, int s2, int s3) {
        System.out.println("Print triangle");
    }

}

public class PolymorphismOverloadingQue14 {

    Run main | Debug main
    public static void main(String[] args) {
        Shape s = new Shape();
        s.draw(2);
        s.draw(9.0f, 8.0f);
        s.draw(9, 9, 9);
    }
}
```

```
PROBLEMS 25   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac PolymorphismOverloadingQue14.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java PolymorphismOverloadingQue14
Print Circle
Print Rectangle
Print triangle
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

15. **Create a class `CurrencyConverter`** that has overloaded methods to convert different currencies (INR to USD, INR to EUR, etc.).

```java
//15. Create a class CurrencyConverter that has overloaded methods to convert different  c

class CurrencyConverter {

    public double convert(double inr) {
        return inr * 0.012;
    }

    public double convert(double inr, String a) {
        return inr * 0.011;
    }

}

public class PolymorphismOverloadingQue15 {

    Run main | Debug main
    public static void main(String[] args) {
        CurrencyConverter c = new CurrencyConverter();
        System.err.println("INR to USD =" + c.convert(9000));
        System.out.println("INR to EUR =" + c.convert(2000, "USD"));
    }
}
```

```
PROBLEMS 68  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac PolymorphismOverloadingQue15.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java PolymorphismOverloadingQue15
INR to USD =108.0
INR to EUR =22.0
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

(B) Runtime Polymorphism (Method Overriding)

16. **Create a base class `Animal`** with `speak()` method. Create subclasses `Dog` and `Cat` that override `speak()` to print different sounds.

```java
class Animal {

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

class Dog extends Animal {

    void sound() {
        System.out.println("Dog barks");
    }
}

class Cat extends Animal {

    void sound() {
        System.out.println("Cat meows");
    }
}

public class PolymorphismOverloadingQue16 {

    Run main | Debug main
    public static void main(String[] args) {
        Animal a = new Dog();
        a.sound();
        Animal b = new Cat();
        b.sound();
    }
}
```

PROBLEMS 27 OUTPUT DEBUG CONSOLE TERMINAL PORTS

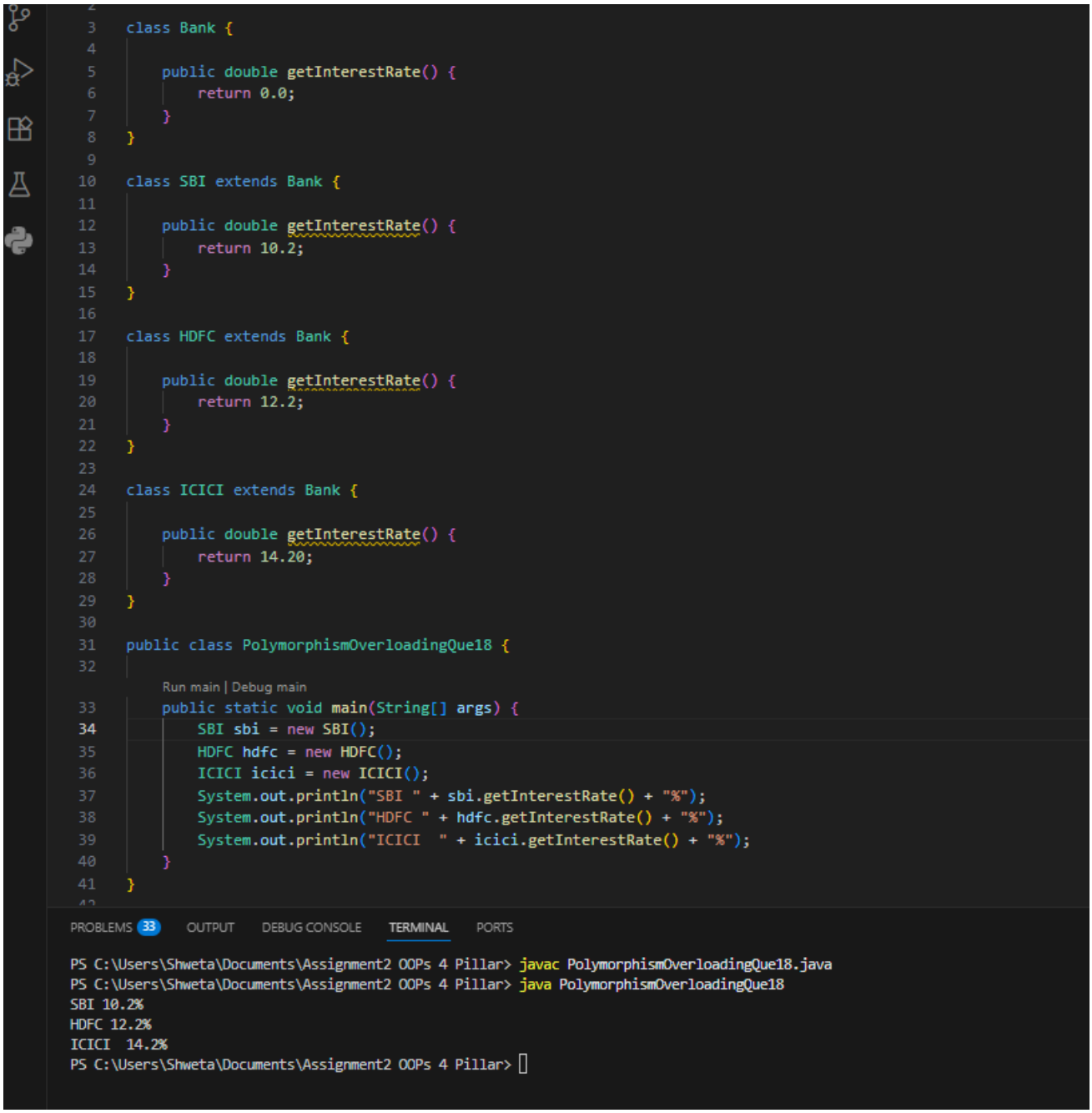
```
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac PolymorphismOverloadingQue16.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java PolymorphismOverloadingQue16
Dog barks
Cat meows
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

17. **Write a Java program** where a `Vehicle` class has a `run()` method. Create subclasses `Bike` and `Car` that override `run()` with specific messages.

```java
class Vehicle {

    public void run() {
        System.out.println("vehicle");
    }
}

class Bike extends Vehicle {

    public void run() {
        System.out.println("Bike");
    }
}

class Car extends Vehicle {

    public void run() {
        System.out.println("Car");
    }
}

class PolymorphismOverloadingQue17 {

    Run main | Debug main
    public static void main(String[] args) {
        Vehicle b = new Bike();
        Vehicle c = new Car();
        System.out.println("Vehicle");
        b.run();
        c.run();
    }
}
```

```
PROBLEMS 30  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac PolymorphismOverloadingQue17.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java PolymorphismOverloadingQue17
Vehicle
Bike
Car
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

18. **Create a `Bank` class** with a method `getInterestRate()`. Create subclasses `SBI`, `HDFC`, and `ICICI` that override the method with their respective interest rates.



```java
class Bank {

    public double getInterestRate() {
        return 0.0;
    }
}

class SBI extends Bank {

    public double getInterestRate() {
        return 10.2;
    }
}

class HDFC extends Bank {

    public double getInterestRate() {
        return 12.2;
    }
}

class ICICI extends Bank {

    public double getInterestRate() {
        return 14.20;
    }
}

public class PolymorphismOverloadingQue18 {

    Run main | Debug main
    public static void main(String[] args) {
        SBI sbi = new SBI();
        HDFC hdfc = new HDFC();
        ICICI icici = new ICICI();
        System.out.println("SBI " + sbi.getInterestRate() + "%");
        System.out.println("HDFC " + hdfc.getInterestRate() + "%");
        System.out.println("ICICI  " + icici.getInterestRate() + "%");
    }
}
```

```
PROBLEMS 33  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac PolymorphismOverloadingQue18.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java PolymorphismOverloadingQue18
SBI 10.2%
HDFC 12.2%
ICICI  14.2%
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

19. **Write a Java program** where a `Phone` class has a method `call()`. Create subclasses `Smartphone` and `Landline` that override `call()` differently.

```java
class Phone {

    public void call() {
        System.out.println("call");
    }
}

class Smartphone extends Phone {

    public void call() {
        System.out.println("Smartphone");
    }
}

class Landline extends Phone {

    public void call() {
        System.out.println("Landline");
    }
}

public class PolymorphismOverloadingQue19 {

    Run main | Debug main
    public static void main(String[] args) {
        Phone myPhone = new Phone();
        myPhone.call();
        Smartphone s = new Smartphone();
        s.call();
        Landline l = new Landline();
        l.call();
    }
}
```

```
PROBLEMS 35  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac PolymorphismOverloadingQue19.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java PolymorphismOverloadingQue19
call
Smartphone
Landline
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

20. **Create a `Browser` class** with a method `openWebsite()`. Create subclasses `Chrome` and `Firefox` that override `openWebsite()` with specific implementation details.

```java
class Browser {

    public void openWebsite() {
        System.out.println("Browser Website");
    }
}

class Chrome extends Browser {

    public void openWebsite() {
        System.out.println("Chrome Website");
    }
}

class Firefox extends Browser {

    public void openWebsite() {
        System.out.println("Firefox Website");
    }
}

public class PolymorphismOverloadingQue20 {

    Run main | Debug main
    public static void main(String[] args) {
        Browser b = new Browser();
        b.openWebsite();
        Chrome c = new Chrome();
        c.openWebsite();
        Firefox f = new Firefox();
        f.openWebsite();
    }
}
```

```
PROBLEMS 37  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac PolymorphismOverloadingQue20.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java PolymorphismOverloadingQue20
Browser Website
Chrome Website
Firefox Website
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

4. Abstraction (5 Questions)

21. **Create an abstract class `Vehicle`** with an abstract method `start()`. Create subclasses `Car` and `Bike` that provide their own implementation of `start()`.

```java
abstract class Vehicle {

    abstract void start();
}

class Car extends Vehicle {

    void start() {
        System.out.println("Car starts");
    }
}

class Bike extends Vehicle {

    void start() {
        System.out.println("Bike starts");
    }
}

class AbstractionQue21 {

    Run main | Debug main
    public static void main(String[] args) {
        Car c = new Car();
        c.start();
        Bike b = new Bike();
        b.start();
        Vehicle v = new Car();
        v.start();
    }
}
```

PROBLEMS 40 OUTPUT DEBUG CONSOLE TERMINAL PORTS

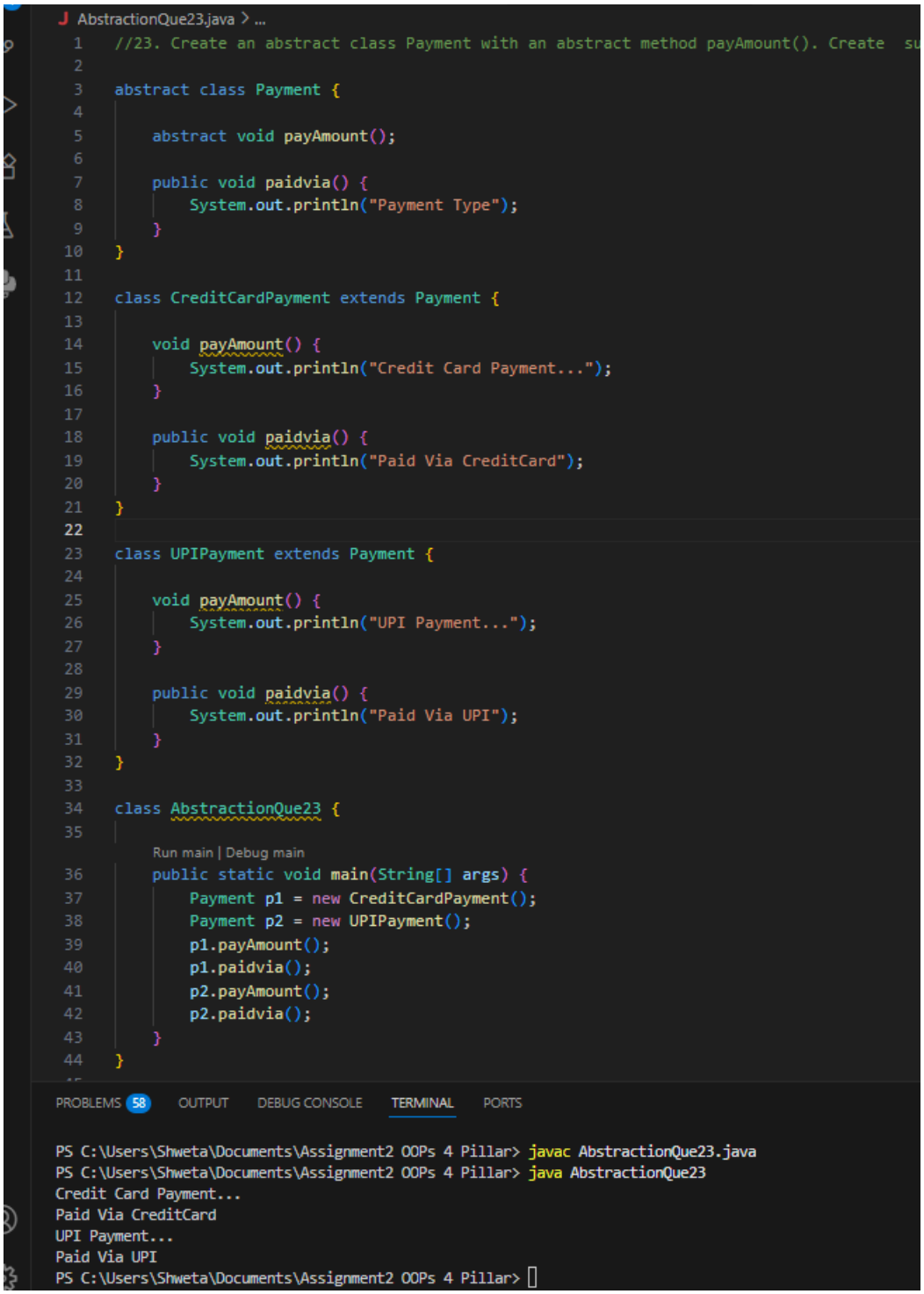
```
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac AbstractionQue21.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java AbstractionQue21
Car starts
Bike starts
Car starts
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

22. **Write a Java program** with an abstract class `Shape` that has an abstract method `calculateArea()`. Implement it in `Circle` and `Rectangle` classes.

```java
abstract class Shape {
    abstract void calculateArea();
}
class Circle extends Shape {
    private double radius;
    public Circle(double r) {
        this.radius = r;
    }
    void calculateArea() {
        double area = 3.14 * radius * radius;
        System.out.println("Area of Circle = " + area);
    }
}
class Rectangle extends Shape {

    private double length;
    private double width;
    public Rectangle(double l, double w) {
        this.length = l;
        this.width = w;
    }
    void calculateArea() {
        double area = length * width;
        System.out.println("Area of Rectangle = " + area);
    }
}
class AbstractionQue22 {
    Run main | Debug main
    public static void main(String[] args) {
        Shape c = new Circle(9);
        Shape r = new Rectangle(9.0, 9.9);
        c.calculateArea();
        r.calculateArea();
    }
}
```

```
PROBLEMS 52  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac AbstractionQue22.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java AbstractionQue22
Area of Circle = 254.34
Area of Rectangle = 89.10000000000001
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

23. **Create an abstract class `Payment`** with an abstract method `payAmount()`. Create subclasses `CreditCardPayment` and `UPIPayment` that implement it differently.



```java
//23. Create an abstract class Payment with an abstract method payAmount(). Create  su

abstract class Payment {

    abstract void payAmount();

    public void paidvia() {
        System.out.println("Payment Type");
    }
}

class CreditCardPayment extends Payment {

    void payAmount() {
        System.out.println("Credit Card Payment...");
    }

    public void paidvia() {
        System.out.println("Paid Via CreditCard");
    }
}

class UPIPayment extends Payment {

    void payAmount() {
        System.out.println("UPI Payment...");
    }

    public void paidvia() {
        System.out.println("Paid Via UPI");
    }
}

class AbstractionQue23 {

    Run main | Debug main
    public static void main(String[] args) {
        Payment p1 = new CreditCardPayment();
        Payment p2 = new UPIPayment();
        p1.payAmount();
        p1.paidvia();
        p2.payAmount();
        p2.paidvia();
    }
}
```

```
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac AbstractionQue23.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java AbstractionQue23
Credit Card Payment...
Paid Via CreditCard
UPI Payment...
Paid Via UPI
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

24. **Write a Java program** with an abstract class `Employee` that has an abstract method `calculateSalary()`. Implement it in `FullTimeEmployee` and `PartTimeEmployee`classes.

```java

abstract class Employee {

    double salary;

    abstract void calculateSalary();
}

class FullTimeEmployee extends Employee {

    int workingdays = 27;
    int dailypay = 2000;

    void calculateSalary() {
        this.salary = workingdays * dailypay;
        System.out.println("Fulltime Employee Salary = " + salary);
    }
}

class PartTimeEmployee extends Employee {

    int workingdays = 20;
    int dailypay = 700;

    void calculateSalary() {
        this.salary = workingdays * dailypay;
        System.out.println("Parttime Employee Salary = " + salary);
    }
}

class AbstractionQue24 {

    Run main | Debug main
    public static void main(String[] args) {
        Employee e1 = new FullTimeEmployee();
        e1.calculateSalary();
        Employee e2 = new PartTimeEmployee();
        e2.calculateSalary();
    }
}
```

```
PROBLEMS 61  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac AbstractionQue24.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java AbstractionQue24
Fulltime Employee Salary = 54000.0
Parttime Employee Salary = 14000.0
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```

25. **Create an abstract class `Appliance`** with abstract methods `turnOn()` and `turnOff()`. Implement these in `Fan` and `Light` classes.

```java
abstract class Appliance {

    abstract void turnOn();

    abstract void turnOff();

}

class Fan extends Appliance {

    public void turnOn() {
        System.out.println("TurnOnn Fannn");
    }

    public void turnOff() {
        System.out.println("TurnOff Fannn");
    }
}

class Light extends Appliance {

    public void turnOff() {
        System.out.println("TurnOnn Lightt");
    }

    public void turnOn() {
        System.out.println("TurnOff Lightt");
    }
}

public class AbstractionQue25 {

    Run main | Debug main
    public static void main(String[] args) {
        Fan f = new Fan();
        Light l = new Light();
        f.turnOn();
        f.turnOff();
        l.turnOn();
        l.turnOff();
    }
}
```

PROBLEMS 71 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> javac AbstractionQue25.java
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar> java AbstractionQue25
TurnOnn Fannn
TurnOff Fannn
TurnOff Lightt
TurnOnn Lightt
PS C:\Users\Shweta\Documents\Assignment2 OOPs 4 Pillar>
```