Java Basics & OOPs Assignment Questions

Java Basics

1. What is Java? Explain its features.

Java is a high-level, object-oriented, platform-independent programming language. It was developed by Sun Microsystems (now owned by Oracle).

Features:

- **Platform Independent**: Compile once, run anywhere (WORA).
- **Object-Oriented**: Everything is treated as an object.
- **Secure**: Runs in a virtual machine sandbox.
- **Robust**: Strong memory management.
- **Multithreaded**: Supports multithreaded programming.
- **High Performance**: Just-In-Time (JIT) compiler improves performance.

2. Explain the Java program execution process.

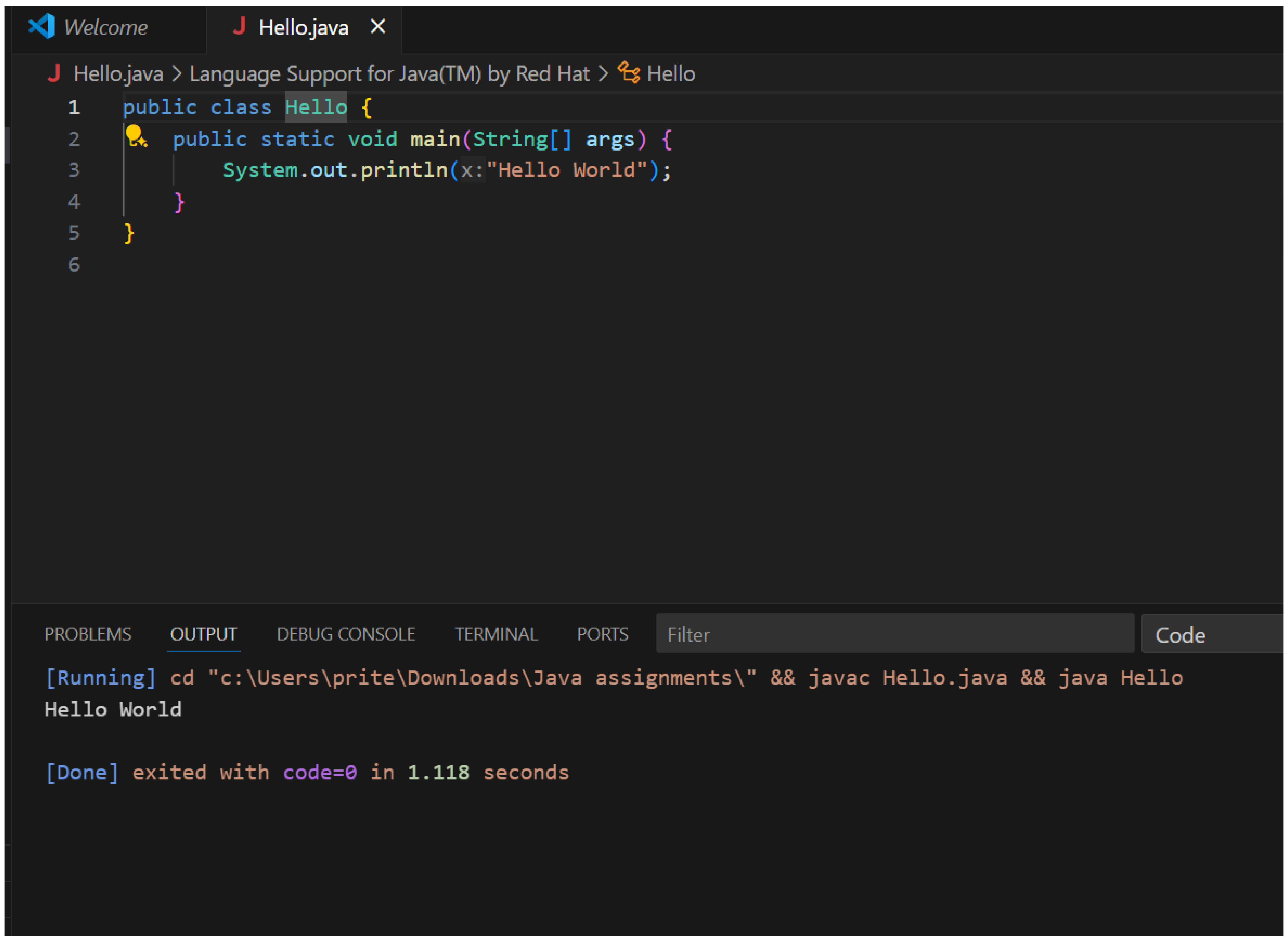
1. Write Java code (.java file)
2. Compile using javac → generates .class bytecode
3. Execute using JVM (java command) → runs on any platform

3. Write a simple Java program to display 'Hello World'.

java

CopyEdit

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```



```java
public class Hello {
    public static void main(String[] args) {
        System.out.println(x:"Hello World");
    }
}
```

```
[Running] cd "c:\Users\prite\Downloads\Java assignments\" && javac Hello.java && java Hello
Hello World

[Done] exited with code=0 in 1.118 seconds
```

4. What are data types in Java? List and explain them.

Java has two types:

- **Primitive**: int, float, double, char, boolean, byte, short, long
- **Non-primitive**: String, Array, Class, Interface

Example:

java

CopyEdit

int age = 25;

String name = "Sonal";

5. Difference between JDK, JRE, and JVM

Term	**Description**
JVM	Runs Java bytecode
JRE	JVM + libraries (for running Java apps)
JDK	JRE + compiler and tools (for developing Java apps)

6. What are variables in Java? Explain with examples.

A **variable** is a container for storing data values.

java

CopyEdit

```
int x = 10; // integer variable
String name = "Harsh"; // string variable
```

7. Different types of operators in Java

- **Arithmetic**: +, -, *, /, %
- **Relational**: ==, !=, >, <, >=, <=
- **Logical**: &&, ||, !
- **Assignment**: =, +=, -=, etc.
- **Unary**: ++, --
- **Bitwise**: &, |, ^

8. Control statements in Java (if, if-else, switch)

java

CopyEdit

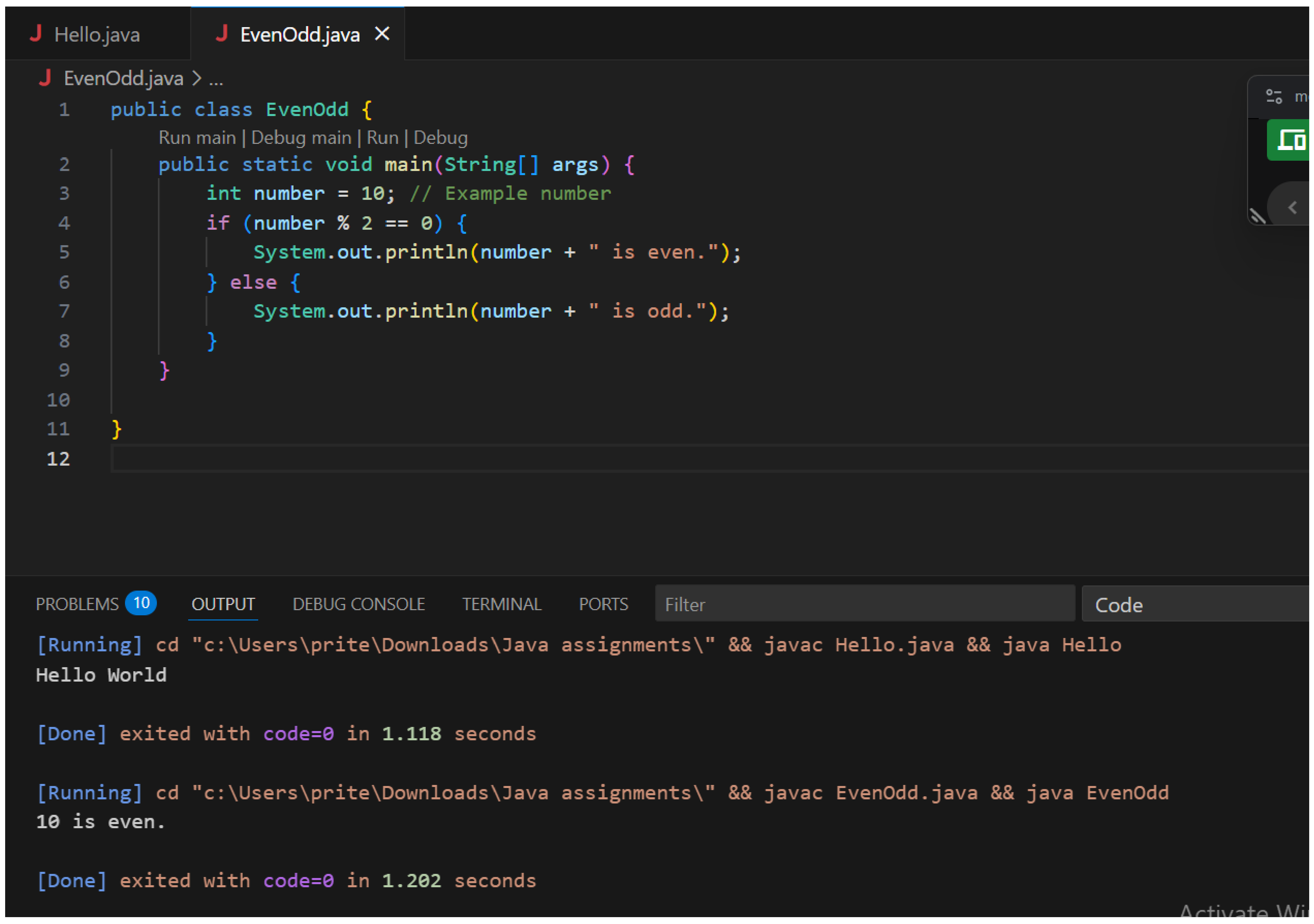
```
int x = 10;
if (x > 5) {
  System.out.println("Greater than 5");
} else {
  System.out.println("5 or less");
}

switch (x) {
  case 10: System.out.println("Ten"); break;
  default: System.out.println("Other");
}
```

9. Java program to find even or odd number

java

CopyEdit

```
import java.util.Scanner;

public class EvenOdd {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int num = sc.nextInt();

      if (num % 2 == 0)
         System.out.println("Even");
      else
         System.out.println("Odd");
   }
}
```



```
public class EvenOdd {
    public static void main(String[] args) {
        int number = 10; // Example number
        if (number % 2 == 0) {
            System.out.println(number + " is even.");
        } else {
            System.out.println(number + " is odd.");
        }
    }

}
```

```
[Running] cd "c:\Users\prite\Downloads\Java assignments\" && javac Hello.java && java Hello
Hello World

[Done] exited with code=0 in 1.118 seconds

[Running] cd "c:\Users\prite\Downloads\Java assignments\" && javac EvenOdd.java && java EvenOdd
10 is even.

[Done] exited with code=0 in 1.202 seconds
```

10. Difference between while and do-while loop

While Loop	**Do-While Loop**
Condition checked first	Condition checked after execution
May never execute	Executes at least once

Object-Oriented Programming (OOPs)

1. Principles of OOPs in Java

- **Encapsulation**: Data hiding using classes
- **Abstraction**: Hiding implementation details
- **Inheritance**: Code reuse through subclasses
- **Polymorphism**: Many forms of methods/objects

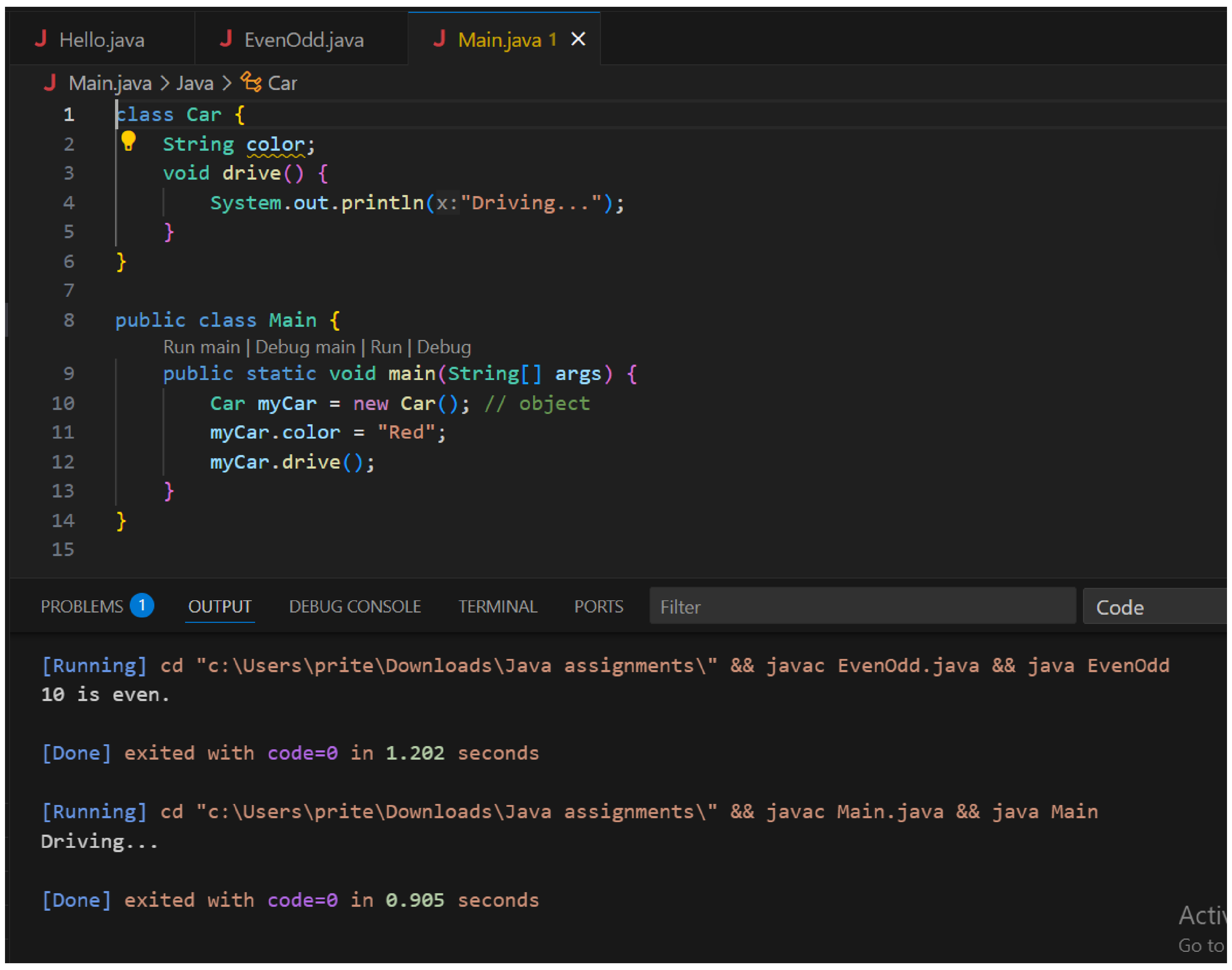
2. What is a class and object in Java?

java

CopyEdit

```
class Car {
  String color;
  void drive() {
    System.out.println("Driving...");
  }
}

public class Main {
  public static void main(String[] args) {
    Car myCar = new Car(); // object
    myCar.color = "Red";
    myCar.drive();
  }
```

}



```
class Car {
    String color;
    void drive() {
        System.out.println(x:"Driving...");
    }
}

public class Main {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Car myCar = new Car(); // object
        myCar.color = "Red";
        myCar.drive();
    }
}
```

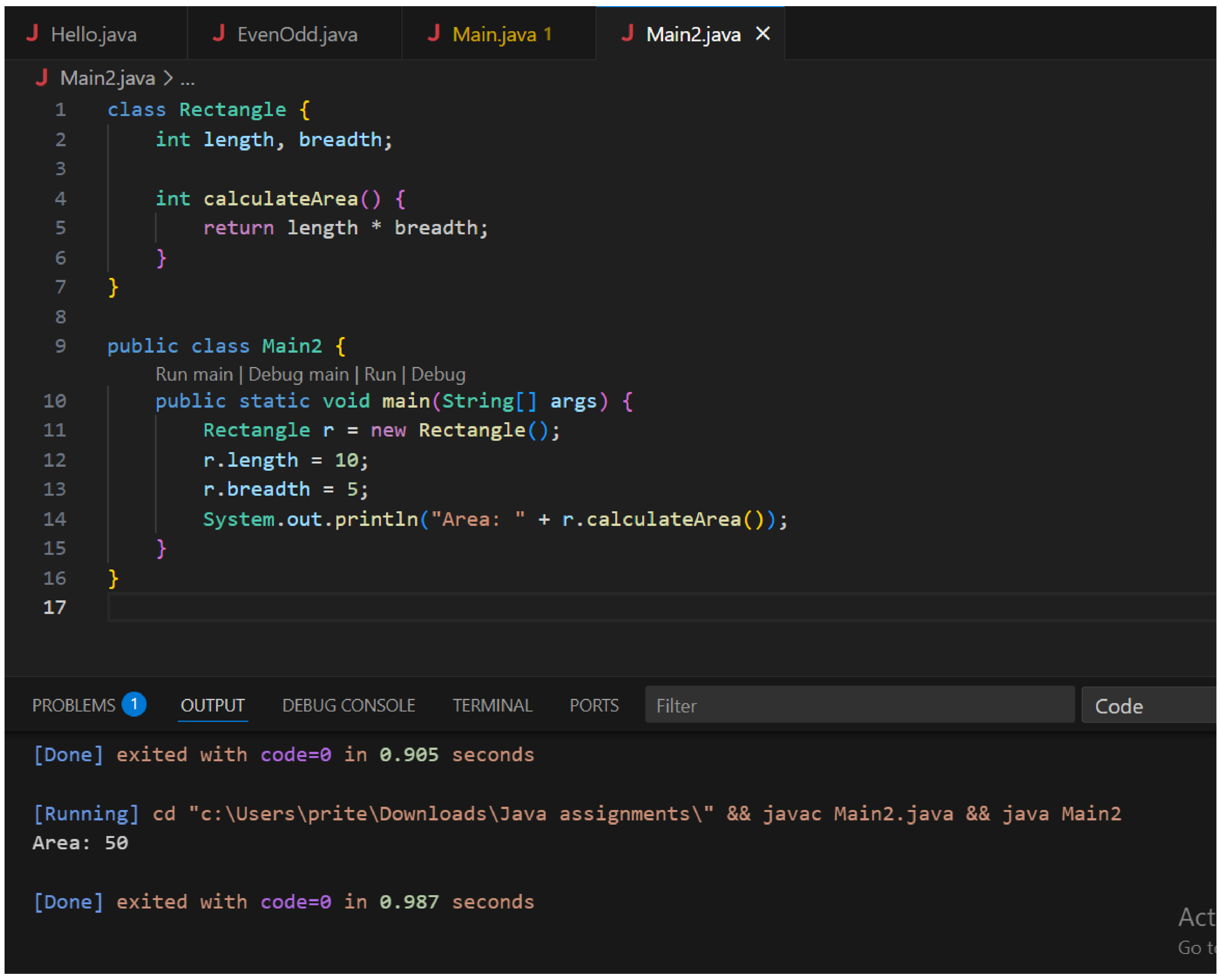
```
[Running] cd "c:\Users\prite\Downloads\Java assignments\" && javac EvenOdd.java && java EvenOdd
10 is even.

[Done] exited with code=0 in 1.202 seconds

[Running] cd "c:\Users\prite\Downloads\Java assignments\" && javac Main.java && java Main
Driving...

[Done] exited with code=0 in 0.905 seconds
```

3. Program to calculate area of rectangle

java

CopyEdit

```
class Rectangle {
    int length, breadth;

    int calculateArea() {
        return length * breadth;
    }
}

public class Main {
    public static void main(String[] args) {
        Rectangle r = new Rectangle();
        r.length = 10;
        r.breadth = 5;
        System.out.println("Area: " + r.calculateArea());
    }
}
```



```java
class Rectangle {
    int length, breadth;

    int calculateArea() {
        return length * breadth;
    }
}

public class Main2 {
    public static void main(String[] args) {
        Rectangle r = new Rectangle();
        r.length = 10;
        r.breadth = 5;
        System.out.println("Area: " + r.calculateArea());
    }
}
```

```
[Done] exited with code=0 in 0.905 seconds

[Running] cd "c:\Users\prite\Downloads\Java assignments\" && javac Main2.java && java Main2
Area: 50

[Done] exited with code=0 in 0.987 seconds
```

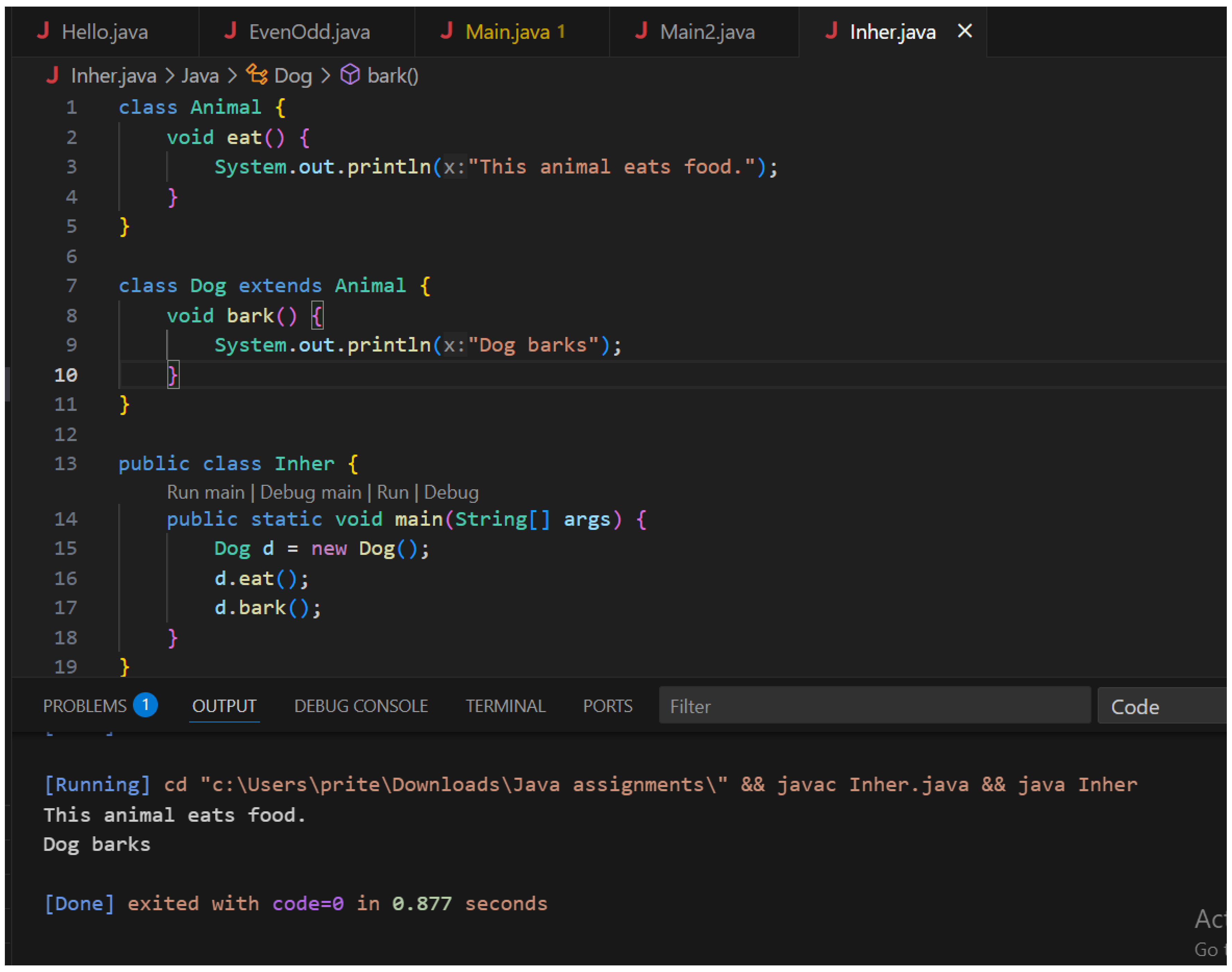
4. Inheritance with real-life example

java

CopyEdit

```
class Animal {
    void eat() {
        System.out.println("This animal eats food.");
    }
}

class Dog extends Animal {
    void bark() {
        System.out.println("Dog barks");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.eat();
        d.bark();
    }
}
```



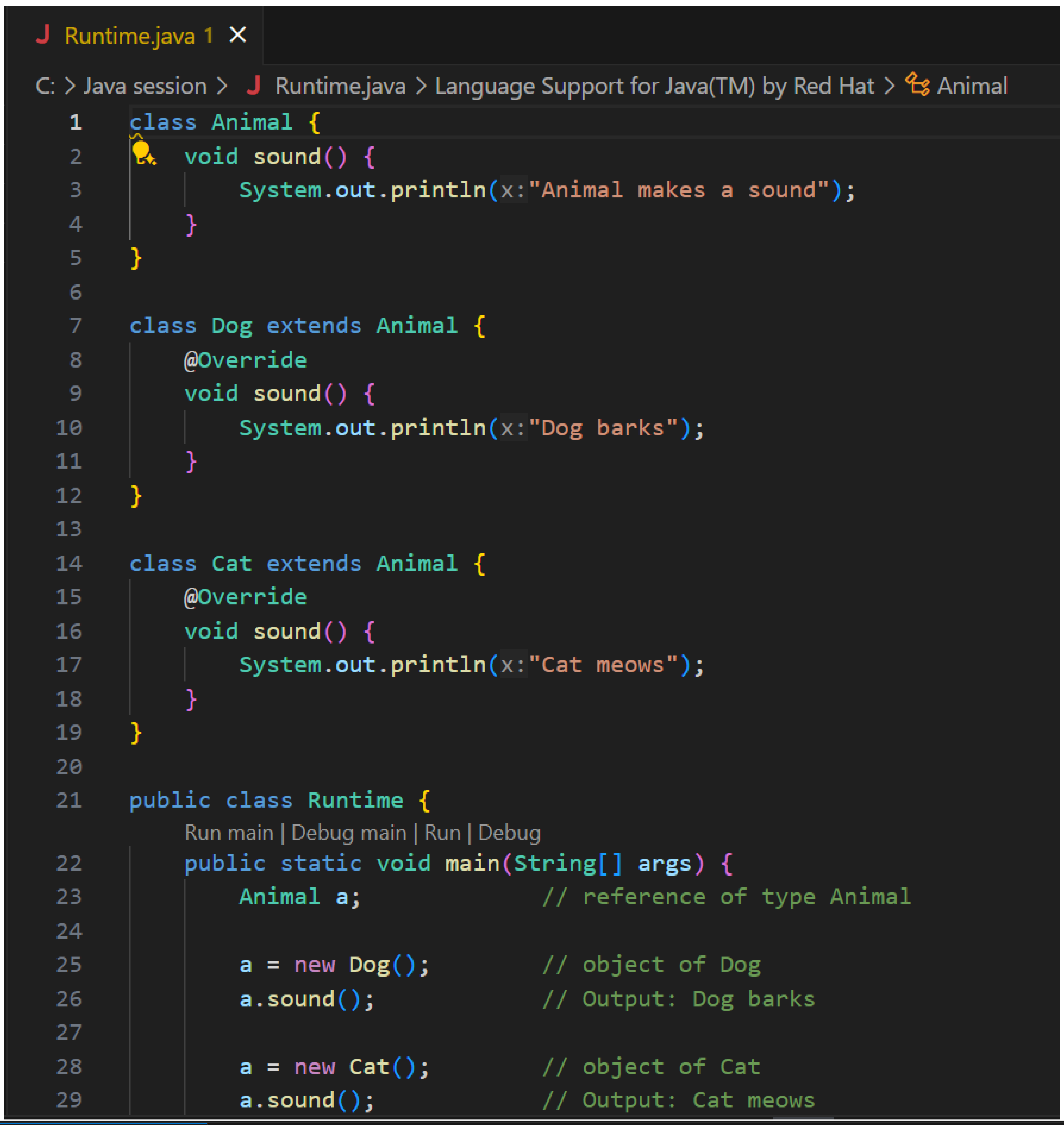
```
class Animal {
    void eat() {
        System.out.println(x:"This animal eats food.");
    }
}

class Dog extends Animal {
    void bark() {
        System.out.println(x:"Dog barks");
    }
}

public class Inher {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.eat();
        d.bark();
    }
}
```

```
[Running] cd "c:\Users\prite\Downloads\Java assignments\" && javac Inher.java && java Inher
This animal eats food.
Dog barks

[Done] exited with code=0 in 0.877 seconds
```

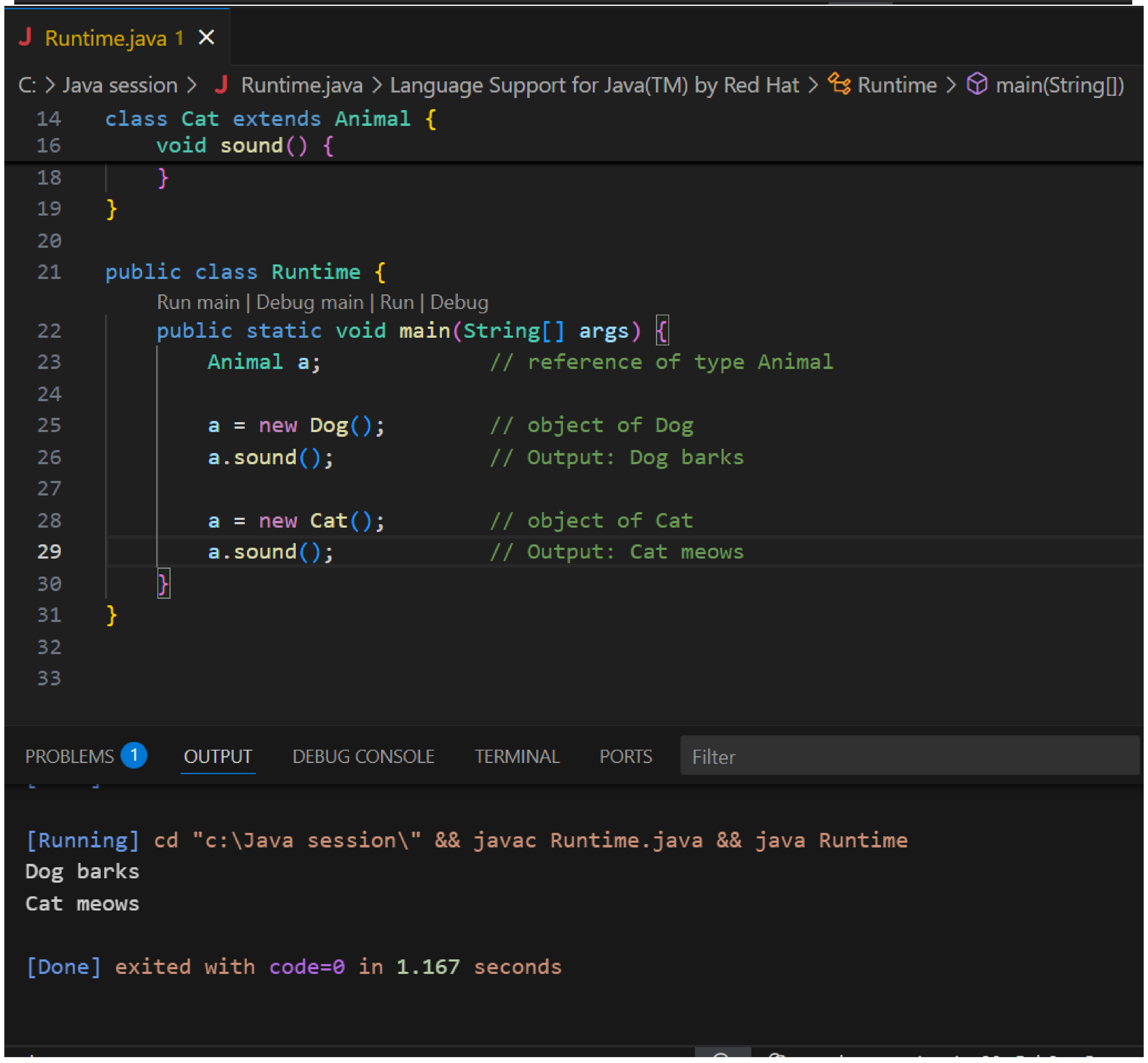
5. What is polymorphism?

Runtime (method overriding):

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

class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}

class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Cat meows");
    }
}

public class Runtime {
    public static void main(String[] args) {
        Animal a;            // reference of type Animal

        a = new Dog();       // object of Dog
        a.sound();           // Output: Dog barks

        a = new Cat();       // object of Cat
        a.sound();           // Output: Cat meows
```


Runtime.java 1
C: > Java session > Runtime.java > Language Support for Java(TM) by Red Hat > Animal

```java
class Animal {
    void sound() {
        System.out.println(x:"Animal makes a sound");
    }
}

class Dog extends Animal {
    @Override
    void sound() {
        System.out.println(x:"Dog barks");
    }
}

class Cat extends Animal {
    @Override
    void sound() {
        System.out.println(x:"Cat meows");
    }
}

public class Runtime {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Animal a;                // reference of type Animal

        a = new Dog();           // object of Dog
        a.sound();               // Output: Dog barks

        a = new Cat();           // object of Cat
        a.sound();               // Output: Cat meows
```

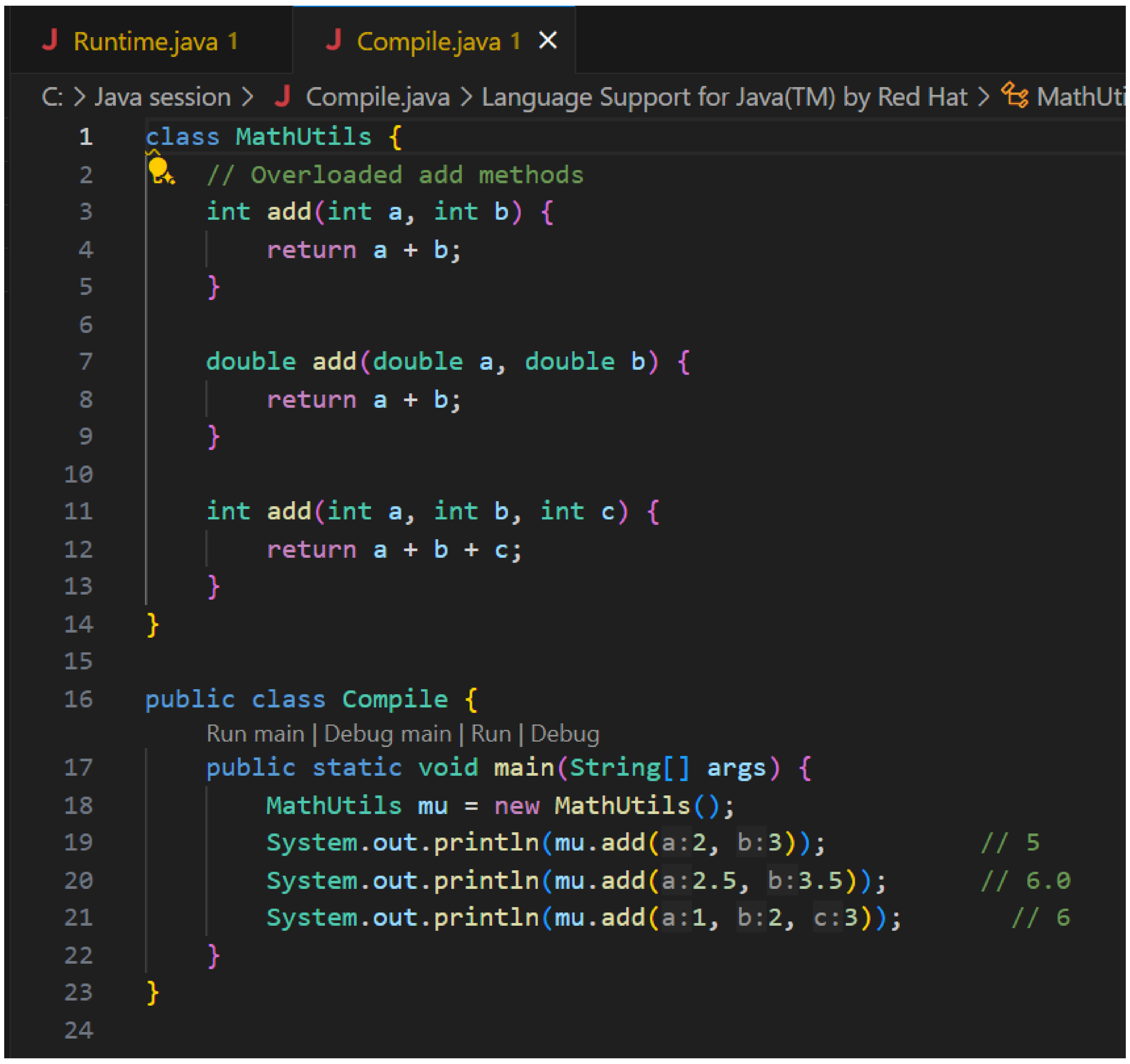

Runtime.java 1
C: > Java session > Runtime.java > Language Support for Java(TM) by Red Hat > Runtime > main(String[])
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter

```java
class Cat extends Animal {
    void sound() {
    }
}

public class Runtime {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Animal a;                // reference of type Animal

        a = new Dog();           // object of Dog
        a.sound();               // Output: Dog barks

        a = new Cat();           // object of Cat
        a.sound();               // Output: Cat meows
    }
}
```

```
[Running] cd "c:\Java session\" && javac Runtime.java && java Runtime
Dog barks
Cat meows

[Done] exited with code=0 in 1.167 seconds
```

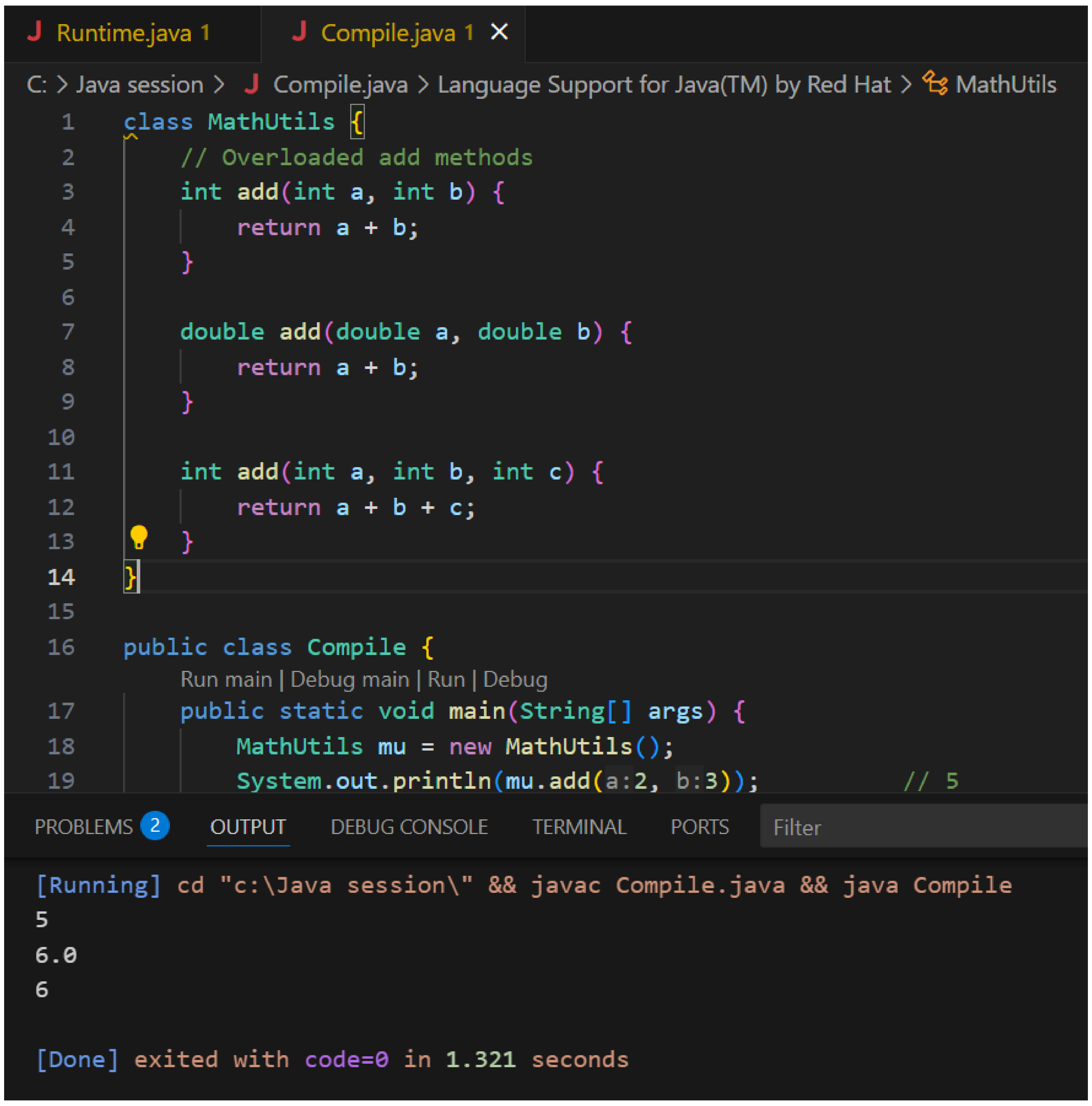
Compile-time (method overloading):

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

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

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

public class Compile {
    public static void main(String[] args) {
        MathUtils mu = new MathUtils();
        System.out.println(mu.add(2, 3));         // 5
        System.out.println(mu.add(2.5, 3.5));     // 6.0
        System.out.println(mu.add(1, 2, 3));      // 6
    }
}
```



Runtime.java 1
Compile.java 1
C: > Java session > Compile.java > Language Support for Java(TM) by Red Hat > MathUti

```java
class MathUtils {
    // Overloaded add methods
    int add(int a, int b) {
        return a + b;
    }

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

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

public class Compile {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        MathUtils mu = new MathUtils();
        System.out.println(mu.add(a:2, b:3));          // 5
        System.out.println(mu.add(a:2.5, b:3.5));      // 6.0
        System.out.println(mu.add(a:1, b:2, c:3));       // 6
    }
}
```



Runtime.java 1
Compile.java 1
C: > Java session > Compile.java > Language Support for Java(TM) by Red Hat > MathUtils
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter

```java
class MathUtils {
    // Overloaded add methods
    int add(int a, int b) {
        return a + b;
    }

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

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

public class Compile {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        MathUtils mu = new MathUtils();
        System.out.println(mu.add(a:2, b:3));          // 5
```

```
[Running] cd "c:\Java session\" && javac Compile.java && java Compile
5
6.0
6

[Done] exited with code=0 in 1.321 seconds
```

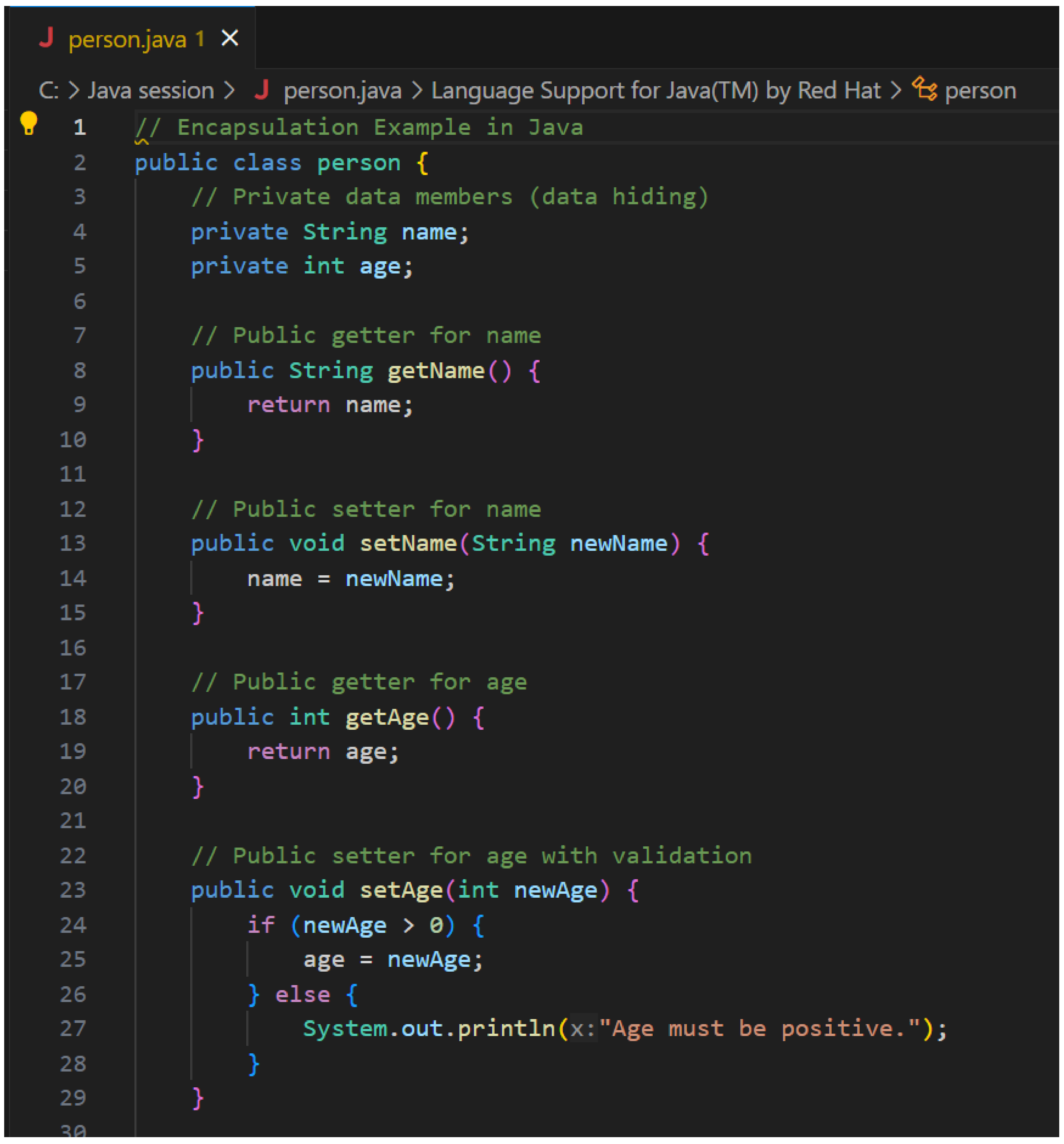
6. Method Overloading vs Overriding

Overloading: Same method name, different parameters (same class)

Overriding: Same method name and parameters in subclass

7. Program for encapsulation

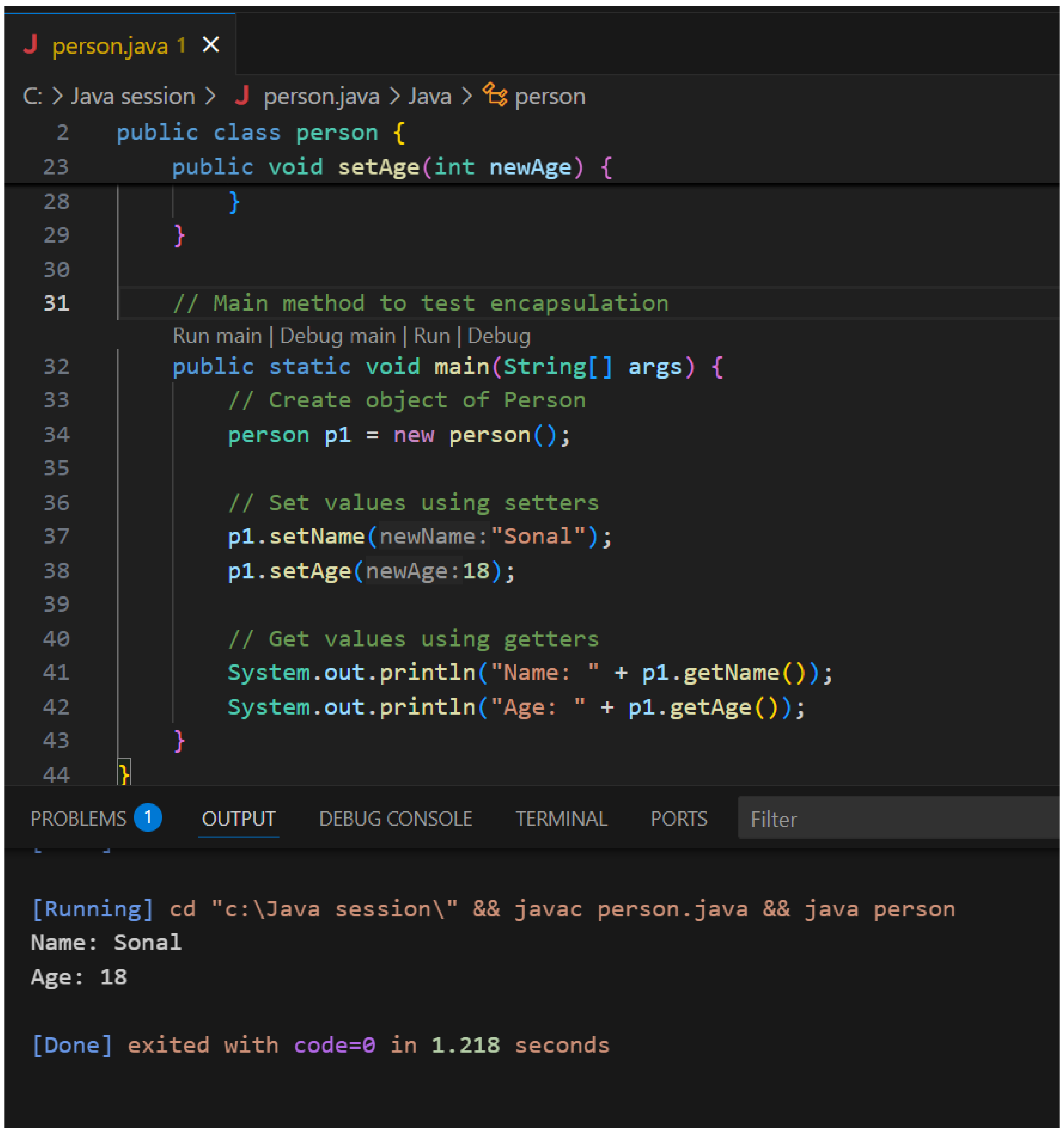
```
public class person {
    private String name;
    private int age;


    public String getName() {
        return name;
    }


    public void setName(String newName) {
        name = newName;
    }


    public int getAge() {
        return age;
    }
    public void setAge(int newAge) {
        if (newAge > 0) {
            age = newAge;
        } else {
            System.out.println("Age must be positive.");
        }
    }


    public static void main(String[] args) {
        person p1 = new person();
        p1.setName("Sonal");
        p1.setAge(18);

        System.out.println("Name: " + p1.getName());
        System.out.println("Age: " + p1.getAge());
    }
}
```



person.java 1 ×

C: > Java session > person.java > Language Support for Java(TM) by Red Hat > person

```java
// Encapsulation Example in Java
public class person {
    // Private data members (data hiding)
    private String name;
    private int age;

    // Public getter for name
    public String getName() {
        return name;
    }

    // Public setter for name
    public void setName(String newName) {
        name = newName;
    }

    // Public getter for age
    public int getAge() {
        return age;
    }

    // Public setter for age with validation
    public void setAge(int newAge) {
        if (newAge > 0) {
            age = newAge;
        } else {
            System.out.println(x:"Age must be positive.");
        }
    }
```


person.java 1
C: > Java session > person.java > Java > person
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Filter
Run main | Debug main | Run | Debug

```java
public class person {
    public void setAge(int newAge) {
        }
    }

    // Main method to test encapsulation
    public static void main(String[] args) {
        // Create object of Person
        person p1 = new person();

        // Set values using setters
        p1.setName(newName:"Sonal");
        p1.setAge(newAge:18);

        // Get values using getters
        System.out.println("Name: " + p1.getName());
        System.out.println("Age: " + p1.getAge());
    }
}
```

```
[Running] cd "c:\Java session\" && javac person.java && java person
Name: Sonal
Age: 18

[Done] exited with code=0 in 1.218 seconds
```

8. What is abstraction?

Abstraction means hiding details and showing only essential features. Achieved using:

- **Abstract class**
- **Interface**

9. Abstract class vs Interface

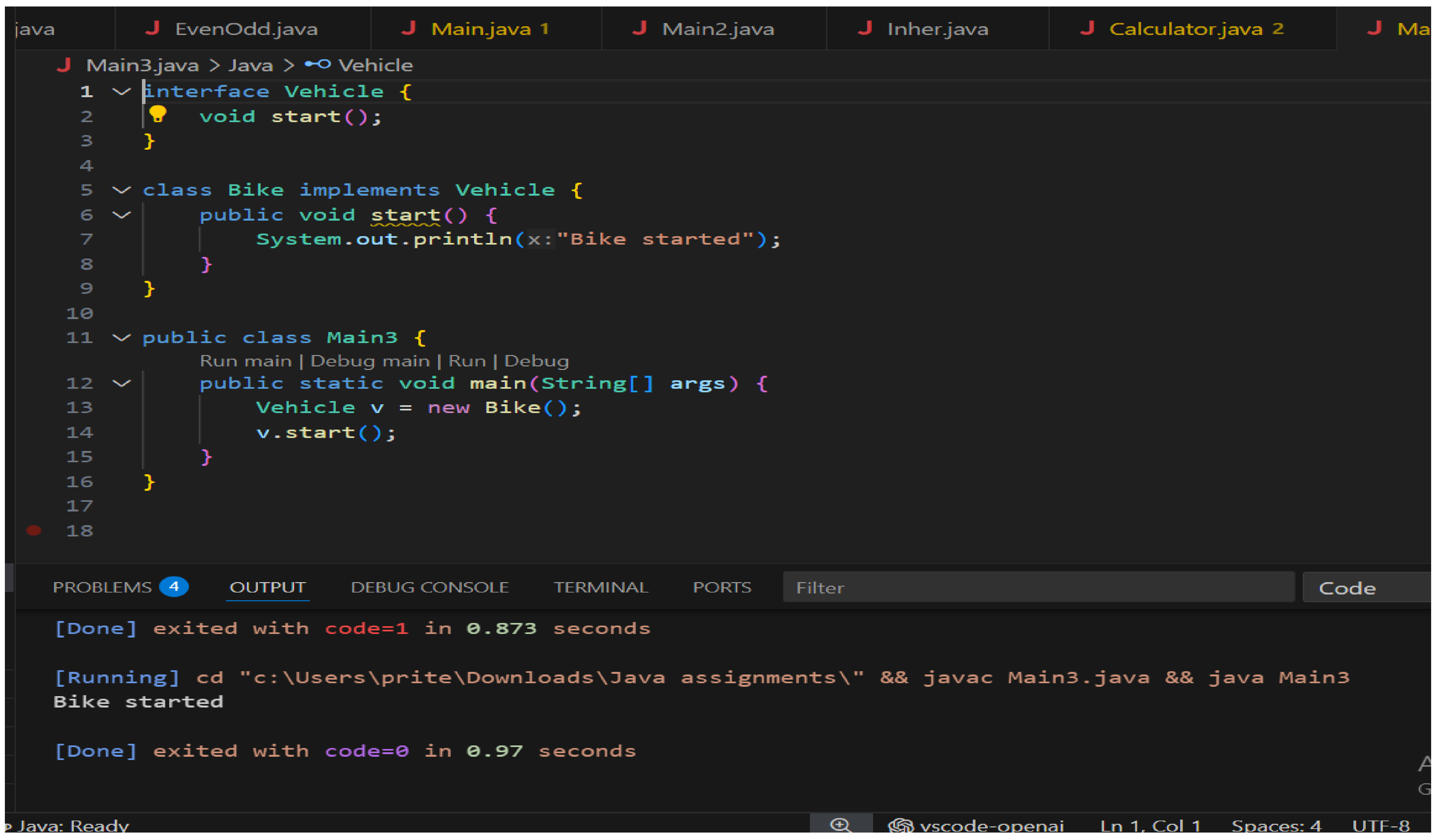
Abstract Class	**Interface**
Can have constructors	Cannot have constructors
Can have both abstract and concrete methods	All methods abstract (Java 7)
Supports inheritance	Supports multiple inheritance

10. Program using Interface

java

CopyEdit

```java
interface Vehicle {
    void start();
}

class Bike implements Vehicle {
    public void start() {
        System.out.println("Bike started");
    }
}

public class Main {
    public static void main(String[] args) {
        Vehicle v = new Bike();
        v.start();
    }
}
```



```java
interface Vehicle {
    void start();
}

class Bike implements Vehicle {
    public void start() {
        System.out.println(x:"Bike started");
    }
}

public class Main3 {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Vehicle v = new Bike();
        v.start();
    }
}
```

```
[Done] exited with code=1 in 0.873 seconds

[Running] cd "c:\Users\prite\Downloads\Java assignments\" && javac Main3.java && java Main3
Bike started

[Done] exited with code=0 in 0.97 seconds
```