

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.

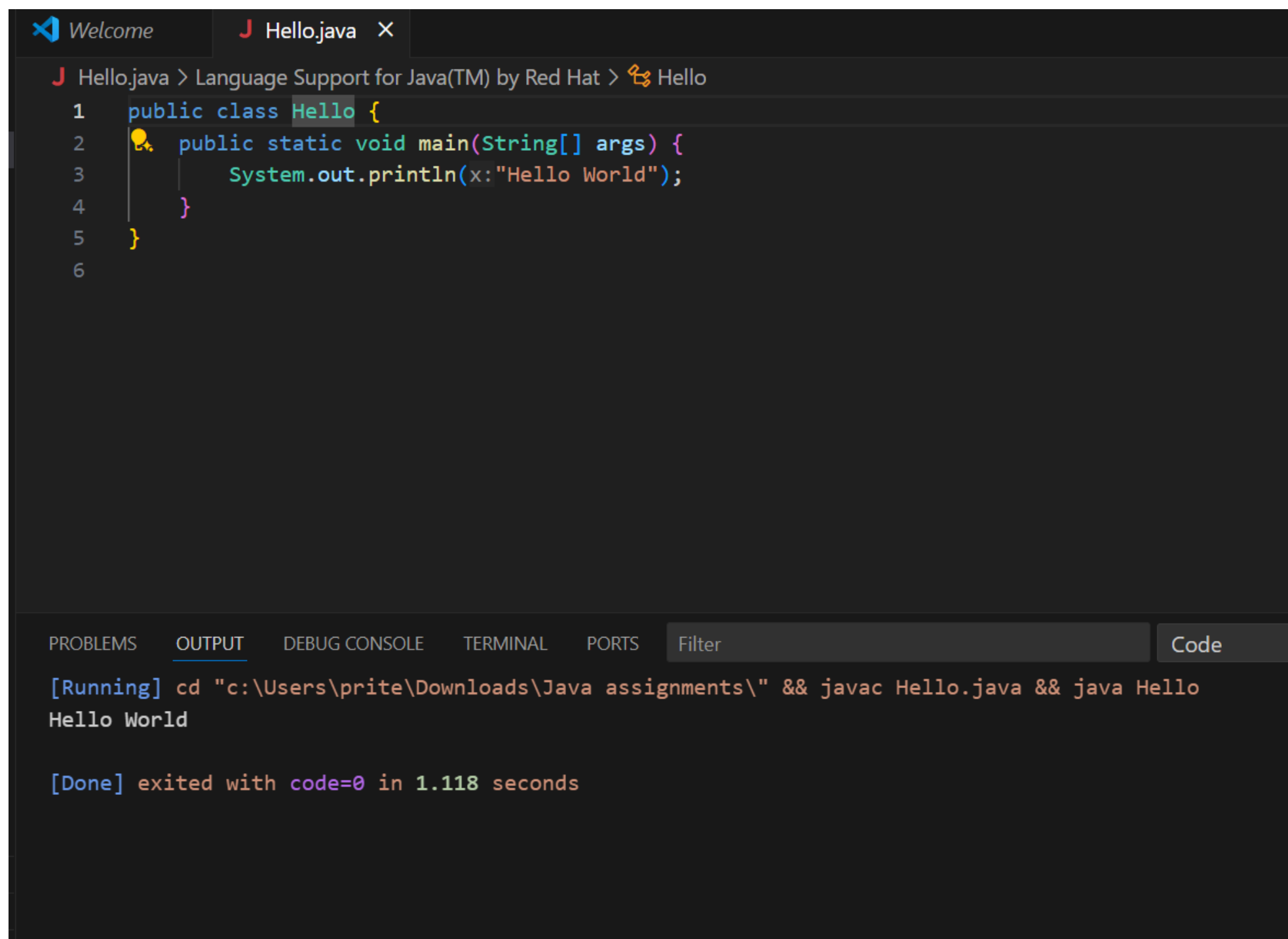
1. Write Java code (.java file)
2. Compile using javac → generates .class bytecode
3. Execute using JVM (java command) → runs on any platform

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");  
    }  
}
```

A screenshot of an IDE window. The top bar shows 'Welcome' and 'Hello.java X'. The editor displays a Java class named 'Hello' with a 'main' method that prints 'Hello World'. The bottom panel shows the 'OUTPUT' tab with the command 'cd "c:\Users\prite\Downloads\Java assignments\" && javac Hello.java && java Hello' and the output 'Hello World'. The status bar indicates '[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");
```

```
}
```

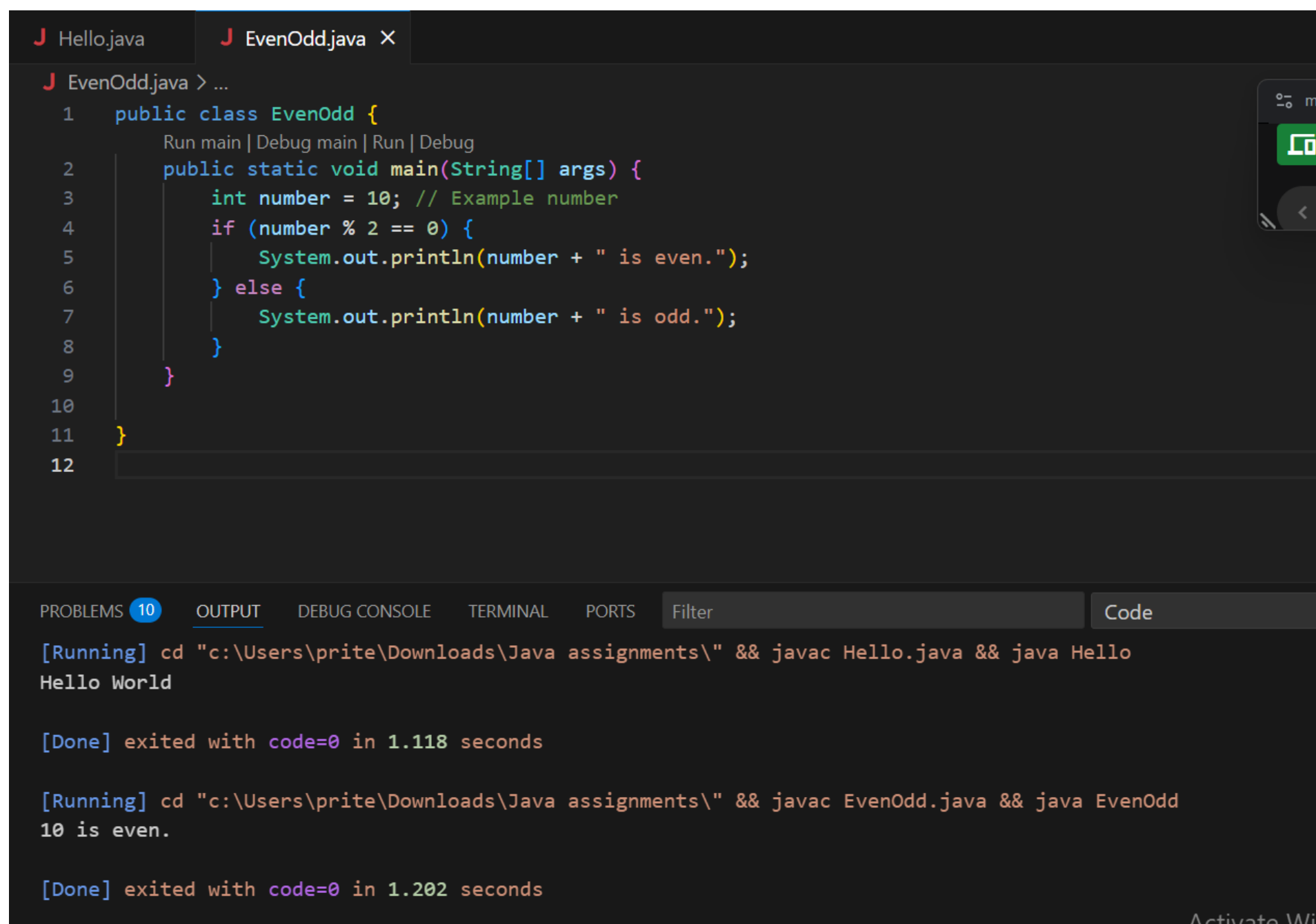
9. Java program to find even or odd number

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");  
  
    }  
  
}
```



The screenshot shows an IDE with two tabs: 'Hello.java' and 'EvenOdd.java'. The 'EvenOdd.java' tab is active, displaying the following code:

```
1 public class EvenOdd {  
2     public static void main(String[] args) {  
3         int number = 10; // Example number  
4         if (number % 2 == 0) {  
5             System.out.println(number + " is even.");  
6         } else {  
7             System.out.println(number + " is odd.");  
8         }  
9     }  
10 }  
11  
12
```

Below the code editor, the 'OUTPUT' tab is selected, showing the execution results:

```
[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

Condition checked first

May never execute

Do-While Loop

Condition checked after execution

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?

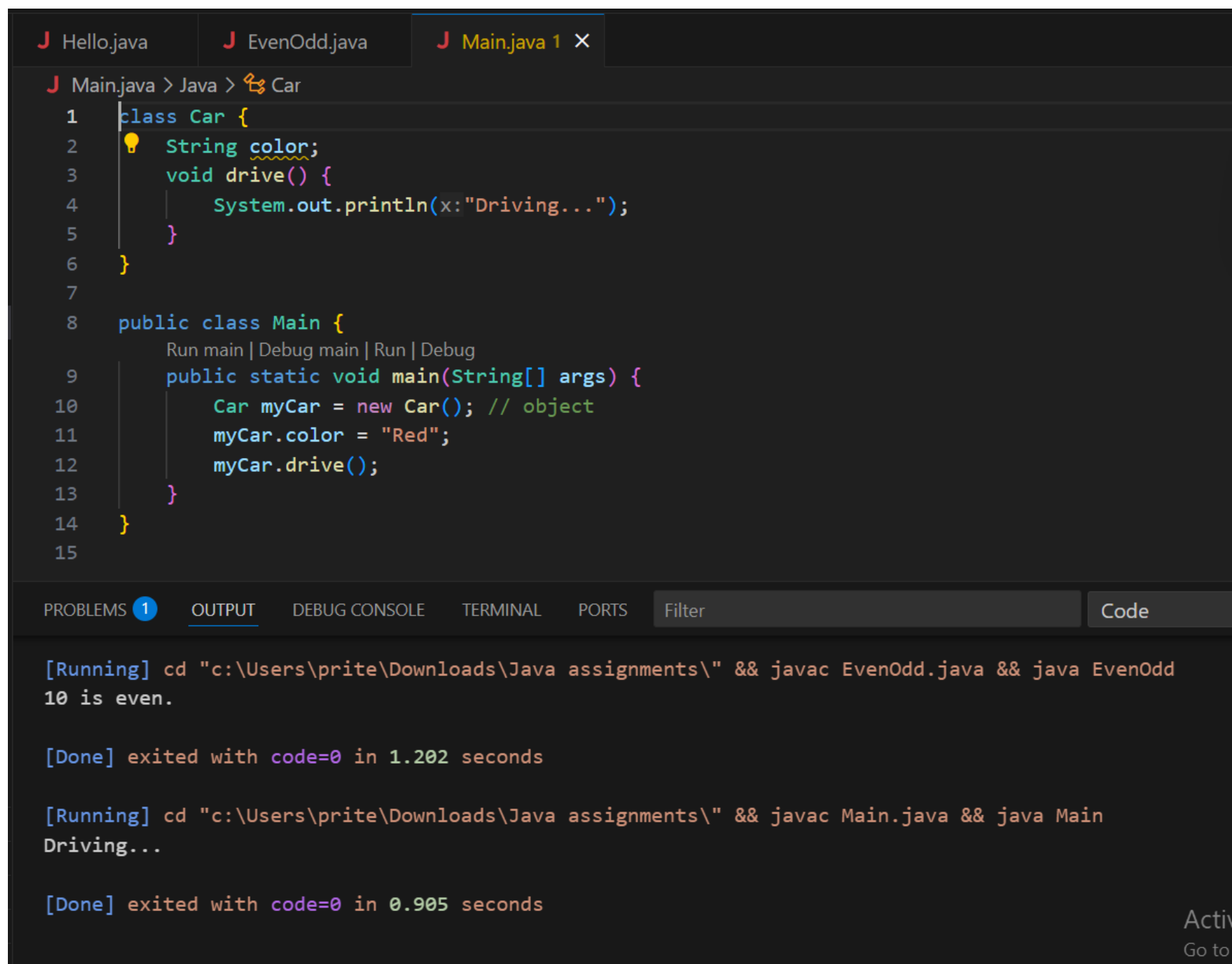
java

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();  
    }  
}
```

```
}
```



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) {
```

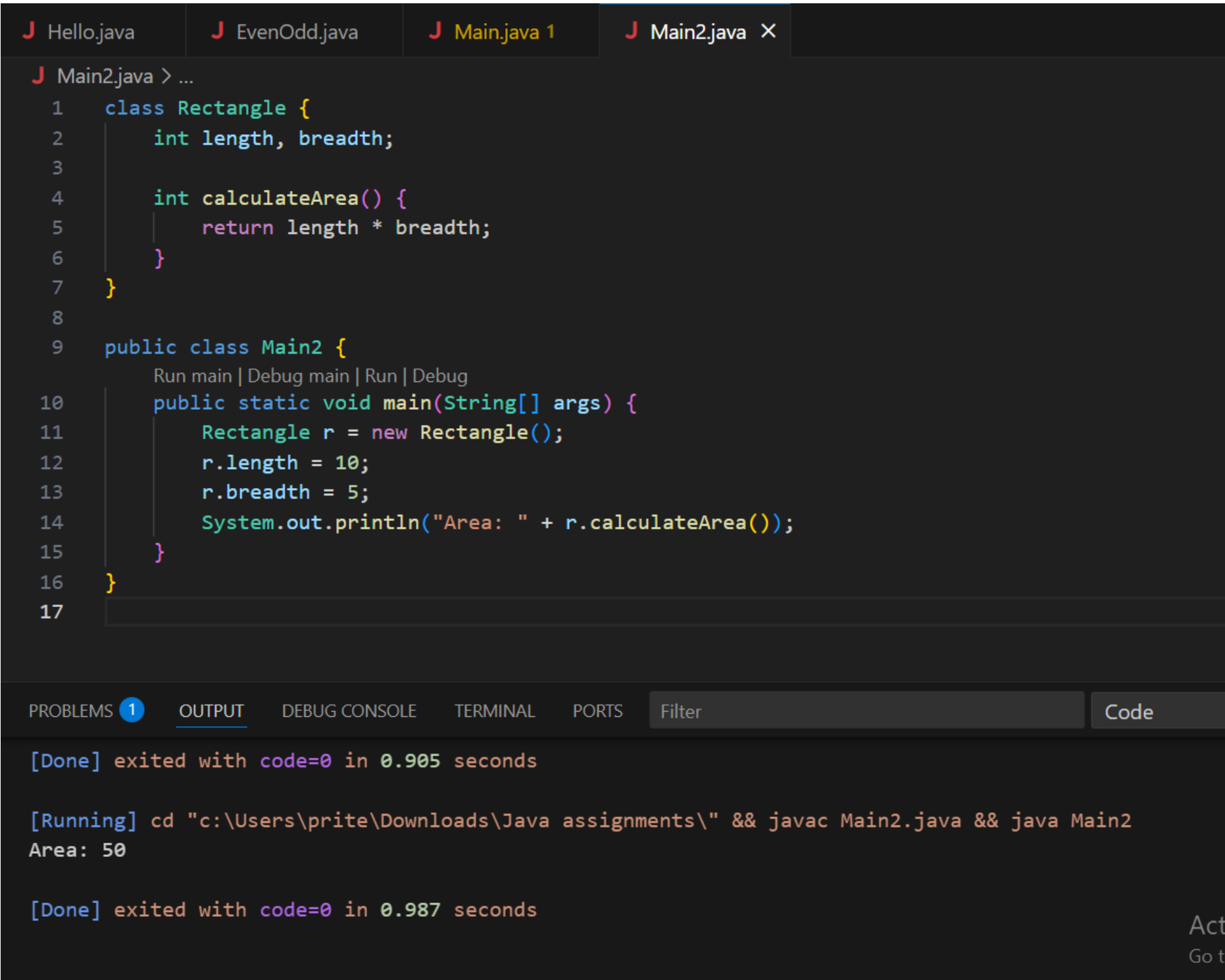
```
        Rectangle r = new Rectangle();
```

```
        r.length = 10;
```

```
        r.breadth = 5;
```

```
        System.out.println("Area: " + r.calculateArea());
```

```
}  
  
}
```



```
J Hello.java J EvenOdd.java J Main.java 1 J Main2.java X  
J Main2.java > ...  
1 class Rectangle {  
2     int length, breadth;  
3  
4     int calculateArea() {  
5         return length * breadth;  
6     }  
7 }  
8  
9 public class Main2 {  
10     Run main | Debug main | Run | Debug  
11     public static void main(String[] args) {  
12         Rectangle r = new Rectangle();  
13         r.length = 10;  
14         r.breadth = 5;  
15         System.out.println("Area: " + r.calculateArea());  
16     }  
17 }  
  
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter Code  
[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  
Act  
Go t
```

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();

    }

}

```

The screenshot shows an IDE with several tabs: Hello.java, EvenOdd.java, Main.java 1, Main2.java, and Inher.java. The Inher.java tab is active, showing the following code:

```

1  class Animal {
2      void eat() {
3          System.out.println(x:"This animal eats food.");
4      }
5  }
6
7  class Dog extends Animal {
8      void bark() {
9          System.out.println(x:"Dog barks");
10     }
11 }
12
13 public class Inher {
14     public static void main(String[] args) {
15         Dog d = new Dog();
16         d.eat();
17         d.bark();
18     }
19 }

```

Below the code editor, the OUTPUT tab is selected, showing the execution results:

```

[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 X
C: > Java session > Runtime.java > Language Support for Java(TM) by Red Hat > Animal
1  class Animal {
2      void sound() {
3          System.out.println(x:"Animal makes a sound");
4      }
5  }
6
7  class Dog extends Animal {
8      @Override
9      void sound() {
10         System.out.println(x:"Dog barks");
11     }
12 }
13
14 class Cat extends Animal {
15     @Override
16     void sound() {
17         System.out.println(x:"Cat meows");
18     }
19 }
20
21 public class Runtime {
22     Run main | Debug main | Run | Debug
23     public static void main(String[] args) {
24         Animal a;           // reference of type Animal
25
26         a = new Dog();       // object of Dog
27         a.sound();           // Output: Dog barks
28
29         a = new Cat();       // object of Cat
30         a.sound();           // Output: Cat meows
31     }
32 }
33
```

```
Runtime.java 1 X
C: > Java session > Runtime.java > Language Support for Java(TM) by Red Hat > Runtime > main(String[])
14  class Cat extends Animal {
16      void sound() {
18      }
19  }
20
21  public class Runtime {
22      Run main | Debug main | Run | Debug
23      public static void main(String[] args) {
24          Animal a;           // reference of type Animal
25
26          a = new Dog();       // object of Dog
27          a.sound();           // Output: Dog barks
28
29          a = new Cat();       // object of Cat
30          a.sound();           // Output: Cat meows
31      }
32  }
33
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter

```
[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 1Compile.java 1 X

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

```
1 class MathUtils {
2     // Overloaded add methods
3     int add(int a, int b) {
4         return a + b;
5     }
6
7     double add(double a, double b) {
8         return a + b;
9     }
10
11     int add(int a, int b, int c) {
12         return a + b + c;
13     }
14 }
15
16 public class Compile {
17     Run main | Debug main | Run | Debug
18     public static void main(String[] args) {
19         MathUtils mu = new MathUtils();
20         System.out.println(mu.add(a:2, b:3));           // 5
21         System.out.println(mu.add(a:2.5, b:3.5));       // 6.0
22         System.out.println(mu.add(a:1, b:2, c:3));      // 6
23     }
24 }
```

Runtime.java 1Compile.java 1 X

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

```
1 class MathUtils {
2     // Overloaded add methods
3     int add(int a, int b) {
4         return a + b;
5     }
6
7     double add(double a, double b) {
8         return a + b;
9     }
10
11     int add(int a, int b, int c) {
12         return a + b + c;
13     }
14 }
15
16 public class Compile {
17     Run main | Debug main | Run | Debug
18     public static void main(String[] args) {
19         MathUtils mu = new MathUtils();
20         System.out.println(mu.add(a:2, b:3));           // 5
21     }
22 }
```

PROBLEMS 2OUTPUTDEBUG CONSOLETERMINALPORTSFilter

[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);
```

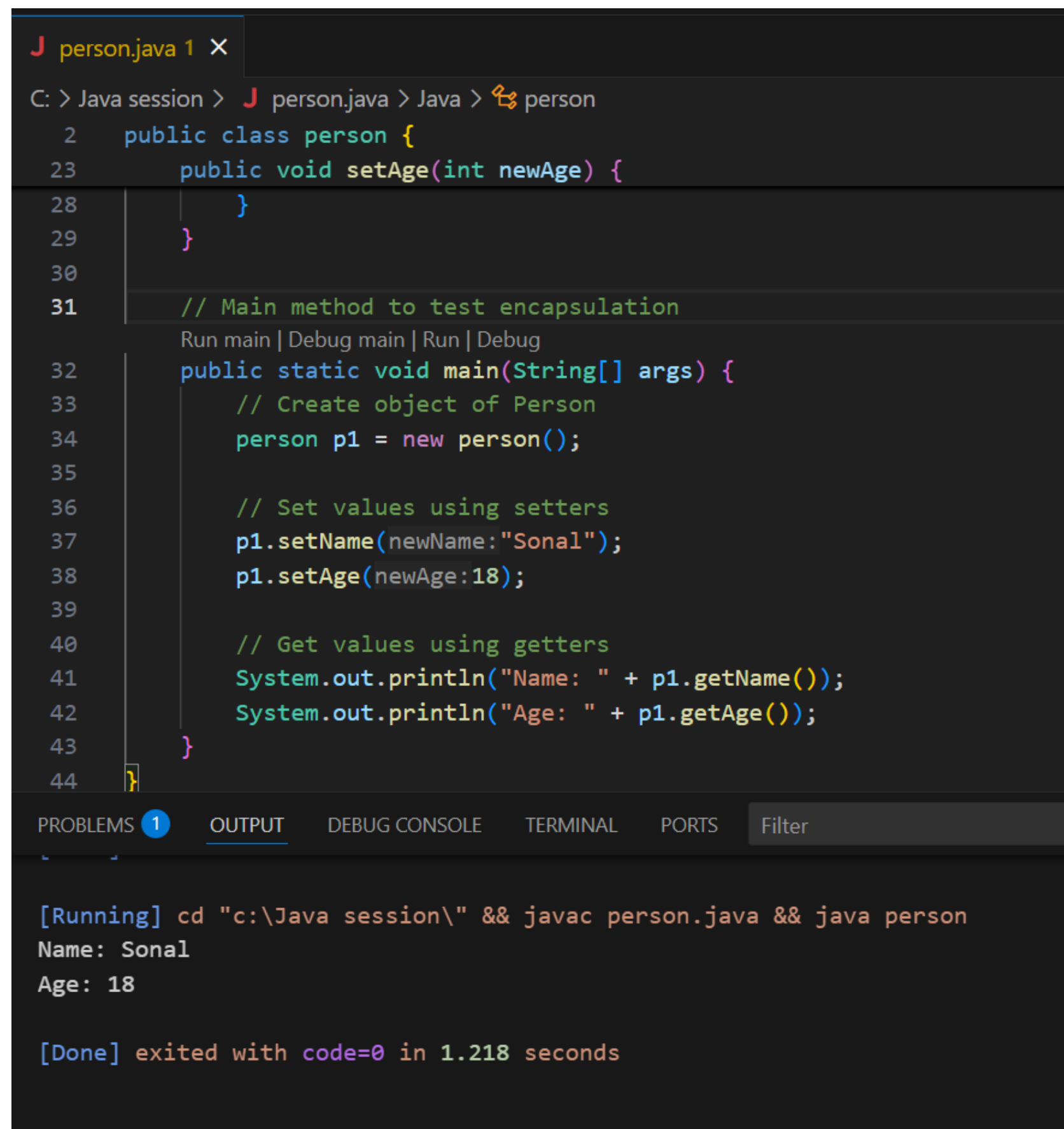
```
System.out.println("Name: " + p1.getName());
```

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

```
}
```

```
}
```

```
person.java 1 X
C: > Java session > person.java > Language Support for Java(TM) by Red Hat > person
1 // Encapsulation Example in Java
2 public class person {
3     // Private data members (data hiding)
4     private String name;
5     private int age;
6
7     // Public getter for name
8     public String getName() {
9         return name;
10    }
11
12    // Public setter for name
13    public void setName(String newName) {
14        name = newName;
15    }
16
17    // Public getter for age
18    public int getAge() {
19        return age;
20    }
21
22    // Public setter for age with validation
23    public void setAge(int newAge) {
24        if (newAge > 0) {
25            age = newAge;
26        } else {
27            System.out.println("Age must be positive.");
28        }
29    }
30 }
```



```
J person.java 1 X
C: > Java session > J person.java > Java > person
2 public class person {
23 public void setAge(int newAge) {
28     }
29 }
30
31 // Main method to test encapsulation
Run main | Debug main | Run | Debug
32 public static void main(String[] args) {
33     // Create object of Person
34     person p1 = new person();
35
36     // Set values using setters
37     p1.setName(newName:"Sonal");
38     p1.setAge(newAge:18);
39
40     // Get values using getters
41     System.out.println("Name: " + p1.getName());
42     System.out.println("Age: " + p1.getAge());
43 }
44 }

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter

[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

Can have constructors

Can have both abstract and concrete methods

Supports inheritance

Interface

Cannot have constructors

All methods abstract (Java 7)

Supports multiple inheritance

10. Program using Interface

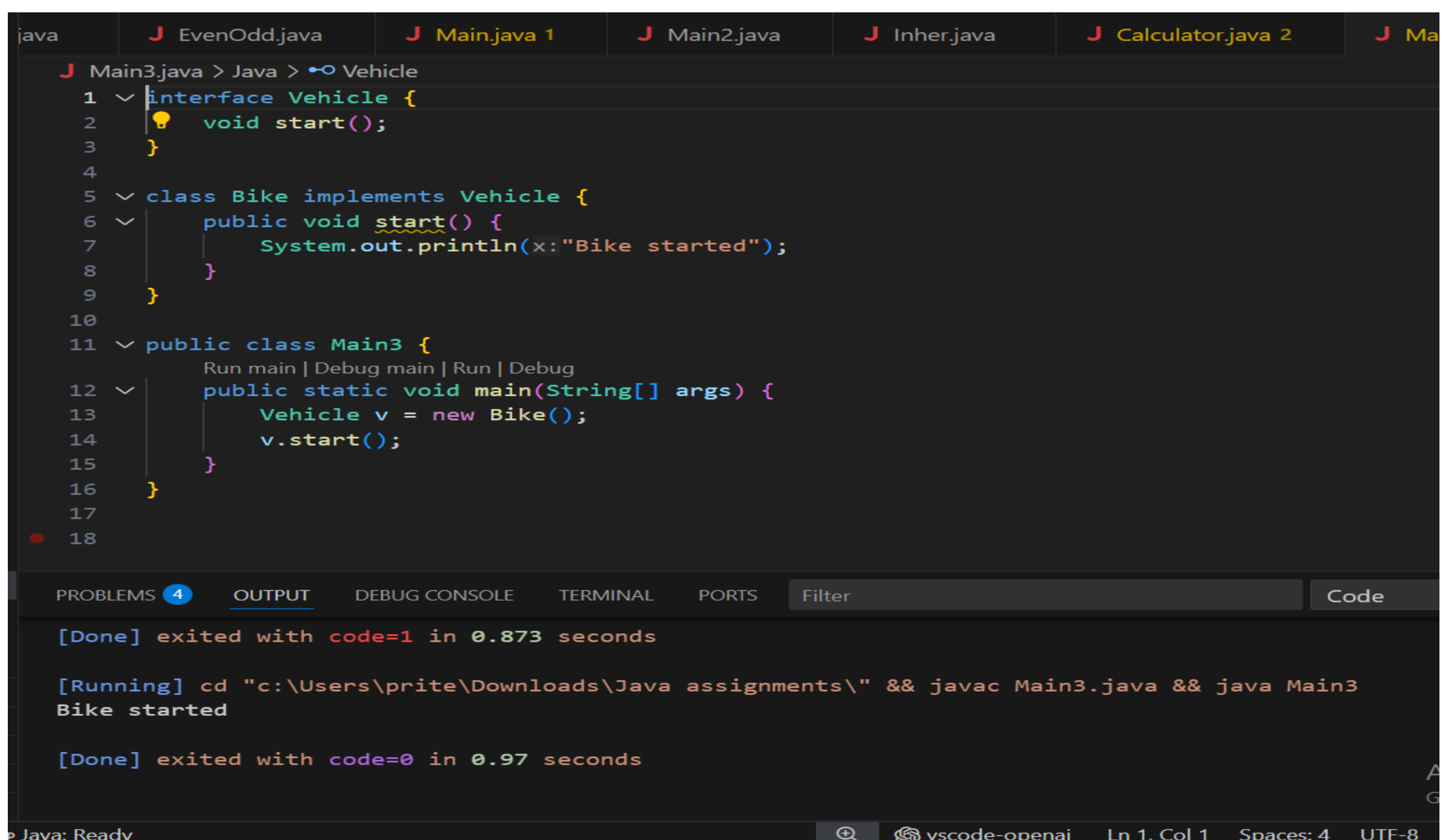
java

CopyEdit

```
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();  
    }  
}
```



The screenshot shows a VS Code editor with a Java project. The file explorer on the left shows a package 'Java' containing a file 'Main3.java'. The editor displays the code from the previous blocks. The output console at the bottom shows the execution of the program, which prints 'Bike started'.

```
java | EvenOdd.java | Main.java 1 | Main2.java | Inher.java | Calculator.java 2 | Ma  
Main3.java > Java > Vehicle  
1 interface Vehicle {  
2     void start();  
3 }  
4  
5 class Bike implements Vehicle {  
6     public void start() {  
7         System.out.println("Bike started");  
8     }  
9 }  
10  
11 public class Main3 {  
12     Run main | Debug main | Run | Debug  
13     public static void main(String[] args) {  
14         Vehicle v = new Bike();  
15         v.start();  
16     }  
17 }  
18  
PROBLEMS 4 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | Filter | Code  
[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  
Java: Ready | vscode-openai | Ln 1, Col 1 | Spaces: 4 | UTF-8
```