MCQ's

1) What is Java?
   a. Machine Language
   b. Binary Language
   c. Assembly Language
   d. **High-Level Language**

2) The process of conversion of a high-level programming language to a machine language is called:
   a. Transpiling
   b. **Compilation**
   c. Interpretation
   d. Conversion

3) Is JVM platform-independent?
   a. Yes
   b. **No**

4) Which of the following components helps Java achieve high performance?
   a. JVM (Java Virtual Machine)
   b. JDK (Java Development Kit)
   c. **JIT (Java in Time)**
   d. JRE (Java Runtime Environment)

5) Can there be a Java Program without a main method?
   a. Yes
   b. **No**

6) Which command in the JDK helps us covert .java files to .class files?
   a. java
   b. **javac**
   c. javapackager
   d. jamf

7) Which of the following data type has the smallest range in Java?
   Byte

8) Is Java a pure OOP language?
   No, a pure OOP language consists of objects in them. In Java the primitive data types won't allow them to be pure.

8) Primitive data types are started with small letters that is int, long, char, boolean.

9) Non primitive data type will start with capital letters that is String.

10) **Casting**: refers to passing lower size data type like int into higher size data type like double.

11) **Narrowing**: refers to passing higher size data type like double into lower size data type like int.

12) **Compilation error**: occurs when codes are written in ways in which the Java compiler does not recognise. Compiler displays error to alert you about something that will not compile and therefore cannot be running.

13) **Runtime error**: occurs when code is running. Most difficult as they lead to unexpected program crashes which can be hard to track down and fix.

```
int arr[] = {11,10,5};
//      Compilation Error
        System.out.println(ar[1]);
//      Runtime Error
        System.out.println(arr[10]);
```

14) Where does the execution of any Java program begin?

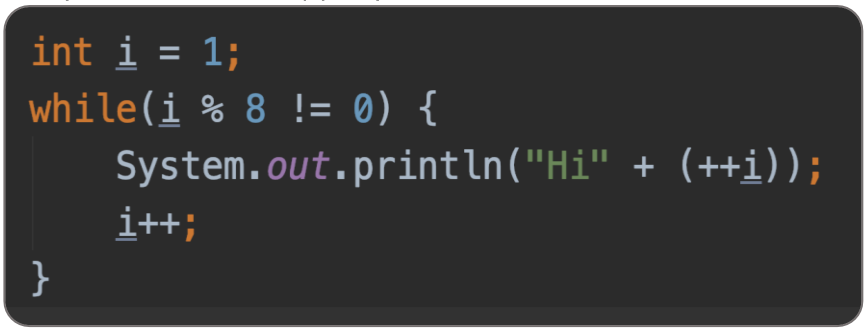
The main method!

15) After the main method starts executing, how does Java determine which lines of code should be executed?
Usually, it would try to execute one statement after another, from the first line of code right down to the last one in the main method.

16) What is an acceptable data type for a conditional expression?

1. boolean
2. int
3. char
4. All of the above

17) How many lines will this snippet print?



```
int i = 1;
while(i % 8 != 0) {
    System.out.println("Hi" + (++i));
    i++;
}
```

a. 1
b. 8
c. **infinite**
d. 0

18) One thing you have to keep in mind that, in *for* loop, we can omit any of the 3 conditions. For example, we can write *for* loop as shown below-

```
int counter = initialValue;
for ( ; counter < endValue; ) {
        Action;
        counterValue += increment;
}
```

In the above example, we omitted both initialization and increment condition in *for* loop, but it is still valid.

19) Below *for* loop is an infinite loop, but it's still valid. One thing to note here is, even if you omit any condition, you always have to put the semicolons.

```
for ( ; ; ) {
   Action;
}
```

20) Which of the following is the correct way to define a method inside a class?

1. changeNameTo(newName: String) : void;
2. void changeNameTo(String newName);
3. **void changeNameTo(String newName) { … }**
4. void changeNameTo(String newName) { … };

21)What will happen when you try to compile a program that contains a statement that throws a checked exception, and you have no provision to handle the exception?

Ans: The program will not compile. Checked exceptions need to be specified by throws keyword or handled by try-catch block for the program to compile.

22)What will happen when you try to compile a program that contains a statement that throws a checked exception, and you have no provision to handle the exception?

Ans: The program will compile but may not be executed as expected due to the unhandled exception. The program will compile but may not be executed as expected due to the unhandled exception.

23) Which of these entities does not use up space in the JVM?

1. Variable
2. **Class**
3. Object
4. None of the above

24) Which of these are types of variables offered by Java?

1. **Local variables**
2. **Class variables**
3. **Static variables**
4. **Instance variables**

25) How does No-Args Constructor manage to create objects for them?

- When no constructors are written in the code, Java creates an empty constructor with no parameters as part of the compilation!
- If we define any constructor, Java does not provide this 'default' constructor.

26) Which of the following statements is true regarding the 'this' keyword?

1. **'this' refers to the current object.**
2. 'this' refers to the current class.
3. 'this' can be used inside methods.
4. **'this' can be used to call a constructor from another constructor.**

27) Where would Java store the data inside an array?

1. Stack memory
2. **Heap memory**
3. Cache memory
4. Random Access Memory (RAM)

28) Where would Java store an attribute of a class, which has an “int” data type?
1. Stack memory
2. **Heap memory**
3. Cache memory
4. RAM

29) Q. What does Java provide us with to hide, or reduce, the accessibility of entities?
Access Modifiers!

30) Which of these entities are abstract?
- A smartphone – Yes
- An Apple smartphone – Yes
- An iPhone – Yes
- An iPhone 12 – No
- An iPhone 12 Pro – No

31) Which of these can be modelled as a parent and child class?
1. Boat and Ship
2. **Dog and Husky**
3. Pizza and Toppings
4. ElectricCar and PetrolCar

32) What can we write to make class A the parent of class B?
- class B extends class A {}
- **class B extends A {}**
- class A extends class B {}
- class A extends B {}

33) Which of the following statements is true regarding inheritance in Java?
1. A class inherits only attributes and methods from its direct superclass.
2. You cannot instantiate parent classes.
3. **You can provide new attributes and methods in a child class.**
4. None of the above

34) Which of the following can the ‘super’ keyword helps us do?
1. Invoke data members of the parent class
2. Invoke methods of the parent class
3. Invoke constructors of the parent class
4. **All of the above**

35) Overloaded methods must have ____.
Note: More than one option can be correct.
1. **Same method names**
2. Different method names
3. Same parameter lists
4. **Different parameter lists**

36) Which of the following is true with respect to final methods?
1. They can only be present in a final class.
2. They cannot be overloaded.
3. **They cannot be overridden.**
4. They cannot be public.

37) Mention Some Java feature?

- Platform-independent
- In Java, class files are loaded at runtime. So, Java is dynamic.
- Architecture-neutral. In simple terms, architecture-neutral means that a Java program can be run on any processor irrespective of its architecture.

2. Java does not use Pointers.

3. Java Development Kit: provides an environment to develop and execute Java programs.

4. Java Runtime Environment is an installation package that provides an environment to run only Java programs.

5. Java Virtual Machine is responsible for executing Java programs line-by-line by converting Java bytecode into machine language.

6. Just-In-Time Compiler is responsible for the performance optimisation of Java-based applications.

7. You can run as many JVMs on your laptop as you want, provided it has sufficient memory.

8. The Java compiler converts the code into an intermediate bytecode. Then, JVM converts this bytecode into the machine code and runs it.

9. You would have noticed that you get operating-system-specific download options for Java. This shows that these (JDK, JVM, JRE) are platform-dependent.
10. Explain the main method line:

- The main() method is the starting point for JVM to start the execution of a Java program. It has the following signature:
- **public**: It is the access modifier. The main() method should be public so that Java can execute this method at runtime.
- **static**: Initially, at runtime, no object of the Main class is present. The main() method must be static so that JVM can load the Main class into the memory and call the method.
- void: The main() method does not return anything and, after its execution, the program is terminated. Hence, the return type is 'void'.
- **main**: It is the name of the method, which is fixed and, upon execution, Java starts to look for this method.
- **String[] args**: The Java main() method also needs a single-line argument of a String array, also known as the Java command-line argument.

11. You can have multiple 'main' methods in a class. However, you need to make sure that each of them has a different signature. Only the method with the following signature will be the entry point for that class:

public static void main(String[] args)

Here is a code snippet to help you visualise better.

```
class Main {
  public static void main (String args[]) {
    System.out.println("First main class");
  }
  public static void main (int i) {
    System.out.println("Second main class");
  }
}
```

12. **Pro Tip**: An experienced Java Developer will always tend to run Java applications in debug mode. debug mode = run + breakpoints. Breakpoints can be easily muted and unmuted in IntelliJ.

13. A high-level language, like Java, is converted into instructions of Assembly language, which, in turn, are converted into 0's and 1's. These are finally fed to the hardware.

14. Some statements about identifiers is not true?

- Identifiers do not have any special symbols or punctuation.

```
class Source {
      public static void main(String args[]) {
         Scanner input = new Scanner(System.in);
         int first_Number= input.nextInt();
        System.out.println("The first number is: " +first_Number);
   }
  }
```

- Identifiers define different programming elements such as variables, integers, structures and unions. Identifiers define various programming elements such as variables, integers, structures and unions.

```
class Source {
      public static void main(String args[]) {
         Scanner input = new Scanner(System.in);
         int firstNumber= input.nextInt();
         System.out.println("The first number is: " +firstNumber);
      }
}
```

In the given piece of code, Identifiers are Source, String, main, args, Scanner, input, num, square, out, println, nextInt, etc.

- An identifier contains only alphabetical characters. Identifiers can have alphabetical characters, digits and special characters (underscores). For example, firstNumber1 is a valid identifier.

```
class Source {
     public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        int firstNumber1= input.nextInt();
        System.out.println("The first number is: " +firstNumber1);
     }
  }
```

15. Which of the following are the key differences between a variable and an identifier?

- Every variable is an identifier, but every identifier need not be a variable. Every variable is an identifier, although identifiers can also be of other types, such as function names and class names.
- An identifier identifies an entity when a program is getting executed, whereas a variable is a name given to a memory location, which holds a value.
- Variables should start with either underscore (_), dollar ($) or alphabets.

16. Here is how the widening casting flow looks:
byte -> short -> char -> int -> long -> float -> double
17. Here is how the narrowing casting flow looks:
double -> float -> long -> int -> char -> short -> byte

```
int x = 3;
int a = x++;
System.out.print(x); //4
System.out.print(a); //3

int b = ++x;
System.out.print(x); //5
System.out.print(b); //5
```

For example, let's take x=3
int x = 3;
int a = x++;
Here, a=3 because the value of x until now is 3.
Once a=3, then the value of x is incremented. So, now, the value of x is 4.
So, until now, a=3, x=4.

int b = ++x;
Since this is a prefix operator, x is incremented first. So, x is 5, and then, the value of x is assigned to b. So, b is 5.

So, the printed values will be as follows:
4 3 5 5

- **Post-Increment:** Value is first used for computing the result and then incremented.
- **Pre-Increment:** Value is incremented first and then the result is computed.

18) Can you define attributes and methods outside a class in Java? Choose the correct answer from the options given below.

No, Java is a rigid class-based object-oriented programming language, which means that you cannot define any attributes or methods outside a class. Unlike Java, there are some programming languages, such as C++, that allow you to provide attributes and methods outside a class. Thus, this is the correct option.

19) Suppose your friend wants you to create a function that takes a string as input and then replaces the string with the upper case version.

```
public class Loops {
   public static void main(String[] args) {
      String string1 = "Welcome";
      uppercase(string1);
      System.out.println(string1);       //Welcome
   }
   public static String uppercase(String string1) {
      string1.toUpperCase();
      return string1;
   }
   }
```

Java uses pass by value, and thus, although the string inside the function would have been converted to uppercase, it would not be reflected in the original string. Hence, the original string remains as it is.

20) A class is a logical blueprint of an object that describes all the methods and data that are provided to an object of a specific class. Class, in general, describes all the properties and behaviour of an object. Hence, this is the correct option.

21) Which of the following access modifiers allows a class to be accessible across the same package, but restricts its access outside the package only to its subclasses?

Protected, Correct! By using this access modifier, we can access it by its subclass even outside the package.

22) Given below is a class object with some entities. Which of the following have the strictest access scope?

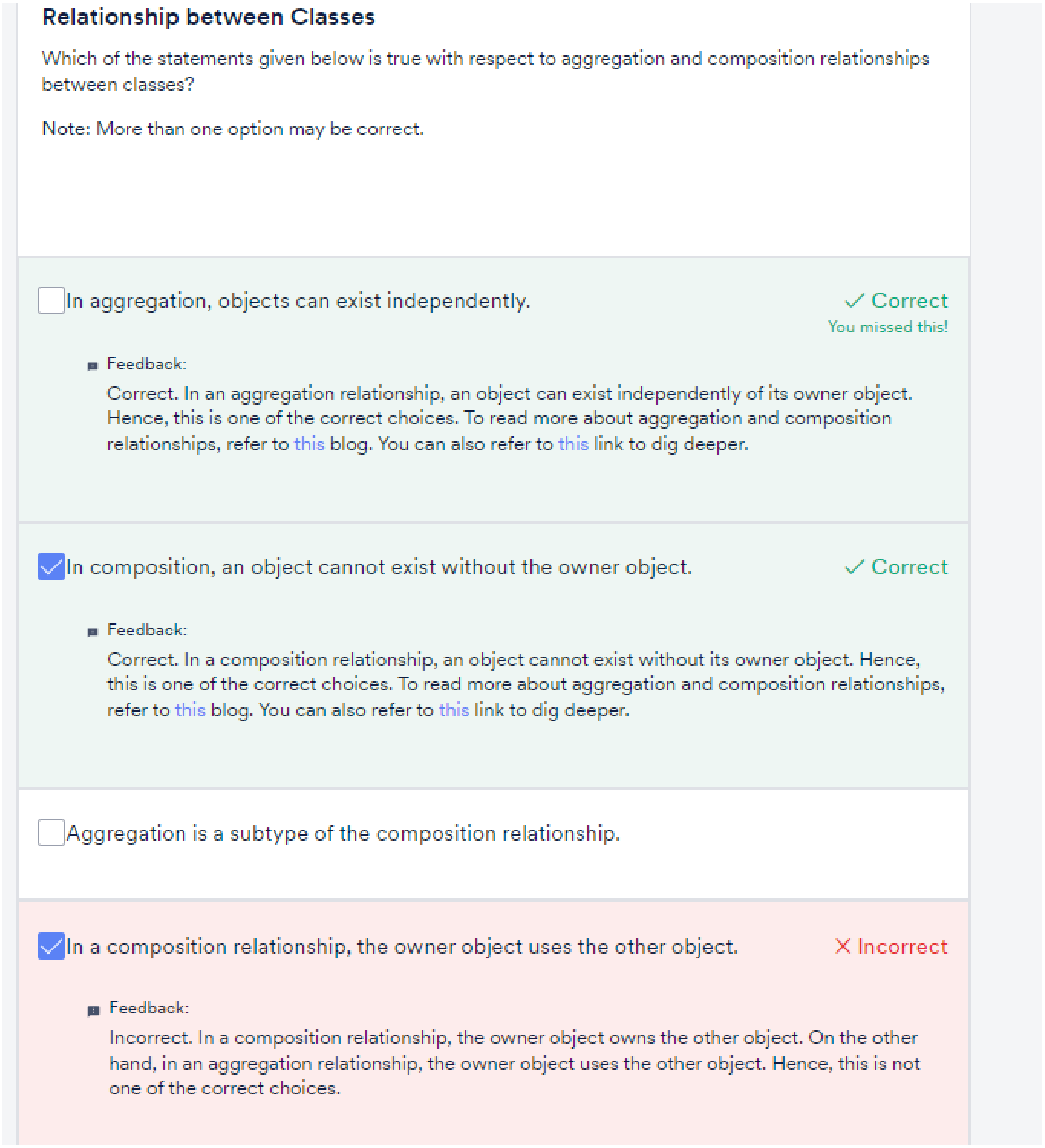
```
class Weapon {
   public String weaponName;
   public String weaponType;

   private int bulletsInRound;
   private double reloadTime;

   int damagePerBullet;
   String ownerName;
}
```

Solution: bulletsInRound, reloadTime.

Properties with 'private' scope have the strictest access. The order of the access modifiers in order of decreasing restrictive access is given as: public>protected>default>private

23) Can we convert a constructor into a method?

Shouldn't we worry about the reference trap for String?
Nope. When you assign the variable to a string, it equals a brand-new string object.
In other words, the reference trap doesn't apply, so you don't have to worry.



Relationship between Classes

Which of the statements given below is true with respect to aggregation and composition relationships between classes?

Note: More than one option may be correct.

☐ In aggregation, objects can exist independently. ✓ Correct
You missed this!

Feedback:
Correct. In an aggregation relationship, an object can exist independently of its owner object. Hence, this is one of the correct choices. To read more about aggregation and composition relationships, refer to this blog. You can also refer to this link to dig deeper.

[✓ In composition, an object cannot exist without the owner object.] ✓ Correct

Feedback:
Correct. In a composition relationship, an object cannot exist without its owner object. Hence, this is one of the correct choices. To read more about aggregation and composition relationships, refer to this blog. You can also refer to this link to dig deeper.

☐ Aggregation is a subtype of the composition relationship.

[✓ In a composition relationship, the owner object uses the other object.] ✗ Incorrect

Feedback:
Incorrect. In a composition relationship, the owner object owns the other object. On the other hand, in an aggregation relationship, the owner object uses the other object. Hence, this is not one of the correct choices.

Which of the following is true about encapsulation?

Note: More than one option may be correct.

[✓ Provides classes for problem-domain entities] ✕ Incorrect

Feedback:

Incorrect. When you provide classes for problem-domain entities, it is referred to as 'abstraction'. Hence, this option is incorrect.

To read more about abstraction, refer to this article.

☐ Hides the internal details of the class ✓ Correct

You missed this!

Feedback:

Correct. Encapsulation is all about hiding the internal details of an object and exposing a simple interface to work with that object. Thus, this is one of the correct choices.

To read more about encapsulation, refer to this blog.

[✓ Focuses on the observable behaviour of an object] ✕ Incorrect

Feedback:

Incorrect. It is 'abstraction' that focuses on the observable behaviour of the object. Hence, this option is incorrect. To read more about abstraction, refer to this article.

☐ Focuses on the implementation that results in the observable behaviour ✓ Correct

You missed this!

Feedback:

Correct. Encapsulation is all about hiding the internal details of an object and exposing a simple interface to work with that object. Abstraction focuses on the observable behaviour of an object, whereas encapsulation focuses on the implementation that gives rise to the observable behaviour. Thus, this is one of the correct choices.

Analyse the following code:

```java
public class User {
   private int id;
   public String name;

   User(int id, String name) {
       this.id = id;
       this.name = name;
   }

   public String getName() {
       return name;
   }
}
```

Which of the following statements is correct with respect to the above code?

Note: More than one option may be correct.

☐ You can only declare variables of User type inside the same package.

☐ The id attribute can be accessed by any class.

☑ The User class can be instantiated only in the same package. ✓ Correct

- Feedback:
  Correct. As there is no access modifier for the User constructor, it will have the package-private access modifier. Thus, the User class can only be instantiated inside the same package. Hence, this is one of the correct choices. To read more about the scope of different access modifiers in Java, refer to this blog.

☑ The name attribute can be accessed by any class. ✓ Correct

- Feedback:
  Correct. As the name attribute is declared public, it can be accessed anywhere in your code. Hence, this is one of the correct choices. To read more about the scope of different access modifiers in Java, refer to this blog.

What is the output of the following code?

```java
public class Main {
  public static void main(String[] args) {   //line 1
    TestClass c1 = new TestClass(2, 5.2);   //line 2
    TestClass c2 = new TestClass(3, 7.5);   //line 3
     System.out.println(c1.a + ", " + c1.b);    //line 4
  }
}

class TestClass {
  private int a;
  public double b;
  public TestClass(int first, double second) {
    this.a = first;
    this.b = second;
  }
}
```

[✓ Compiler Error] ✓ Correct

Feedback:

Correct. 'a' is a private variable declared inside TestClass, so it cannot be accessed from the Main class. So, this will show an error. Hence, this is correct.

You can read more about different access modifiers in Java from its official documentation.

2, 5.2

3, 7.5

2, 3

MyClass

The code given below does not compile. Identify the line which will throw an error.

```
public class MyClass {
   int attr1;
   static int attr2;

   void method1() {
       System.out.println(attr1); //Line 1
       System.out.println(attr2); //Line 2
   }

   static void method2() {
       System.out.println(attr1); //Line 3
       System.out.println(attr2); //Line 4
   }
}
```

[✓ Line 1] ✗ Incorrect

Feedback:

Incorrect. In the first line, you are accessing the instance attribute inside an instance method. This is permissible and there is nothing wrong with this code. Hence, this line of code won't throw an error.

Line 2

Line 3 ✓ Correct

Feedback:

Correct. In the third line, you are accessing the instance attribute inside a static method. Static methods cannot access instance members (both instance attributes and instance methods). This is not permissible and it will throw an error. Hence, this option is the correct choice.

Line 4

Your answer is Wrong.

Attempt 2 of 2

Continue >

In a small company, whenever a new employee joins the firm, he/she is given an Employee ID and offered a package as salary. Which of the following pieces of code should be used to achieve the following functionalities:

1. The employee's employeeId and salary attributes should be initialised only using a constructor.
2. The employeeId attribute cannot be changed by anyone once initialised (read-only).
3. The salary attribute can be read and changed by the HR team.

```java
public class Employee {
    private int employeeId;
    private Double salary;

    public Employee(int employeeId, Double salary) {
        this.employeeId = employeeId;
        this.salary = salary;
    }

    public int getEmployeeId() {
        return employeeId;
    }

    public void setEmployeeId(int employeeId) {
        this.employeeId = employeeId;
    }

    public Double getSalary() {
        return salary;
    }

    public void setSalary(Double salary) {
        this.salary = salary;
    }
}
```

```java
public class Employee {
    private int employeeId;
    private Double salary;

    public int getEmployeeId() {
        return employeeId;
    }

    public Double getSalary() {
        return salary;
    }

    public void setSalary(Double salary) {
        this.salary = salary;
    }
}
```

[✓]

```java
public class Employee {
    private int employeeId;
    private Double salary;

    public Employee(int employeeId, Double salary) {
        this.employeeId = employeeId;
        this.salary = salary;
    }

    public int getEmployeeId() {
        return employeeId;
    }

    public Double getSalary() {
        return salary;
    }

    public void setSalary(Double salary) {
        this.salary = salary;
    }
}
```

✓ Correct

Feedback:

Correct. Employees can be assigned employeeId and salary attributes while being created. Removing the setter for employeeId will make sure that no one can change employeeId. And getters and setters for salary will make sure that it can be modified by the HR team whenever required.

You can refer to this blog to read more about how getters and setters work.

Abstraction

Which of the following is true about abstraction?

- Abstraction means binding the code and data into a single unit.
- The principle of inheritance can be achieved using abstraction.
- Encapsulation can be achieved using abstraction.
- [✓ Abstraction is used to avoid duplication.] ✓ Correct

Feedback:

Correct. The main benefit of using abstraction in programming is that it allows you to group several related classes as siblings. Abstraction in object-oriented programming helps reduce the complexity of the design and implementation process of software.

To read more about abstraction and related concepts, you can refer to this blog.

- [✓ Yes, By adding a return keyword.] ✓ Correct

Feedback:

Correct! Once we add return type to constructor signature, constructor becomes a normal method.

```java
public class HelloWorld {
   public HelloWorld(){
       // This is the constructor for given class
   }

   public void HelloWorld(){
       /** As we have added a return type, in this case void
        * this will act as a normal method
       */
   }
}
```

Stack and Heap Memory

Where are local and instance variables stored in Java memory respectively? Choose the correct answer from the options given below.

[✓ Stack, Heap] ✓ Correct

Feedback:

Correct! Local variables are stored in the stack, while instance variables are stored in heap memory. To read more about stack and heap memory, you can refer to the following blog.

Method Overloading

Which of the following statements is INCORRECT in the case of method overloading in Java?

The number of parameters in the overloaded methods must be different.

If there are two overloaded methods with two parameters, then the data types of these two parameters should be different.

[✓ Overloaded methods should have the same number and type of arguments in the same order.] ✓ Correct

Feedback:

If there are two methods with the same name, number and type of arguments in the same order, then these two methods have the same signature. As you know, having two methods with the same signature is not allowed. Hence, this option is the correct choice.

Method Overriding

Is the getDetails() method declaration in the subclass of the following code snippet correct?

```
class Employee {
public void getDetails() {}
}
class Manager extends Employee {
   private void getDetails() {}
}
```

Yes [✓ No] ✓ Correct

- Feedback:

  Correct. The access modifier of a method declared public in a superclass cannot be private in a subclass as it will be more restrictive when compared to the superclass version. Remember, the private access specifier only restricts access to a member or method within the class. Hence, this is correct.

Access Modifier	Within Class	Within Package	Outside Package by Subclass Only	Outside Package
Private	Y	Y	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Dynamic Polymorphism

What is the output of the following code snippet?

```java
public class Main {
    public static void main (String args[]) {
        Scientist[] team = new Scientist[2];
        team[0] = new Godel();
        team[1] = new Tesla();

        System.out.println("Scientist list");
        for(Scientist scientist : team){
            scientist.printTheory();
        }
    }
}

class Scientist {
    private String theory = "Science";
    public void printTheory() {
        System.out.println(theory);
    }
}

class Godel extends Scientist {
    private String theory = "Mathematics";
    public void printTheory() {
        System.out.println(theory);
    }
}


class Tesla extends Scientist {
    private String theory = "Physics";
    public void printTheory() {

        System.out.println(theory);
    }
}
```

○ Scientist list

Science

Science

[✓ Scientist list

Mathematics

Physics] ✓ Correct

- Feedback:

Correct. Based on the concept of method overriding and dynamic polymorphism, the printTheory method of the Scientist parent class will get overridden by the printTheory method of the child classes Godel and Tesla. So, when printTheory() is called at runtime, the methods run as defined for the Godel and Tesla classes, and hence print 'Mathematics' and 'Physics', respectively.

○ Scientist list

Physics

Mathematics

○ Scientist list

Mathematics

Science

Accessing Methods

Which of the following statements is correct?

(Note that more than one option may be correct.)

☐ An object of a superclass can access the public methods of the same superclass (itself). ✓ Correct
You missed this!

Feedback:

Correct. An object of a class can access all the methods of that class. Refer to this link to understand in-depth how the object of a class can access all the methods of that class.

☐ An object of a subclass can access the public methods of the same subclass (itself). ✓ Correct
You missed this!

Feedback:

Correct. An object of a class can access all the methods of that class. Refer to this link to understand in-depth how the object of a class can access all the methods of the class.

☑ An object of a superclass can access the public methods of its subclass. ✕ Incorrect

Feedback:

Incorrect. An object of a superclass does not have any access to its subclass. Thus, an object of a superclass cannot invoke a method of its subclass.

☑ An object of a subclass can access the public methods of its superclass. ✓ Correct

Feedback:

Correct. An object of a subclass has access to the public methods of its superclass. Thus, all the public methods of a superclass can be accessed easily by an object of its subclass.

Variable Shadowing

What will be the output of the following program?

```java
class SuperClass{
        public String val = "SUPER_VAL";
}
public class ChildClass extends SuperClass{
        private String val;
        public ChildClass(String value) {
                this.val = value;
        }
        protected void display(){
                System.out.println("val = "+this.val);
                System.out.println("val = "+super.val);
        }
        public static void main(String[] args){
                ChildClass child = new ChildClass("CHILD_VAL");
                child.display();
        }
}
```

[✓ val = CHILD_VAL

val = SUPER_VAL] ✓ Correct

- Feedback:

  Correct. Since the names of the attributes are the same in both the parent and the child class, the constructor in the child class initialises the attribute of the child class. The data in the parent class is shadowed when the display method is called. You have to use the 'super' keyword to access the data of the parent class.

---

val = CHILD_VAL

val = CHILD_VAL

---

val = SUPER_VAL

val = SUPER_VAL

---

val = SUPER_VAL

val = CHILD_VAL

What will be the output of the following program?

```java
class P{
    public P(int n) {
        System.out.print(n);
    }
}

public class C extends P {
    public C(int n) {
        System.out.print(n);
    }
    public static void main(String[] args) {
        C c = new C(10);
        System.out.println(" ");
    }
}
```

- 10

  10

- 10

- [✓ 0

  10] ✕ Incorrect

  Feedback:

  Incorrect. The implied call to P's constructor in C's constructor cannot be satisfied because P lacks a no-argument constructor. A default constructor is generated by the compiler only if the class has no constructor explicitly defined.

- Compile Error ✓ Correct

  Feedback:

  Correct. The implied super() call in C's constructor cannot be satisfied because P lacks a no-argument constructor. A no-argument (default) constructor is generated by the compiler only if the class has no constructor explicitly defined. This produces an error since no argument was passed for the superclass constructor.

Polymorphism

Consider the below code snippet:

```
class sampleClass {
    static void printDetails(String name){
        System.out.println(name);
    }

    void printDetails(String name){
        System.out.println(name);
    }
};
```

What is this an example of?

Note: 'Signature' refers to the return type, the number of parameters and the data types of the parameters. In the above code snippet, the only difference between the two definitions of the printDetails() method is the keyword 'static'. The compiler checks only the method signature and not the 'static' keyword to verify method overloading.

- ○ Method overloading
- ○ Method overriding
- ○ Method duplication ✓ Correct

  Feedback:
  Option (c) is correct because, in the above code snippet, the signature of both the methods is exactly the same. The only difference is in the keyword 'static'. Having one method as static and the other as non-static does not make them two different methods.

- [✓ None of the above] ✗ Incorrect

  Feedback:
  At least one of the above options is correct.

Final classes

Which of the following statements is true with respect to final classes?

[✓ Final classes cannot be extended.] ✓ Correct

Feedback:
Correct. The only limitation of final classes is that they cannot be extended.

Final classes cannot be instantiated.

Final classes cannot extend other classes.

Final classes can only have final attributes and methods.

Super vs Object

By default, every class in Java extends the _________ class.

Object ✓ Correct

Feedback:
Correct. In Java, every class extends the Object class, directly (if there is no superclass) or indirectly (if there is a superclass and if that superclass doesn't have a superclass). In other words, with multi-level inheritance, a class will indirectly inherit the properties of the Object class. To read more about multi-level inheritance, refer to this blog.

[✓ Super] ✕ Incorrect

Feedback:
Incorrect. 'super' is a keyword used to refer to the parent class when a class extends another. 'super' itself isn't a class.

Object-Oriented Programming

What would be the output of the code given below?

```java
class BalanceAmount  {
    int balance;
    public void setBalance(int amount)  {
        balance=amount;
    }
    public void incrementBalance()  {
        balance+=10;
    }
    public int getCurrentBalance()  {
        return  balance;
    }
}
public class Source {
    public static void main(String[] args) {
        BalanceAmount obj= new BalanceAmount();
        obj.setBalance(20);
        obj.incrementBalance();
        System.out.println(obj.getCurrentBalance());
        obj.balance=1000;
        System.out.println(obj.getCurrentBalance());
    }
}
```

- Compilation error

- 20

  30

- 30

  30

- [✓ 30

  1000] ✓ Correct

  - Feedback:

    Correct. The setBalance() method will set the value of the variable ‘balance’ as 20, but after the incrementBalance() method is called, the value of 'balance' will be changed to 30. Therefore, the first output is 30. Now, when the variable ‘balance’ is called through obj.balance and is assigned a value of 1000, the value of 'balance' will change to 1000 because the variable 'balance' is declared public, i.e., it can be accessed outside the class. Hence, 1000 will be printed next.

1) What is the output of the following code snippet?

```
int a = 15;
int b = 20;
int c = 10;
int res = a > b ? (a > c ? a : c) : (b > c ? b : c);

System.out.println(res);      //20
```

2) What will be the output if you use the following piece of code?

```
public class MyClass {
   public static void main(String[] args) {
      boolean keepRunning = false;
      while (keepRunning = true) {
         System.out.println("Hey");
      }
   }
}           //Print "Hey" Infinite times
```

3) Checking if the number is Armstrong:

```
package Practice;
import java.util.Scanner;
public class Loops {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      int num = scan.nextInt();
      int numCopy = num;
      int sum = 0;
      while(num !=0){
         int number = num%10;
         num = num/10;
         int cube = number * number * number;
         sum += cube;
      }
       System.out.println(sum);
   }
   }
```

4) Find the lastNonZeroDigit in the factorial of the given number:

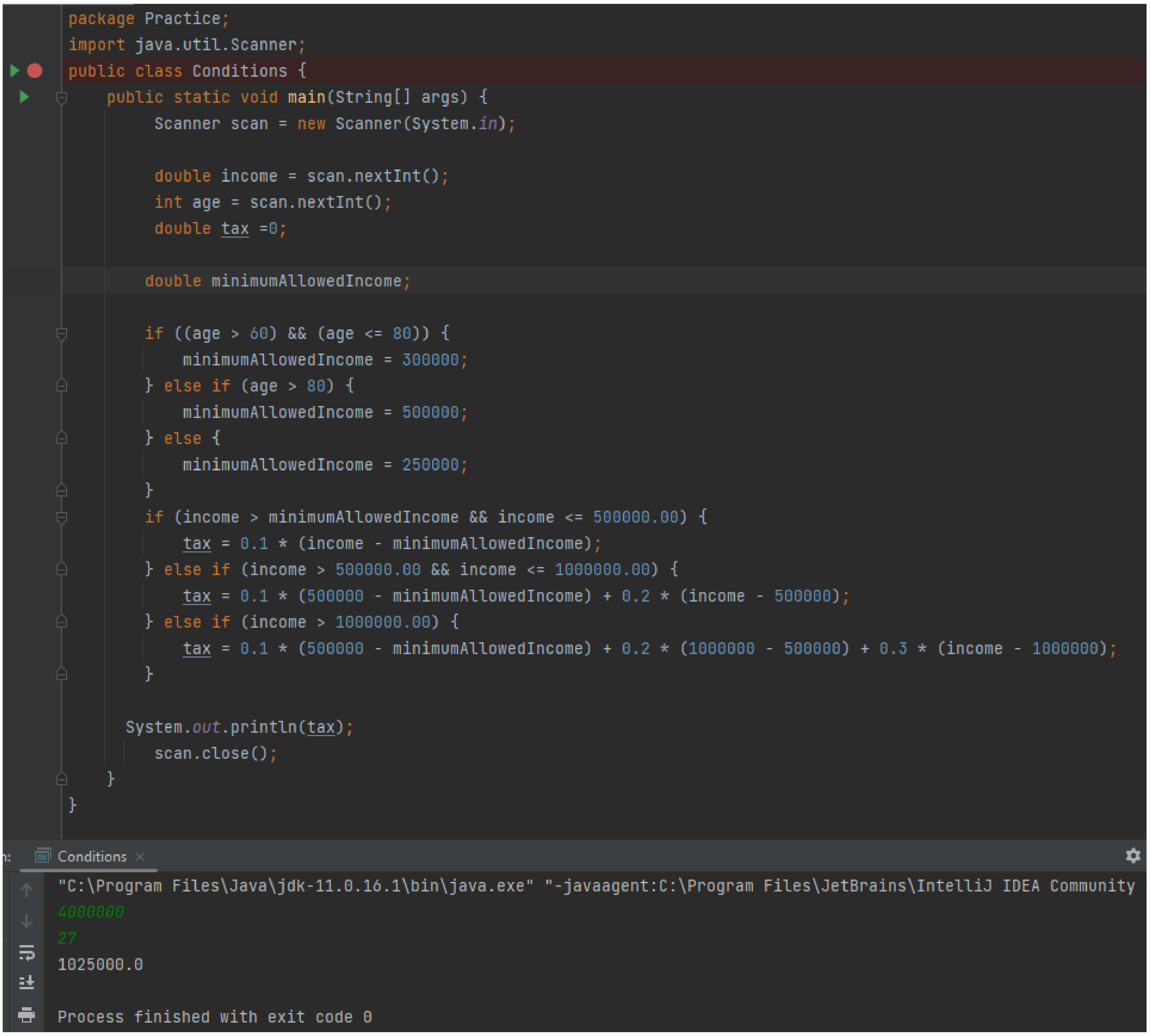
Num =9; 9! = 326880 ➔ 8

```
package Practice;
import java.util.Scanner;
public class PrimitiveTypes {
   public static void main(String[] args) {
      Scanner s = new Scanner(System.in);
      int n = s.nextInt();
      System.out.print(lastNonZeroDigit(n));
   }
   // Method to find the last digit of n!
   static int lastNonZeroDigit(int n) {
      int val = 1;
      for (int i = n; i >= 1; i--) {
          val *= i;
      }
      int num = 0;
      String countString = Integer.toString(val);
      for(int i=0;i<countString.length();i++){
          if(val%10 == 0){
              val = val/10;
          }else{
              num = val%10;
              break;
          }
      }
      return num;
   }
}
```

5) Find the LCM of Numbers:

- Initialize A and B with positive integers.
- Store maximum of A & B to the max.
- Check if max is divisible by A and B.
- If divisible, Display max as LCM.
- If not divisible then step increase max, go to step 3.

"TAX" PROBLEM

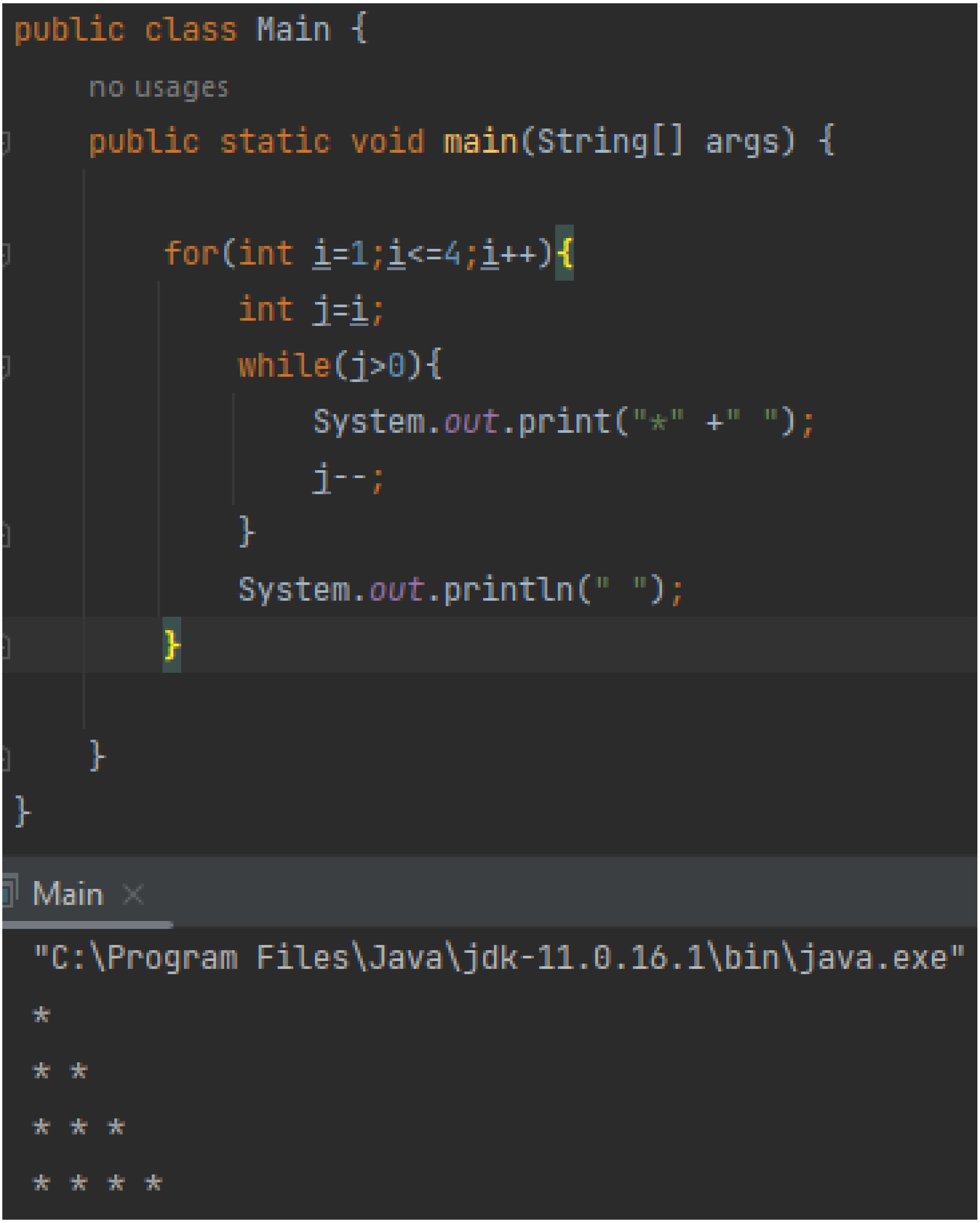


```
package Practice;
import java.util.Scanner;
public class Conditions {
    public static void main(String[] args) {
         Scanner scan = new Scanner(System.in);

         double income = scan.nextInt();
         int age = scan.nextInt();
         double tax =0;

        double minimumAllowedIncome;

        if ((age > 60) && (age <= 80)) {
            minimumAllowedIncome = 300000;
        } else if (age > 80) {
            minimumAllowedIncome = 500000;
        } else {
            minimumAllowedIncome = 250000;
        }
        if (income > minimumAllowedIncome && income <= 500000.00) {
            tax = 0.1 * (income - minimumAllowedIncome);
        } else if (income > 500000.00 && income <= 1000000.00) {
            tax = 0.1 * (500000 - minimumAllowedIncome) + 0.2 * (income - 500000);
        } else if (income > 1000000.00) {
            tax = 0.1 * (500000 - minimumAllowedIncome) + 0.2 * (1000000 - 500000) + 0.3 * (income - 1000000);
        }

      System.out.println(tax);
         scan.close();
    }
}
```

Conditions

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community
4000000
27
1025000.0

Process finished with exit code 0
```

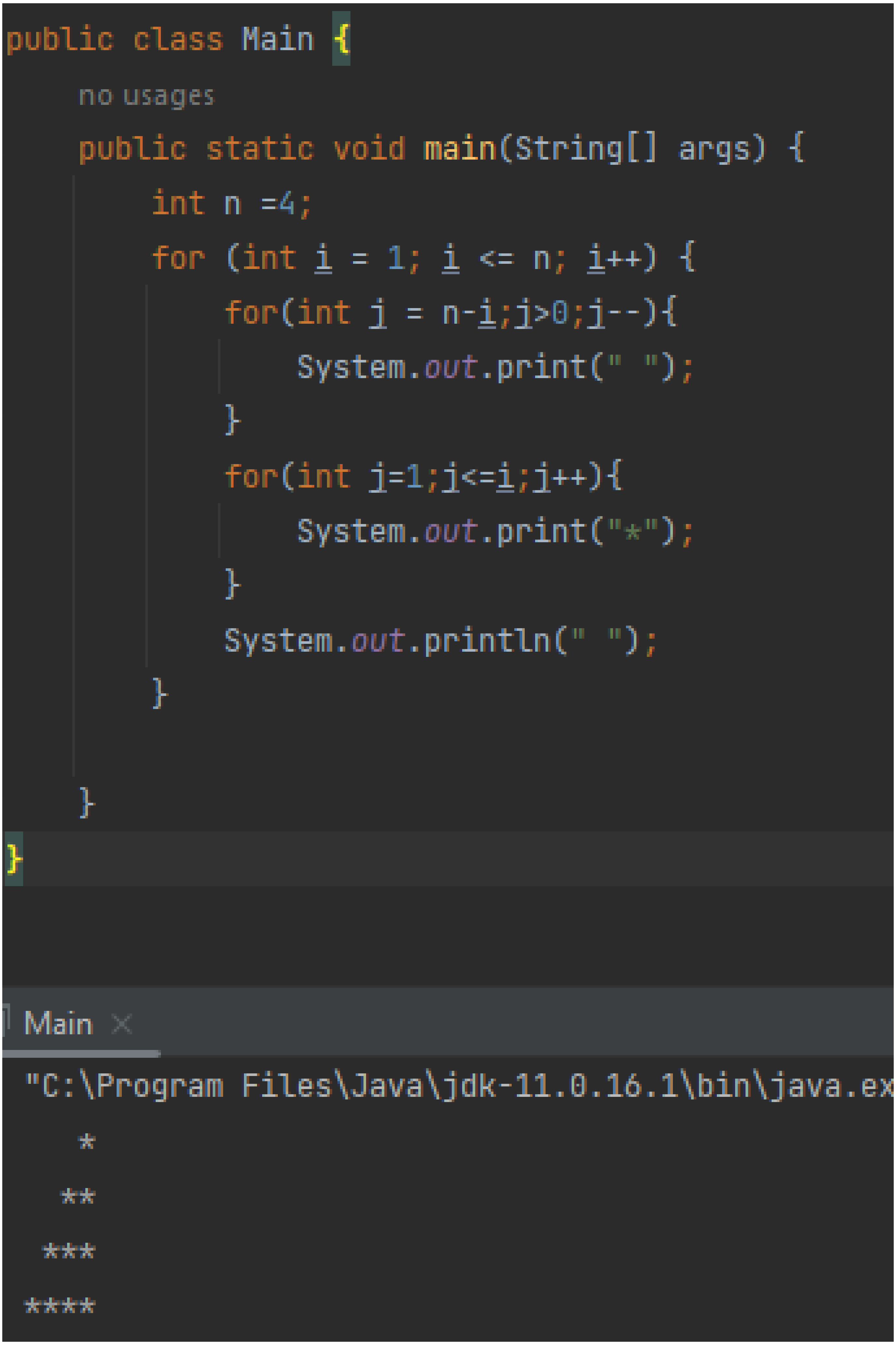
6) Pattern Problem:



```
public class Main {
    no usages
    public static void main(String[] args) {

        for(int i=1;i<=4;i++){
            int j=i;
            while(j>0){
                System.out.print("*" +" ");
                j--;
            }
            System.out.println(" ");
        }

    }
}
```

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe"
*
* *
* * *
* * * *
```

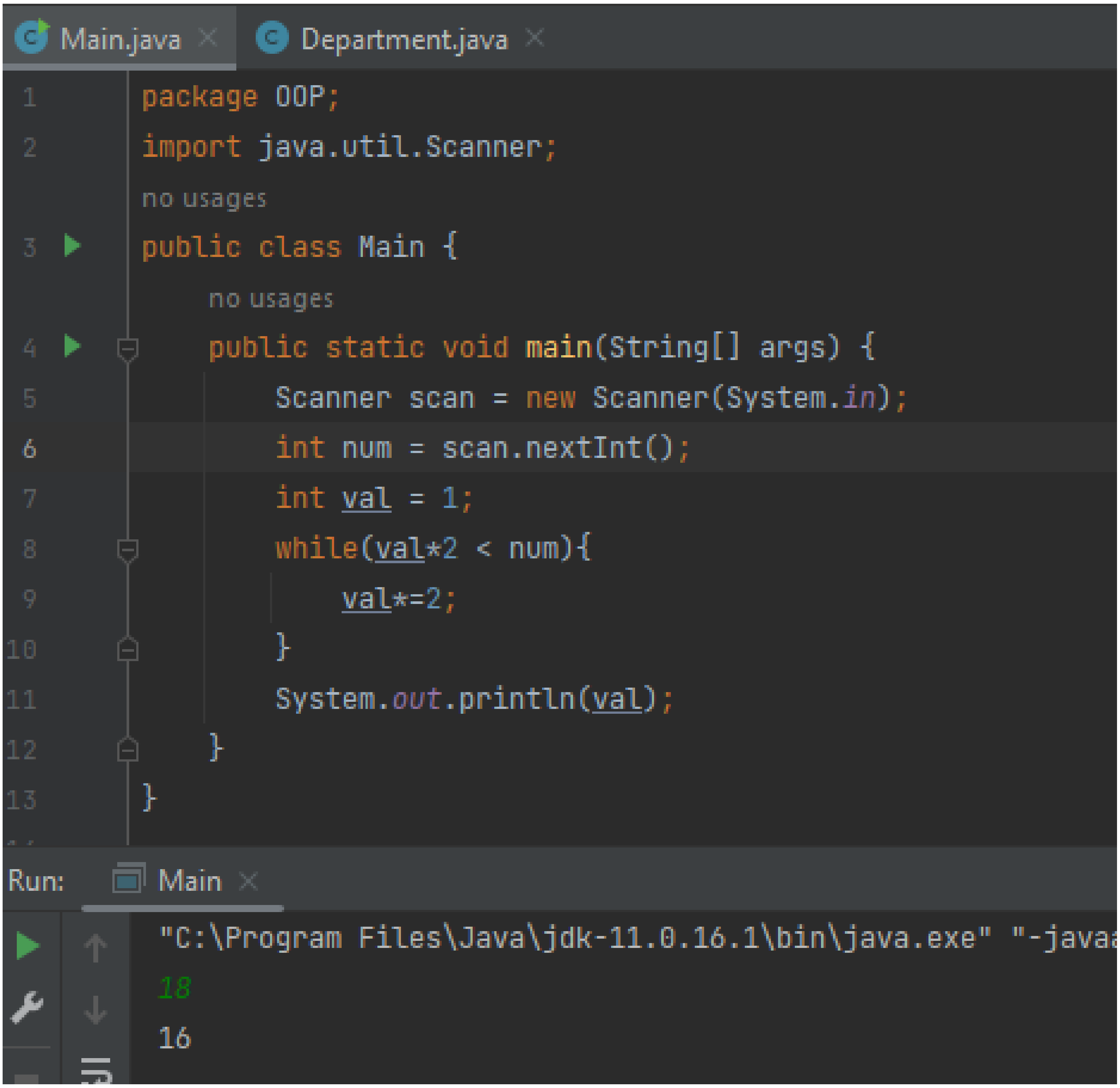
Pattern Problem:



```java
public class Main {
    no usages
    public static void main(String[] args) {
        int n =4;
        for (int i = 1; i <= n; i++) {
            for(int j = n-i;j>0;j--){
                System.out.print(" ");
            }
            for(int j=1;j<=i;j++){
                System.out.print("*");
            }
            System.out.println(" ");
        }

    }
}
```

```
Main

"C:\Program Files\Java\jdk-11.0.16.1\bin\java.ex
   *
  **
 ***
****
```

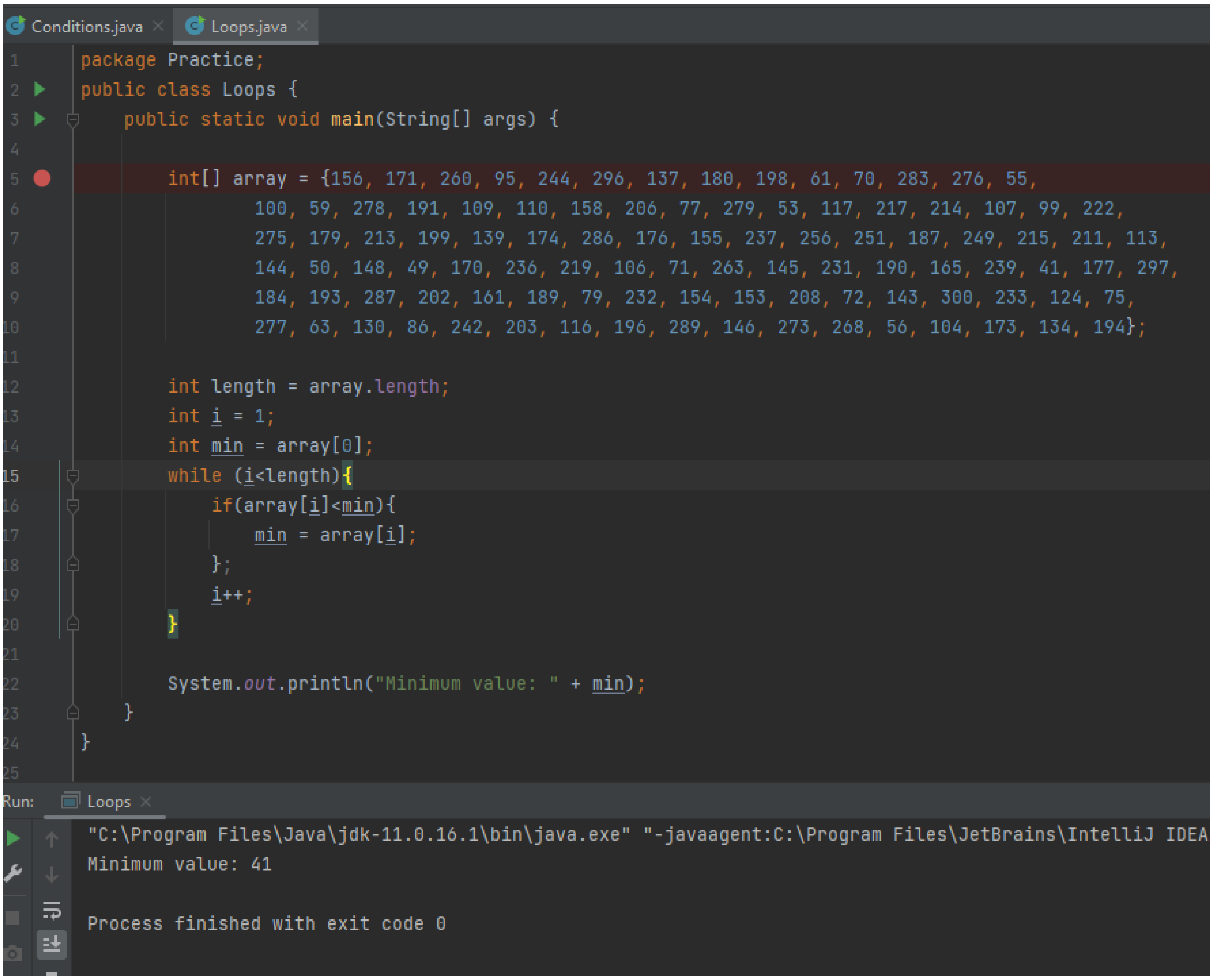
10) Write a code that prints the highest power of 2, less than or equal to a given number. For e.g., if the input number is 9, the code should print 8, as 8 or 23 is the highest power of two which is less than 9.



```java
package OOP;
import java.util.Scanner;
no usages
public class Main {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
        int val = 1;
        while(val*2 < num){
            val*=2;
        }
        System.out.println(val);
    }
}
```

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "-java
18
16
```

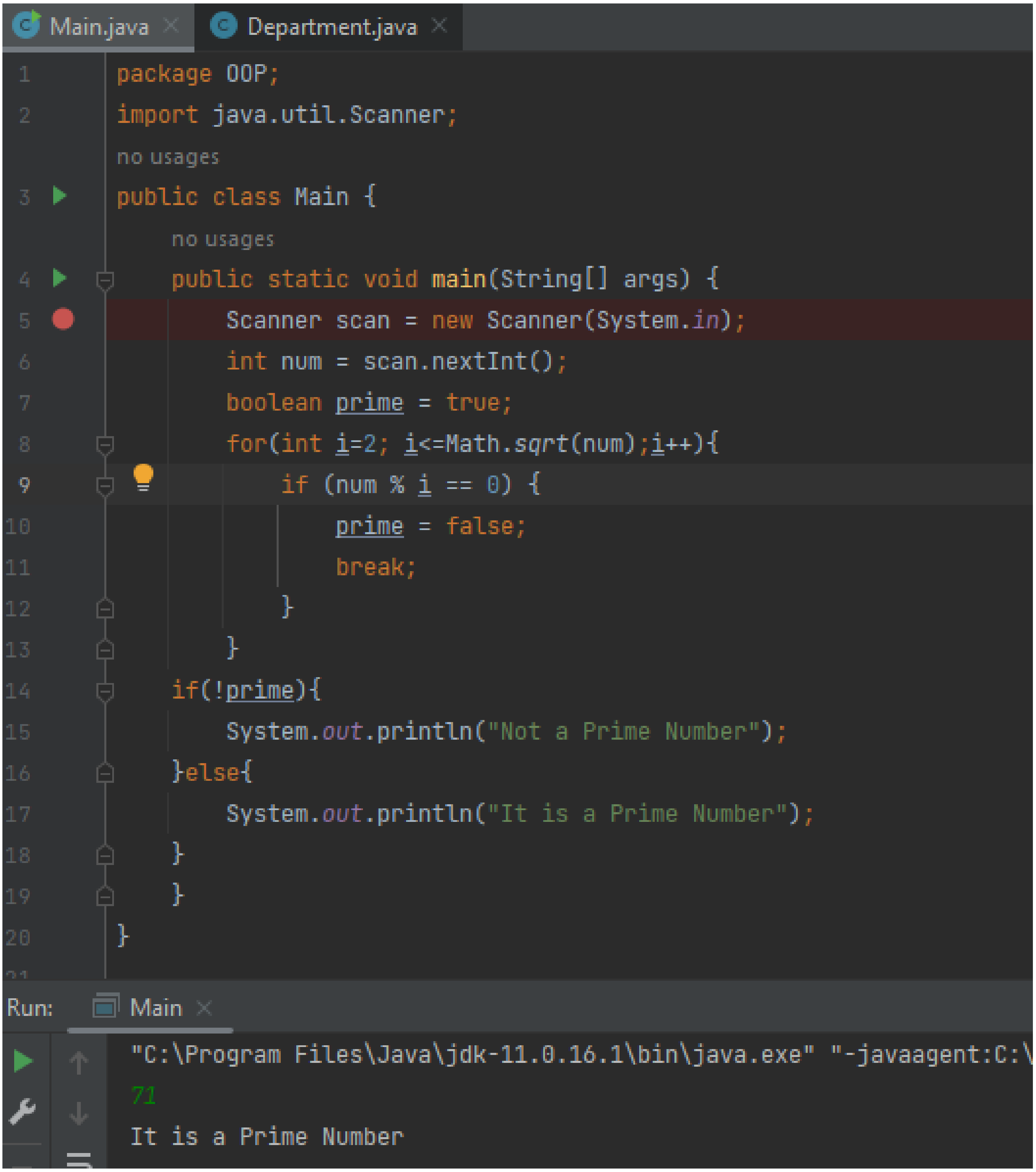
11) Write the programming logic in order to find the minimum element of the provided array.



```java
package Practice;
public class Loops {
    public static void main(String[] args) {

        int[] array = {156, 171, 260, 95, 244, 296, 137, 180, 198, 61, 70, 283, 276, 55,
                100, 59, 278, 191, 109, 110, 158, 206, 77, 279, 53, 117, 217, 214, 107, 99, 222,
                275, 179, 213, 199, 139, 174, 286, 176, 155, 237, 256, 251, 187, 249, 215, 211, 113,
                144, 50, 148, 49, 170, 236, 219, 106, 71, 263, 145, 231, 190, 165, 239, 41, 177, 297,
                184, 193, 287, 202, 161, 189, 79, 232, 154, 153, 208, 72, 143, 300, 233, 124, 75,
                277, 63, 130, 86, 242, 203, 116, 196, 289, 146, 273, 268, 56, 104, 173, 134, 194};

        int length = array.length;
        int i = 1;
        int min = array[0];
        while (i<length){
            if(array[i]<min){
                min = array[i];
            };
            i++;
        }

        System.out.println("Minimum value: " + min);
    }
}
```

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
Minimum value: 41

Process finished with exit code 0
```

12) Prime Number



```java
package OOP;
import java.util.Scanner;
no usages
public class Main {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
        boolean prime = true;
        for(int i=2; i<=Math.sqrt(num);i++){
            if (num % i == 0) {
                prime = false;
                break;
            }
        }
    if(!prime){
        System.out.println("Not a Prime Number");
    }else{
        System.out.println("It is a Prime Number");
    }
    }
}
```

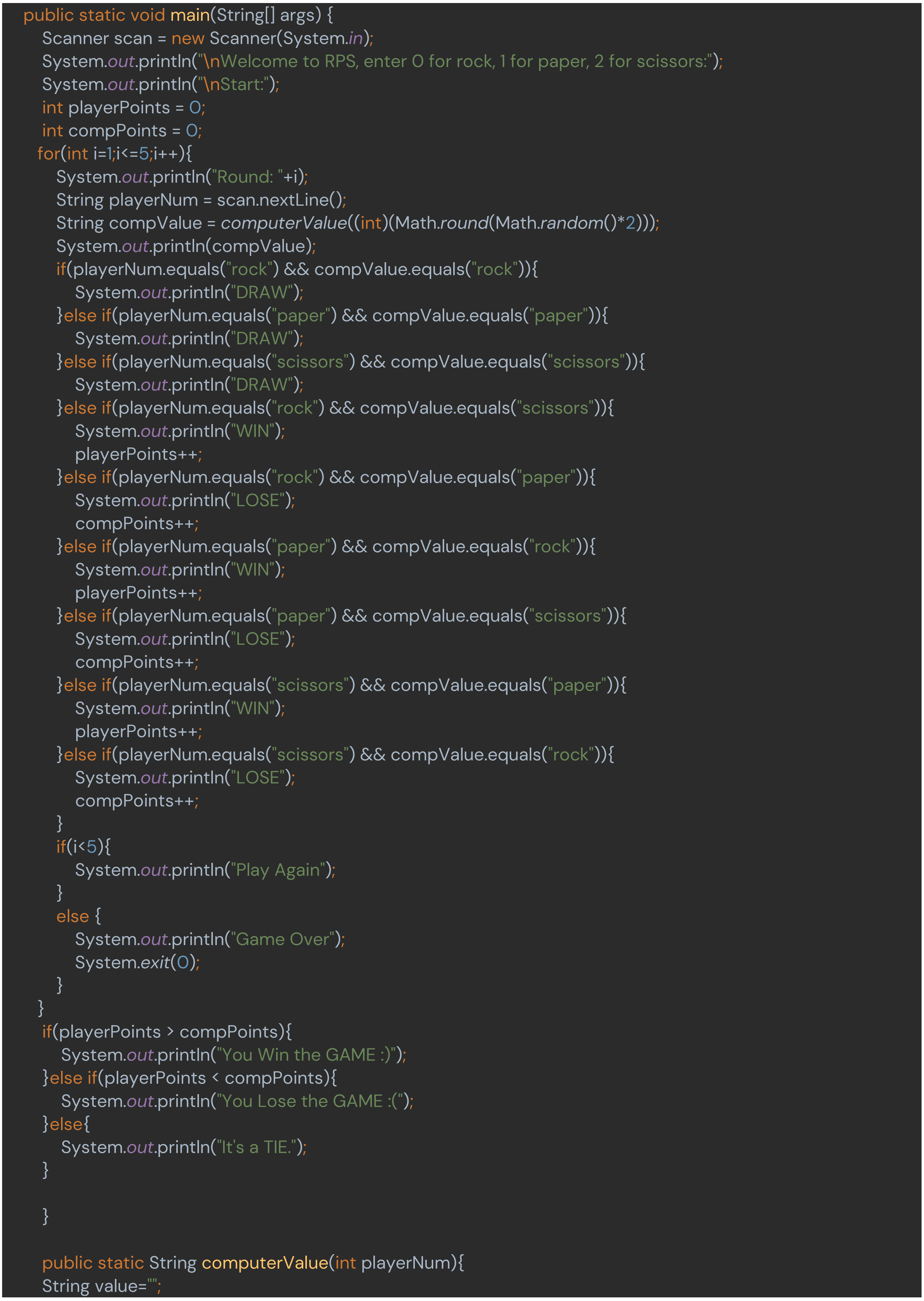
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "-javaagent:C:\
71
It is a Prime Number
```

13)Using data from a File and performing operations on them

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Loops {
    public static void main(String[] args) throws FileNotFoundException{
        File ipFile = new File("C:/Users/Harish S/Desktop/nums.txt");
        Scanner scan = new Scanner(ipFile);
        double sum =0;
        double total = 0;

        while (scan.hasNext()){
            int number = scan.nextInt();
            sum+=number;
            total++;
        }

        System.out.println(sum/total);
    }
}
```

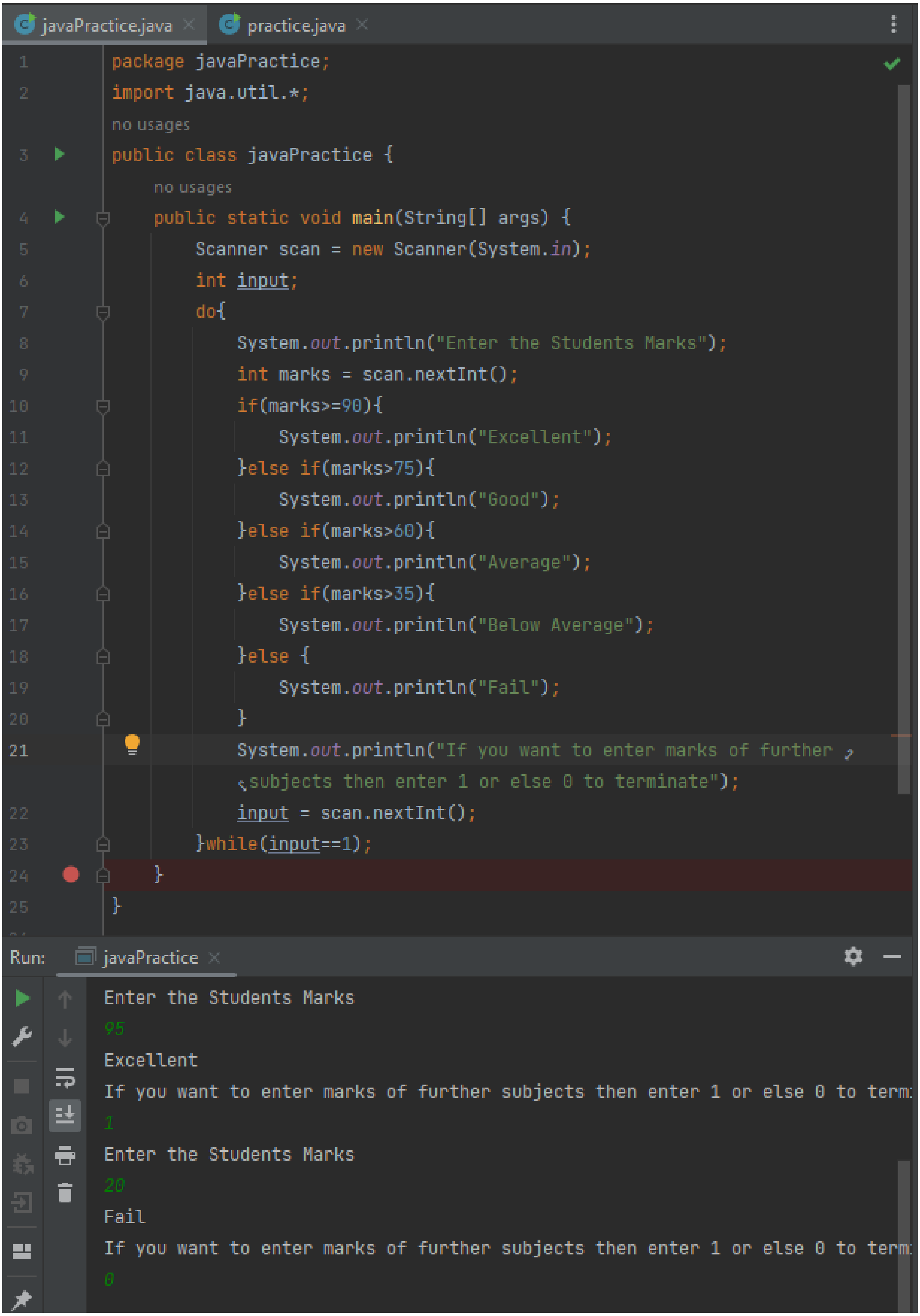
14)Rock Paper Scissors

```java
import java.util.Scanner;
public class HandsOnProblems {
```



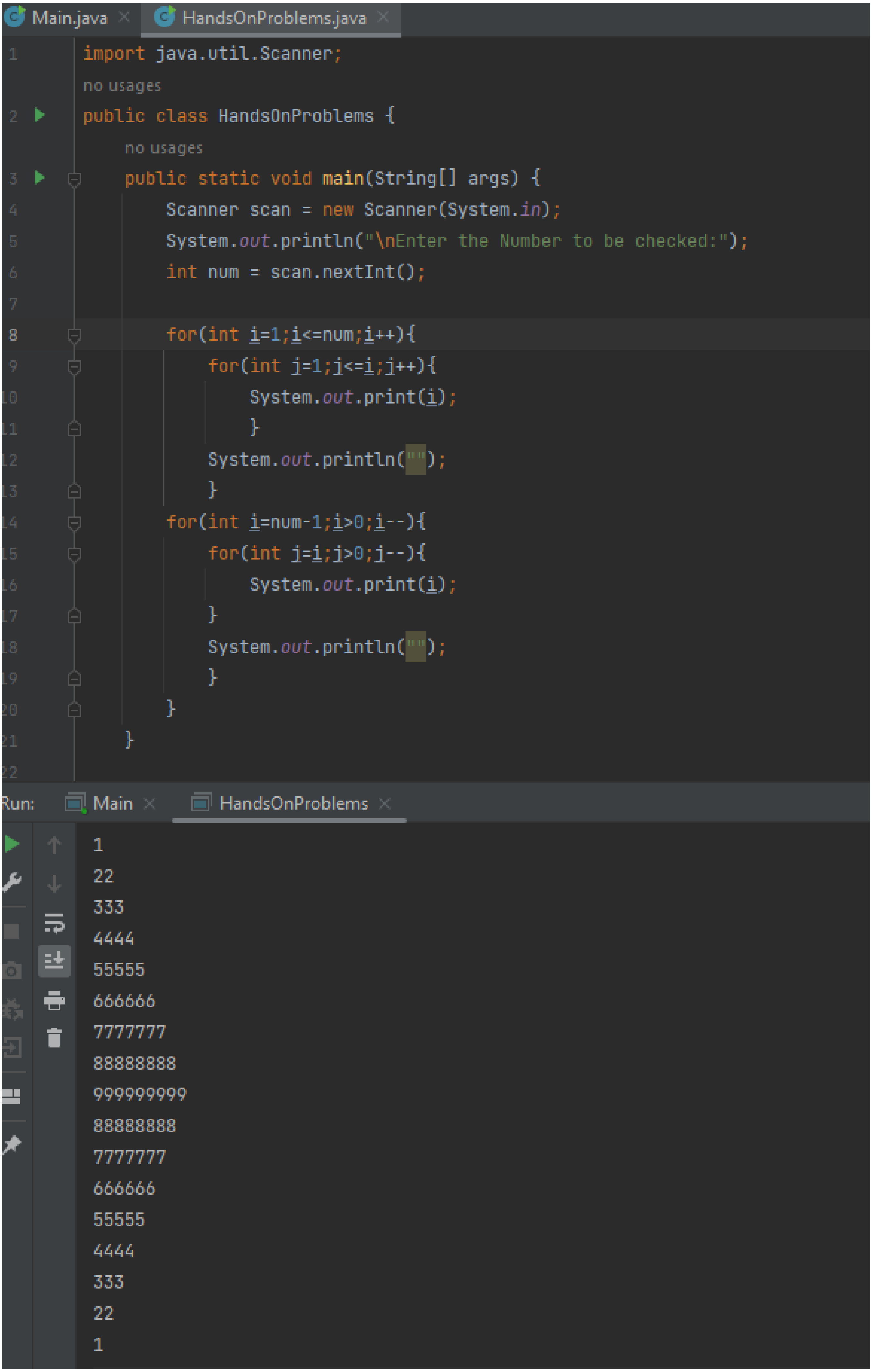
```java
public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    System.out.println("\nWelcome to RPS, enter 0 for rock, 1 for paper, 2 for scissors:");
    System.out.println("\nStart:");
    int playerPoints = 0;
    int compPoints = 0;
   for(int i=1;i<=5;i++){
      System.out.println("Round: "+i);
      String playerNum = scan.nextLine();
      String compValue = computerValue((int)(Math.round(Math.random()*2)));
      System.out.println(compValue);
      if(playerNum.equals("rock") && compValue.equals("rock")){
         System.out.println("DRAW");
      }else if(playerNum.equals("paper") && compValue.equals("paper")){
         System.out.println("DRAW");
      }else if(playerNum.equals("scissors") && compValue.equals("scissors")){
         System.out.println("DRAW");
      }else if(playerNum.equals("rock") && compValue.equals("scissors")){
         System.out.println("WIN");
         playerPoints++;
      }else if(playerNum.equals("rock") && compValue.equals("paper")){
         System.out.println("LOSE");
         compPoints++;
      }else if(playerNum.equals("paper") && compValue.equals("rock")){
         System.out.println("WIN");
         playerPoints++;
      }else if(playerNum.equals("paper") && compValue.equals("scissors")){
         System.out.println("LOSE");
         compPoints++;
      }else if(playerNum.equals("scissors") && compValue.equals("paper")){
         System.out.println("WIN");
         playerPoints++;
      }else if(playerNum.equals("scissors") && compValue.equals("rock")){
         System.out.println("LOSE");
         compPoints++;
      }
      if(i<5){
         System.out.println("Play Again");
      }
      else {
         System.out.println("Game Over");
         System.exit(0);
      }
   }
    if(playerPoints > compPoints){
       System.out.println("You Win the GAME :)");
    }else if(playerPoints < compPoints){
       System.out.println("You Lose the GAME :(");
    }else{
       System.out.println("It's a TIE.");
    }

    }

    public static String computerValue(int playerNum){
    String value="";
```

```
        switch (playerNum){
            case 0:
                value= "rock";
                break;
            case 1:
                value= "paper";
                break;
            case 2:
                value= "scissors";
                break;
        }
        return value;
    }
}
```
15) Make a menu driven program. The user can enter 2 numbers, either 1 or 0. If the user enters 1 then keep taking input from the user for a student's marks (out of 100). If they enter 0 then stop.



```java
package javaPractice;
import java.util.*;
public class javaPractice {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int input;
        do{
            System.out.println("Enter the Students Marks");
            int marks = scan.nextInt();
            if(marks>=90){
                System.out.println("Excellent");
            }else if(marks>75){
                System.out.println("Good");
            }else if(marks>60){
                System.out.println("Average");
            }else if(marks>35){
                System.out.println("Below Average");
            }else {
                System.out.println("Fail");
            }
            System.out.println("If you want to enter marks of further 
            subjects then enter 1 or else 0 to terminate");
            input = scan.nextInt();
        }while(input==1);
    }
}
```

```
Enter the Students Marks
95
Excellent
If you want to enter marks of further subjects then enter 1 or else 0 to term
1
Enter the Students Marks
20
Fail
If you want to enter marks of further subjects then enter 1 or else 0 to term
0
```

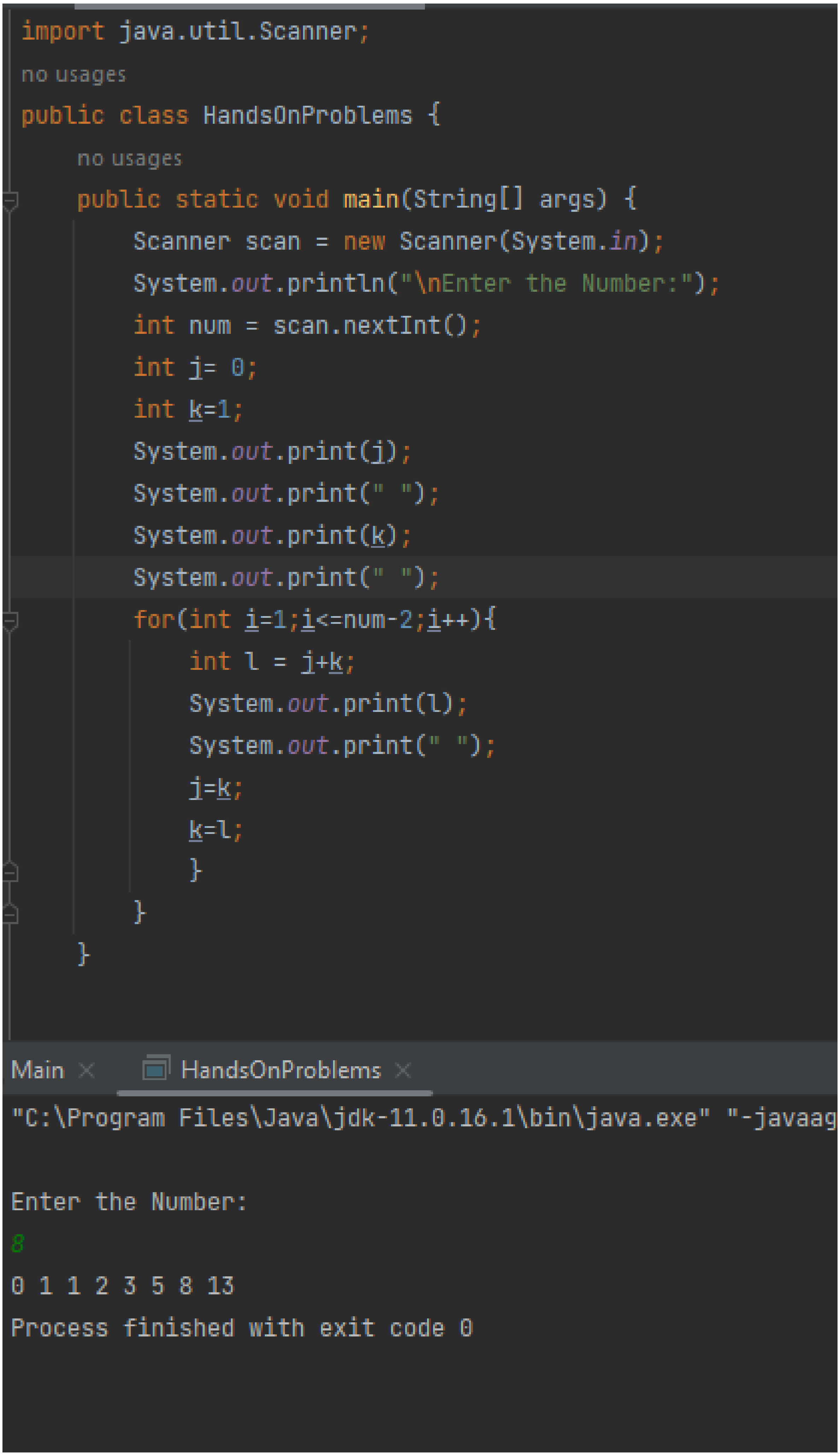

Main.java HandsOnProblems.java

```java
import java.util.Scanner;
no usages
public class HandsOnProblems {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("\nEnter the Number to be checked:");
        int num = scan.nextInt();

        for(int i=1;i<=num;i++){
            for(int j=1;j<=i;j++){
                System.out.print(i);
                }
            System.out.println("");
            }
        for(int i=num-1;i>0;i--){
            for(int j=i;j>0;j--){
                System.out.print(i);
            }
            System.out.println("");
            }
        }
    }

```

Run: Main HandsOnProblems

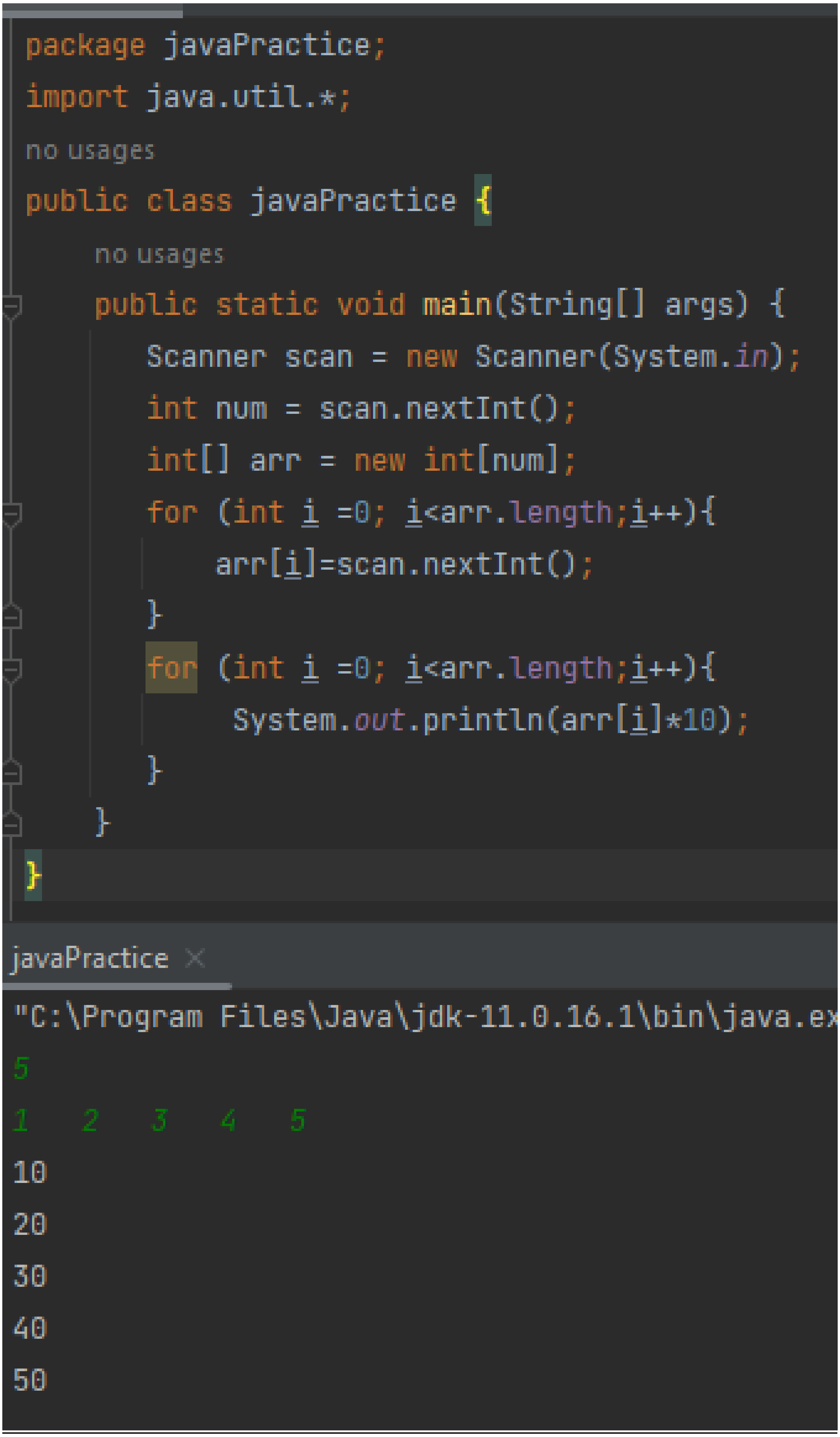
```
1
22
333
4444
55555
666666
7777777
88888888
999999999
88888888
7777777
666666
55555
4444
333
22
1
```

16) Pattern Problem



```
import java.util.Scanner;
no usages
public class HandsOnProblems {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("\nEnter the Number:");
        int num = scan.nextInt();
        int j= 0;
        int k=1;
        System.out.print(j);
        System.out.print(" ");
        System.out.print(k);
        System.out.print(" ");
        for(int i=1;i<=num-2;i++){
            int l = j+k;
            System.out.print(l);
            System.out.print(" ");
            j=k;
            k=l;
            }
        }
    }
```

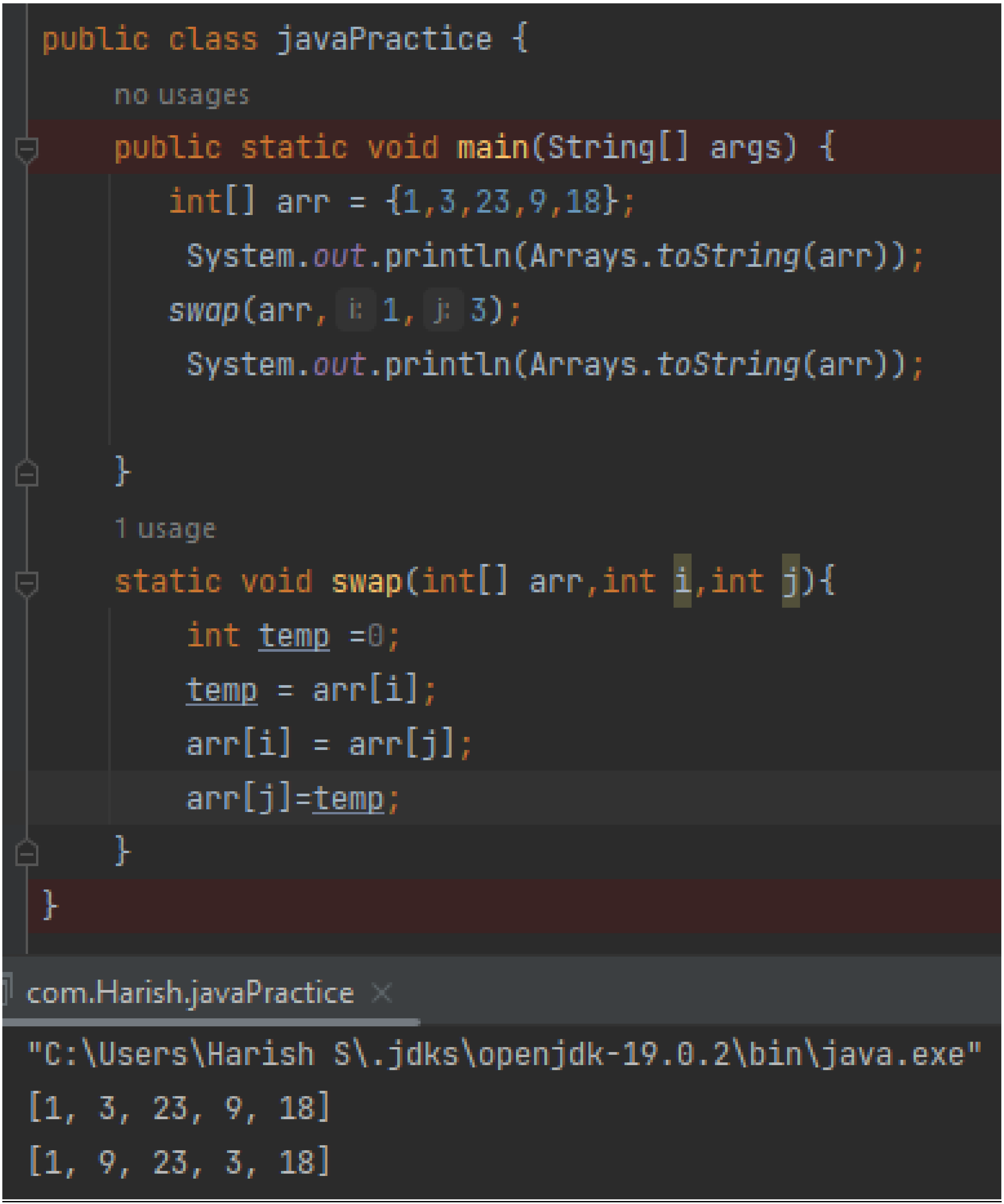
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "-javaag

Enter the Number:
8
0 1 1 2 3 5 8 13
Process finished with exit code 0
```



```java
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
        int[] arr = new int[num];
        for (int i =0; i<arr.length;i++){
            arr[i]=scan.nextInt();
        }
        for (int i =0; i<arr.length;i++){
             System.out.println(arr[i]*10);
        }
    }
}
```

javaPractice

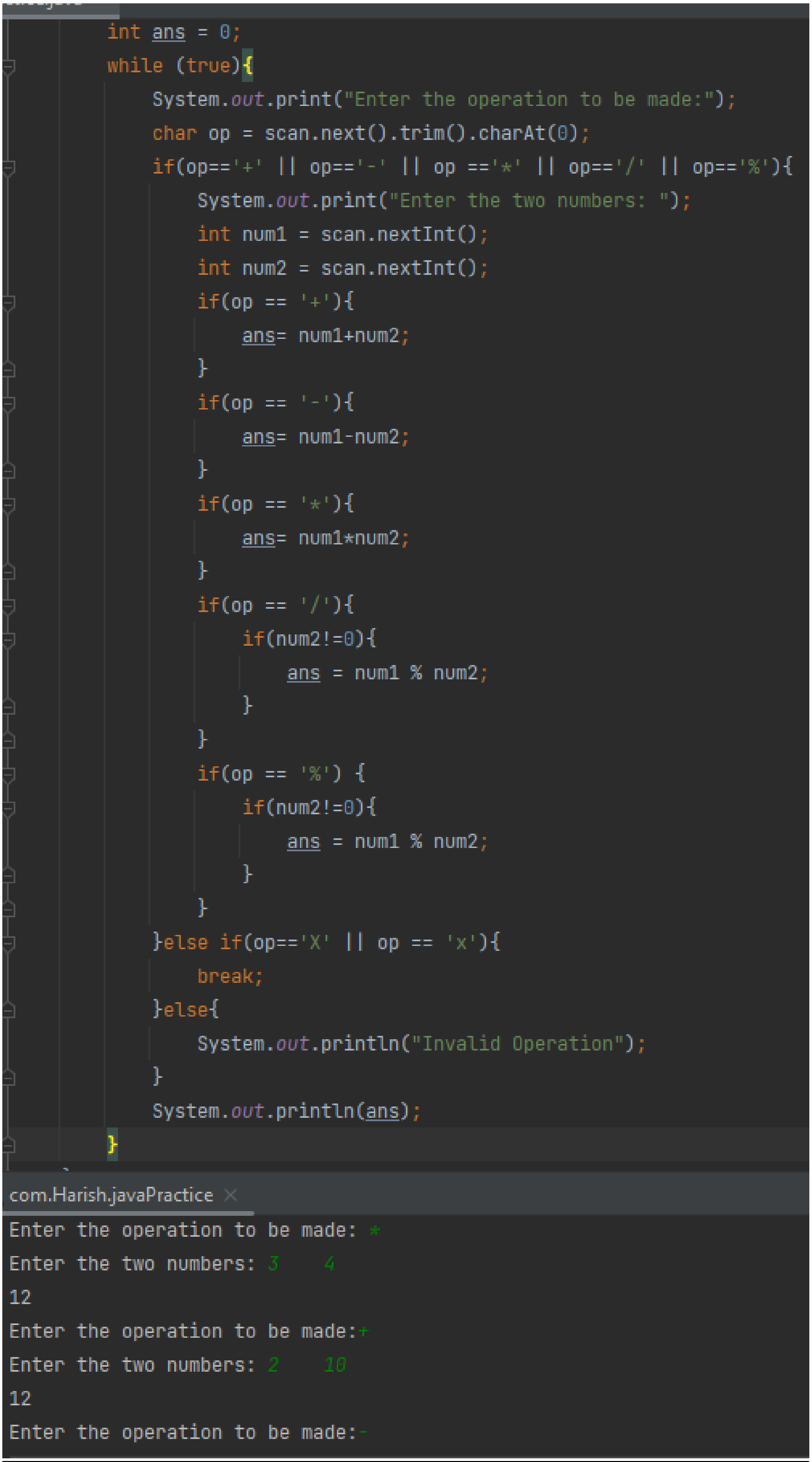
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.ex
5
1    2    3    4    5
10
20
30
40
50
```



```java
public class javaPractice {
    no usages
    public static void main(String[] args) {
        int[] arr = {1,3,23,9,18};
         System.out.println(Arrays.toString(arr));
        swap(arr, i: 1, j: 3);
         System.out.println(Arrays.toString(arr));

    }
    1 usage
    static void swap(int[] arr,int i,int j){
         int temp =0;
         temp = arr[i];
         arr[i] = arr[j];
         arr[j]=temp;
    }
}
```

com.Harish.javaPractice

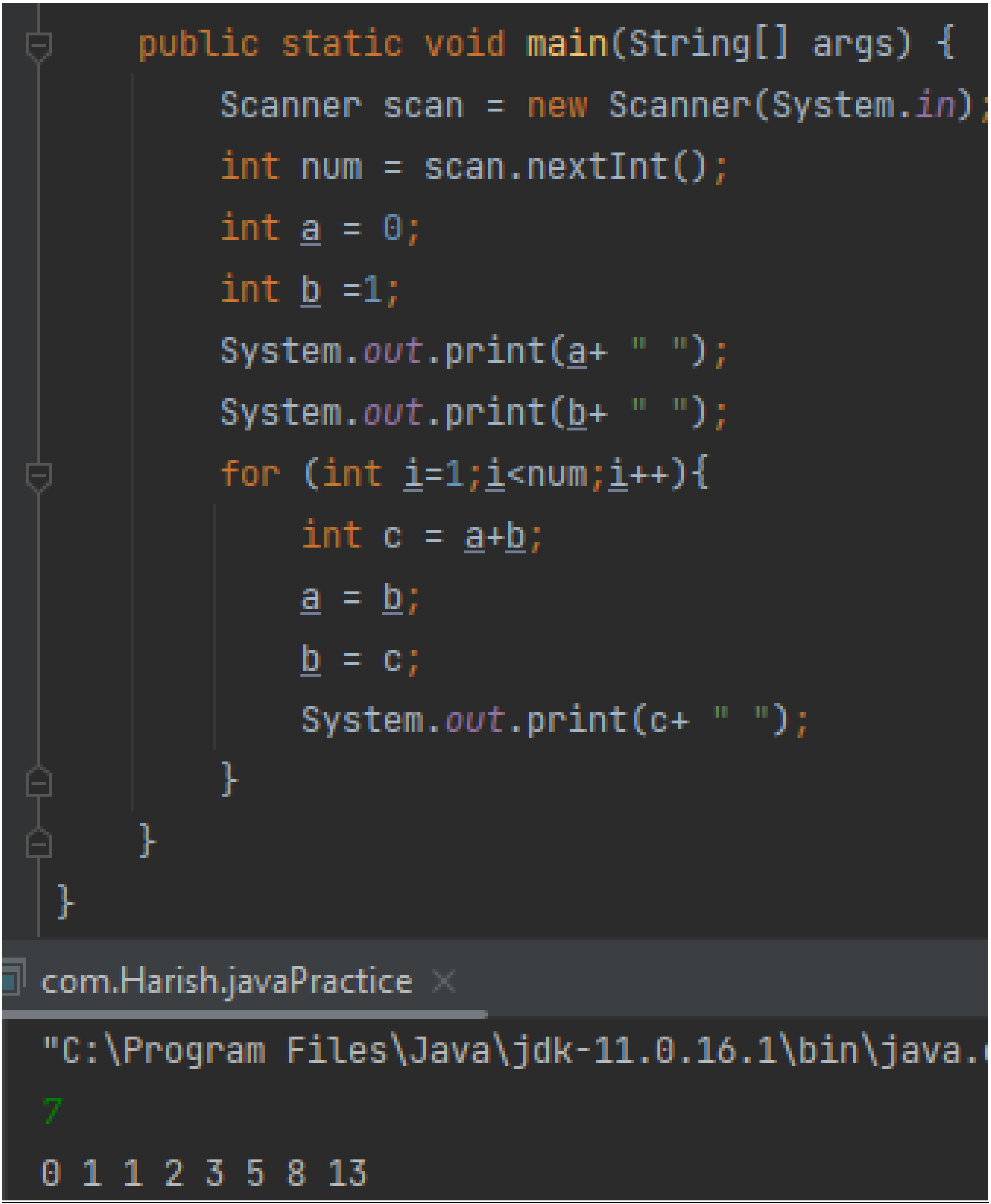
```
"C:\Users\Harish S\.jdks\openjdk-19.0.2\bin\java.exe"
[1, 3, 23, 9, 18]
[1, 9, 23, 3, 18]
```



```java
        int ans = 0;
        while (true){
            System.out.print("Enter the operation to be made:");
            char op = scan.next().trim().charAt(0);
            if(op=='+' || op=='-' || op =='*' || op=='/' || op=='%'){
                System.out.print("Enter the two numbers: ");
                int num1 = scan.nextInt();
                int num2 = scan.nextInt();
                if(op == '+'){
                    ans= num1+num2;
                }
                if(op == '-'){
                    ans= num1-num2;
                }
                if(op == '*'){
                    ans= num1*num2;
                }
                if(op == '/'){
                    if(num2!=0){
                        ans = num1 % num2;
                    }
                }
                if(op == '%') {
                    if(num2!=0){
                        ans = num1 % num2;
                    }
                }
            }else if(op=='X' || op == 'x'){
                break;
            }else{
                System.out.println("Invalid Operation");
            }
            System.out.println(ans);
        }
```

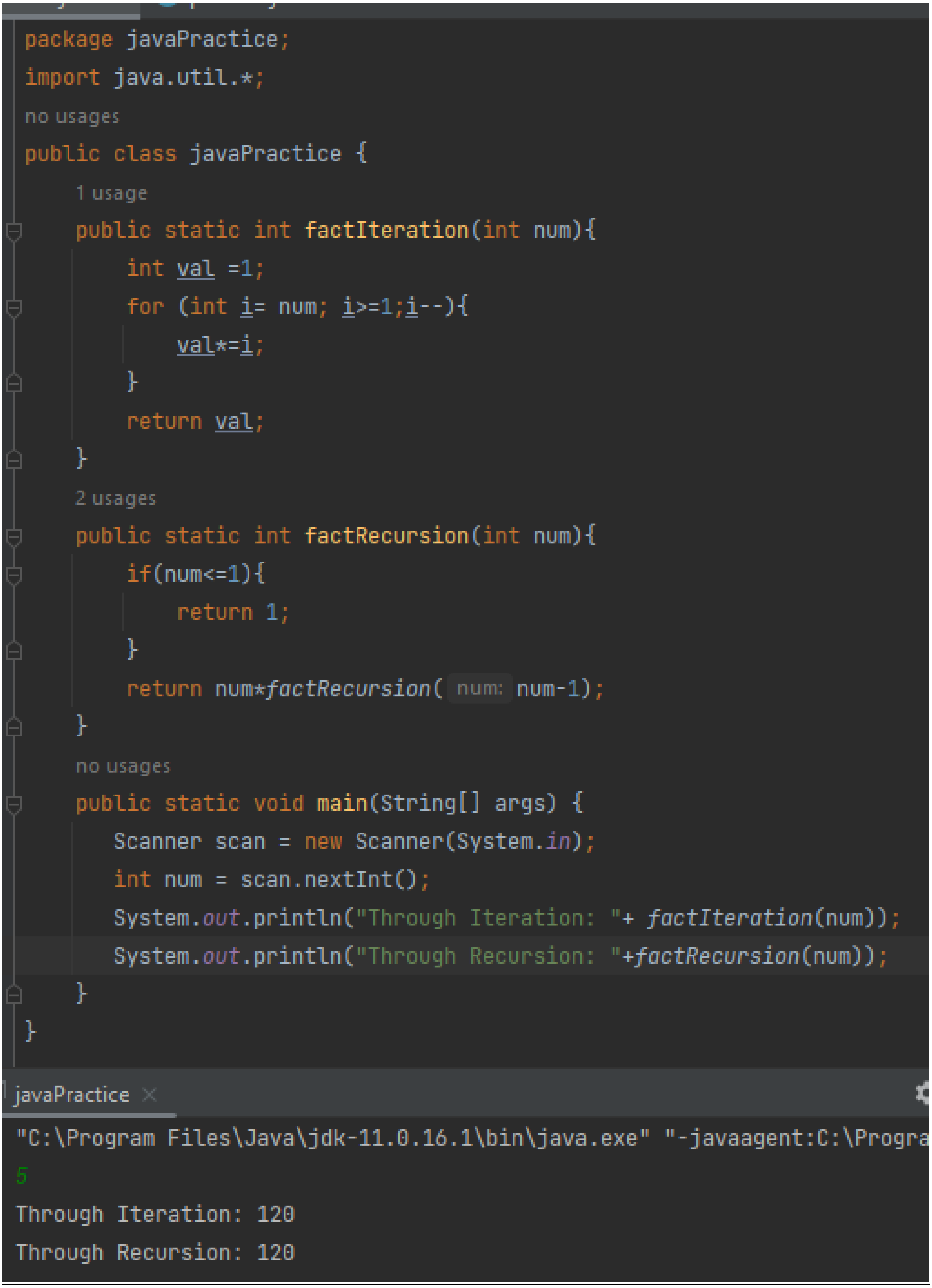
com.Harish.javaPractice

```
Enter the operation to be made: *
Enter the two numbers: 3    4
12
Enter the operation to be made:+
Enter the two numbers: 2    10
12
Enter the operation to be made:-
```



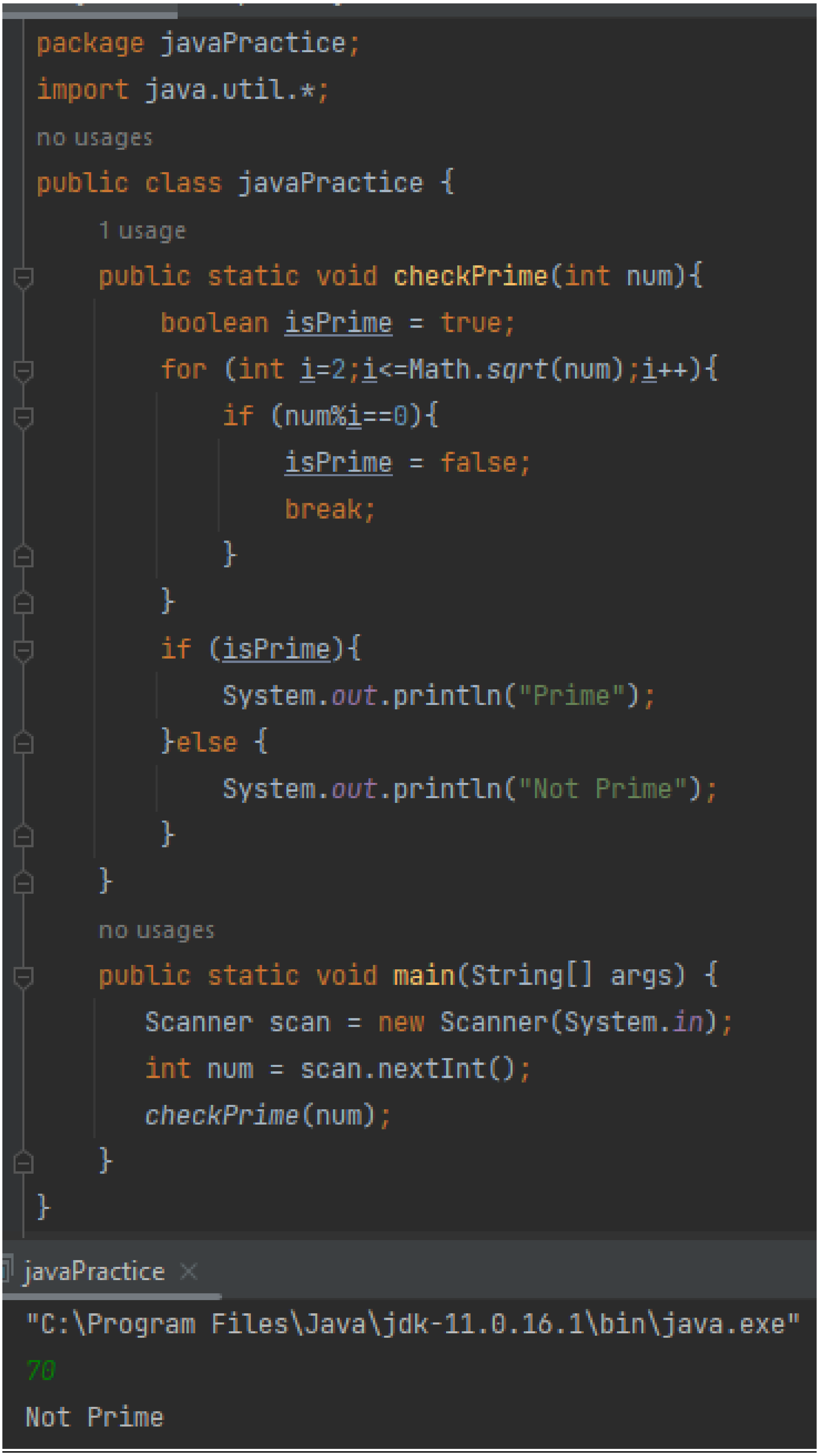
```
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
        int a = 0;
        int b =1;
        System.out.print(a+ " ");
        System.out.print(b+ " ");
        for (int i=1;i<num;i++){
            int c = a+b;
            a = b;
            b = c;
            System.out.print(c+ " ");
        }
    }
}
```

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.
7
0 1 1 2 3 5 8 13
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    1 usage
    public static int factIteration(int num){
        int val =1;
        for (int i= num; i>=1;i--){
            val*=i;
        }
        return val;
    }
    2 usages
    public static int factRecursion(int num){
        if(num<=1){
            return 1;
        }
        return num*factRecursion( num: num-1);
    }
    no usages
    public static void main(String[] args) {
       Scanner scan = new Scanner(System.in);
       int num = scan.nextInt();
       System.out.println("Through Iteration: "+ factIteration(num));
       System.out.println("Through Recursion: "+factRecursion(num));
    }
}
```

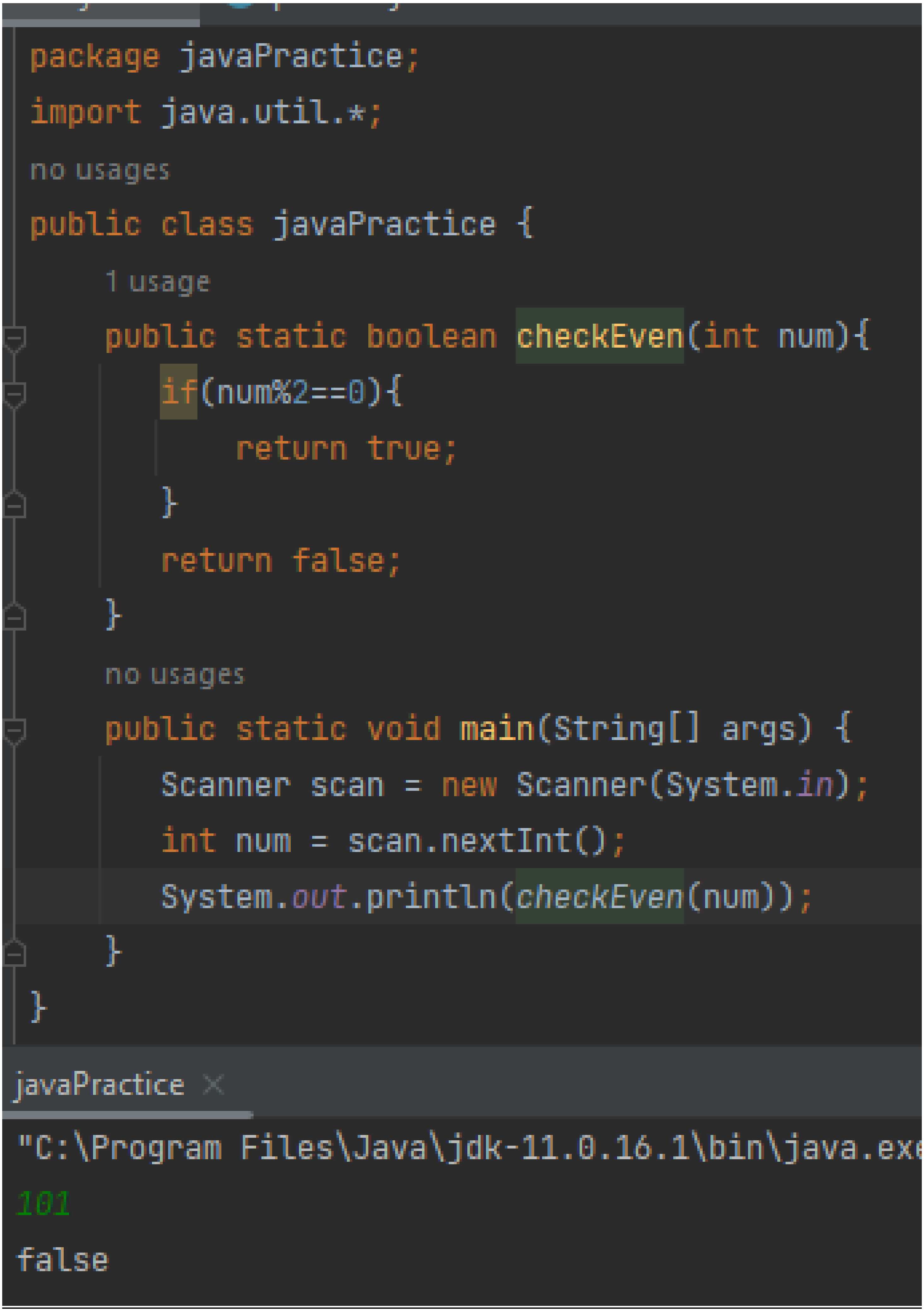
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "-javaagent:C:\Progra
5
Through Iteration: 120
Through Recursion: 120
```



```java
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    1 usage
    public static void checkPrime(int num){
        boolean isPrime = true;
        for (int i=2;i<=Math.sqrt(num);i++){
            if (num%i==0){
                isPrime = false;
                break;
            }
        }
        if (isPrime){
            System.out.println("Prime");
        }else {
            System.out.println("Not Prime");
        }
    }
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
        checkPrime(num);
    }
}
```

javaPractice

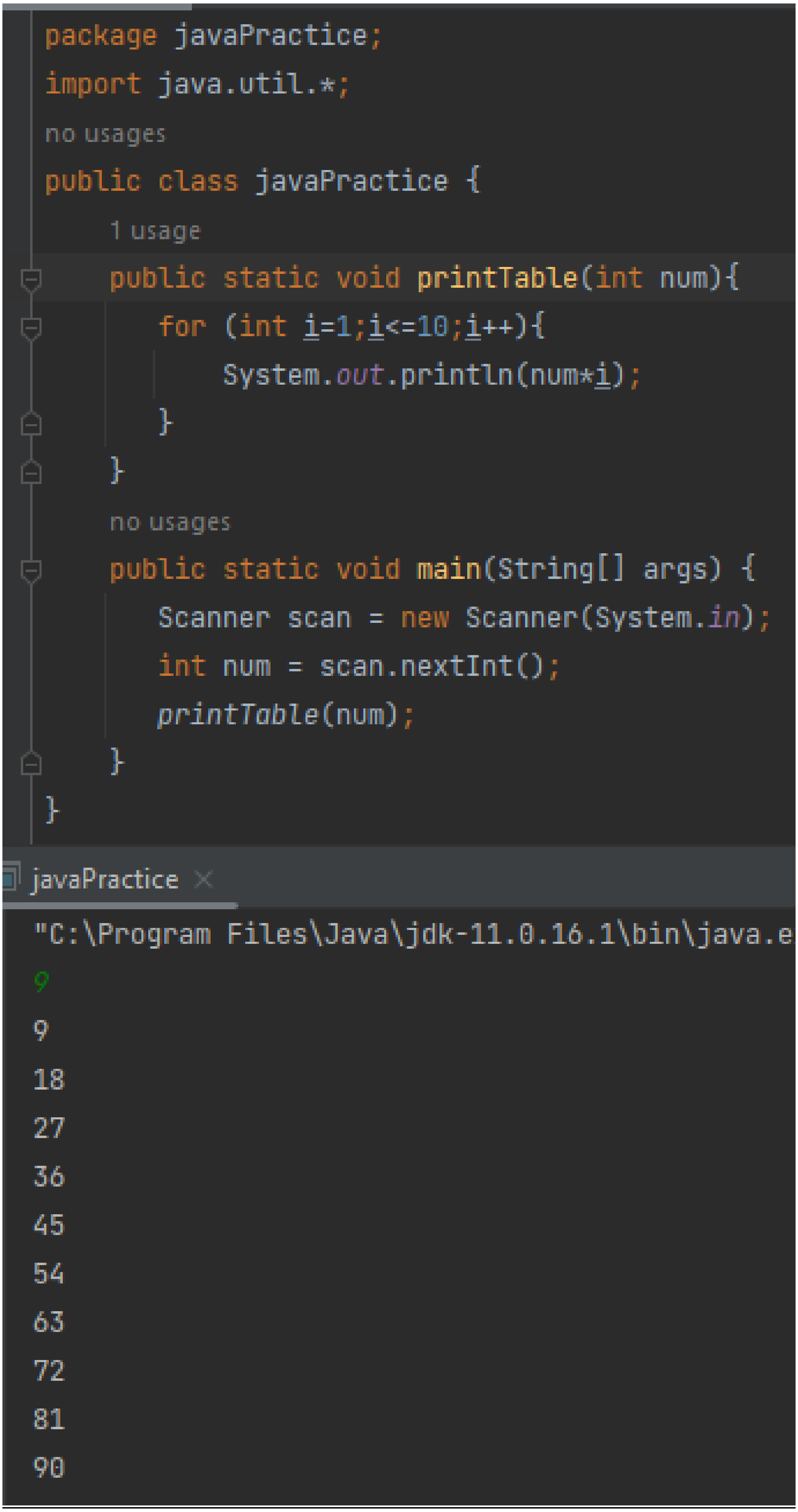
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe"
70
Not Prime
```



```java
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    1 usage
    public static boolean checkEven(int num){
        if(num%2==0){
            return true;
        }
        return false;
    }
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
        System.out.println(checkEven(num));
    }
}
```

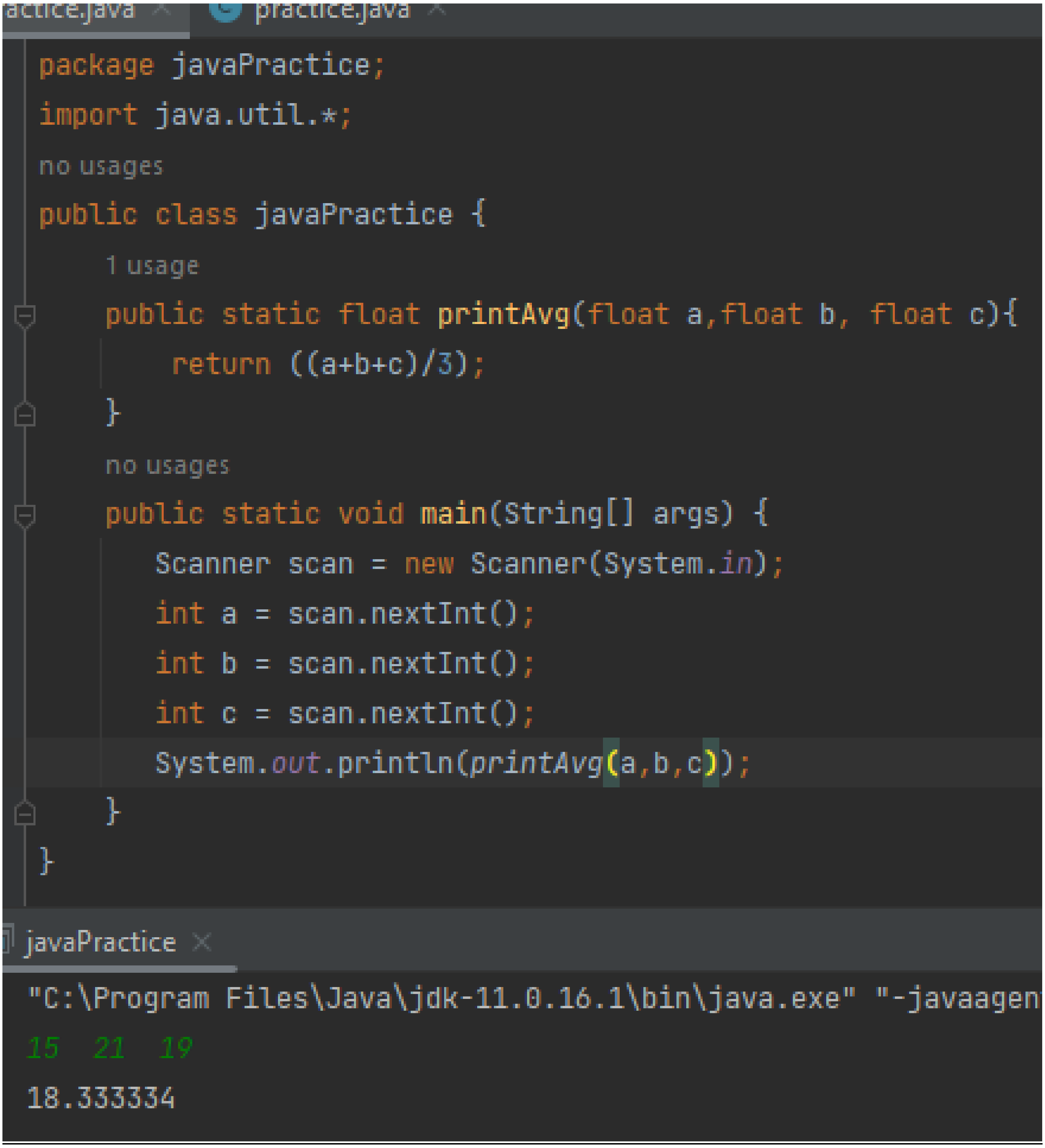
javaPractice

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
101
false
```



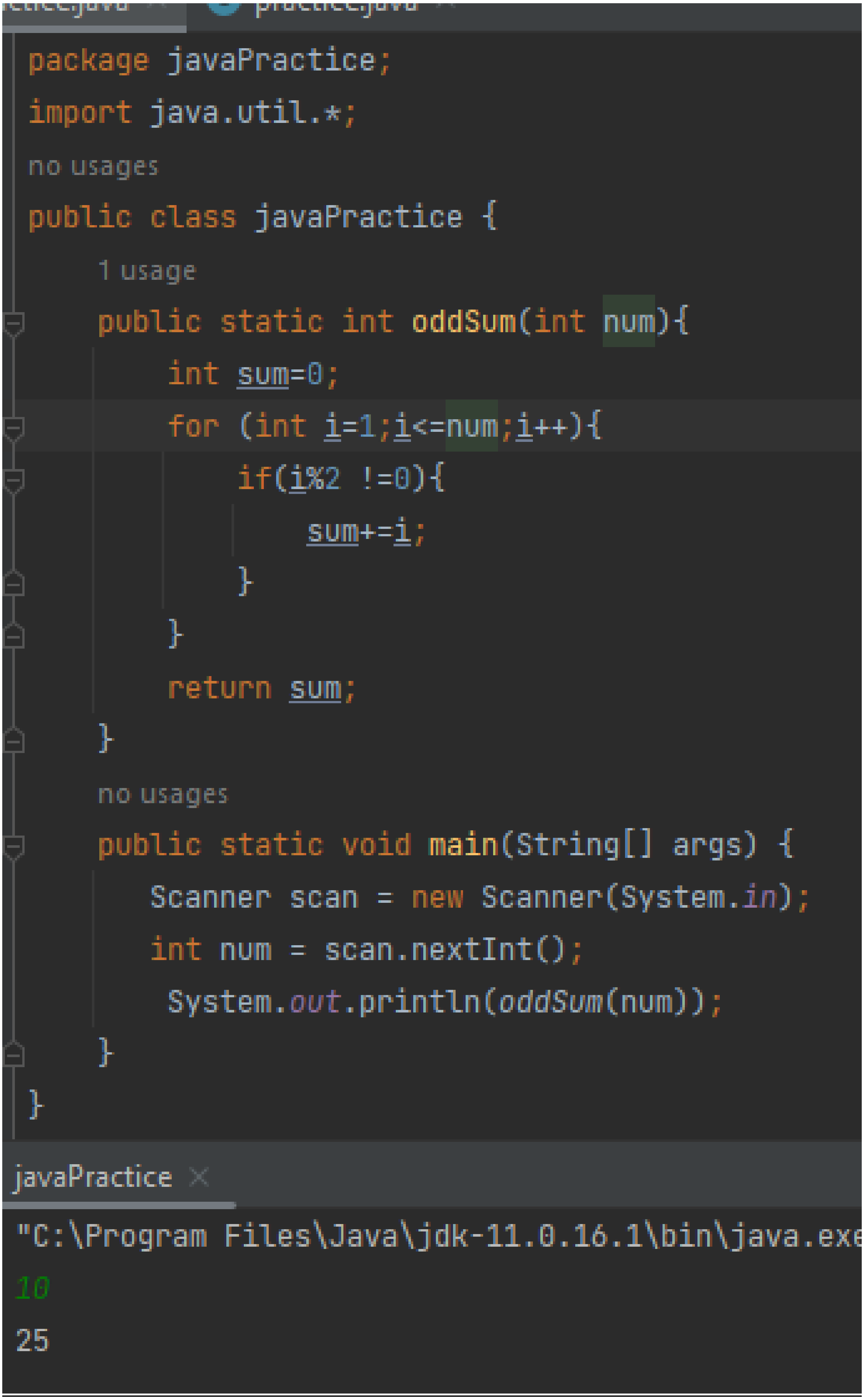
```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    1 usage
    public static void printTable(int num){
        for (int i=1;i<=10;i++){
            System.out.println(num*i);
        }
    }
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
        printTable(num);
    }
}
```

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.e
9
9
18
27
36
45
54
63
72
81
90
```



```java
package javaPractice;
import java.util.*;
public class javaPractice {
    public static float printAvg(float a,float b, float c){
        return ((a+b+c)/3);
    }
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int a = scan.nextInt();
        int b = scan.nextInt();
        int c = scan.nextInt();
        System.out.println(printAvg(a,b,c));
    }
}
```

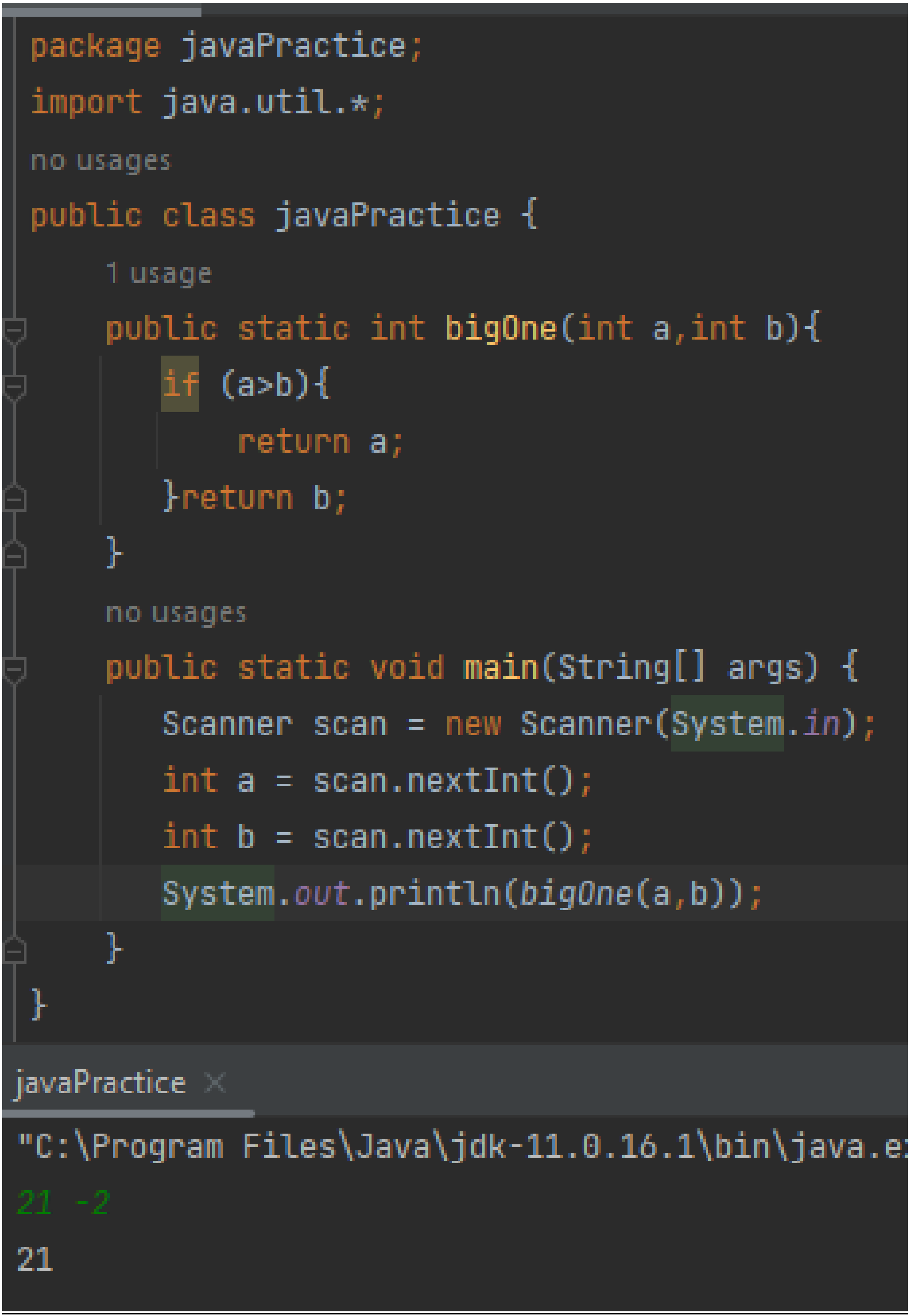
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "-javaagen
15  21  19
18.333334
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    1 usage
    public static int oddSum(int num){
        int sum=0;
        for (int i=1;i<=num;i++){
            if(i%2 !=0){
                sum+=i;
            }
        }
        return sum;
    }
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
         System.out.println(oddSum(num));
    }
}
```

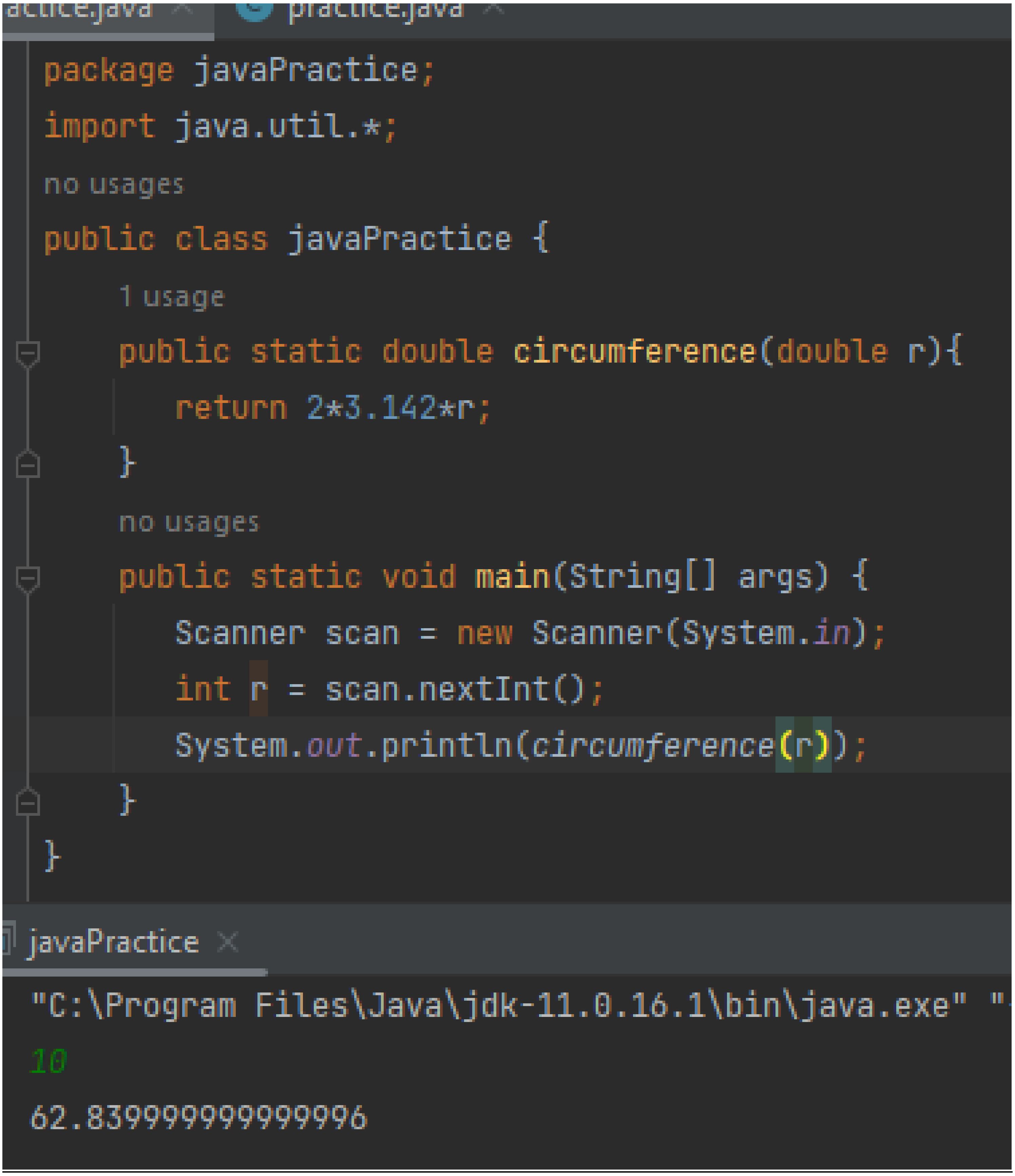
javaPractice

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
10
25
```



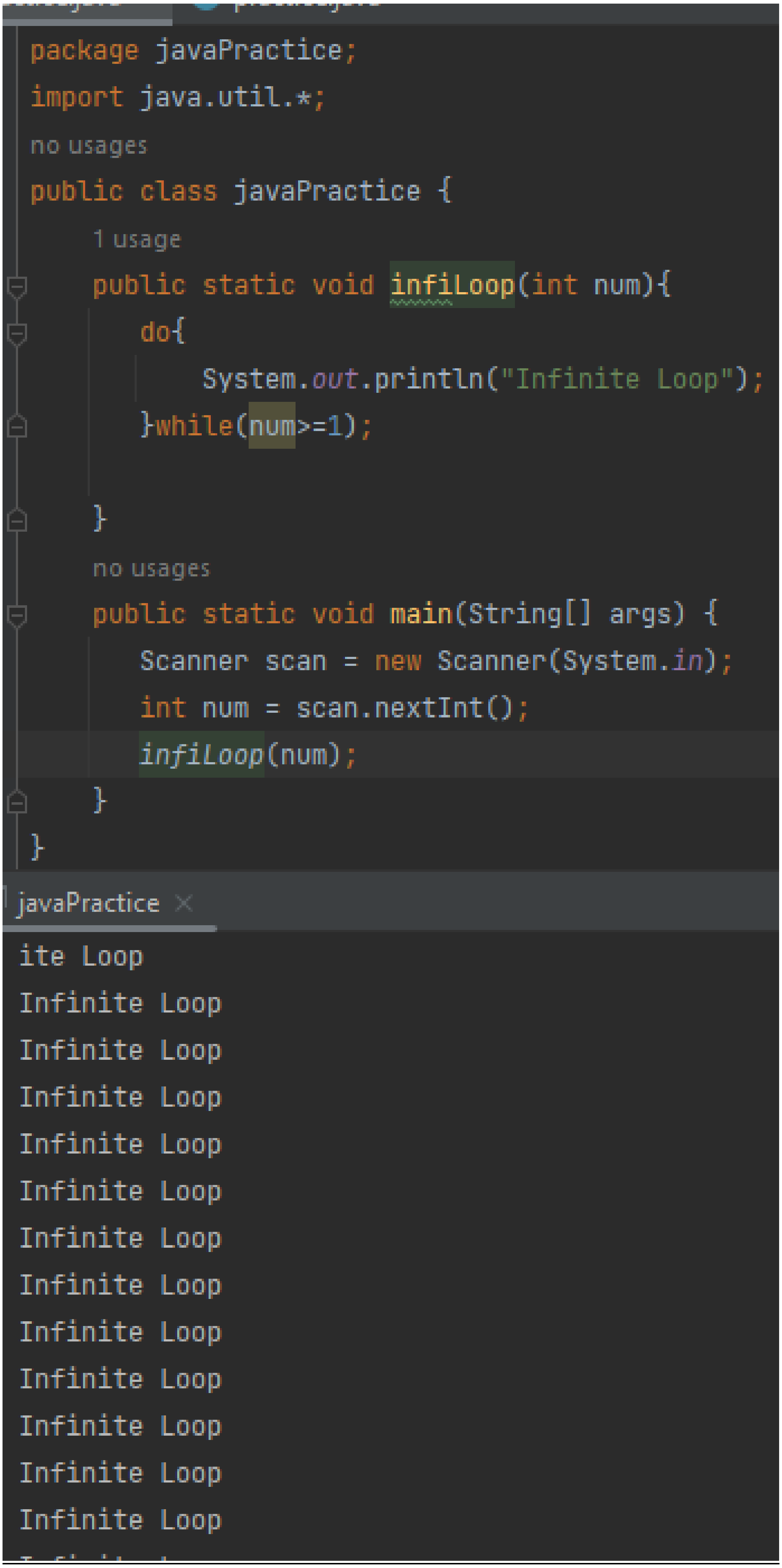
```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    1 usage
    public static int bigOne(int a,int b){
        if (a>b){
            return a;
        }return b;
    }
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int a = scan.nextInt();
        int b = scan.nextInt();
        System.out.println(bigOne(a,b));
    }
}
```

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.e
21 -2
21
```



```java
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    1 usage
    public static double circumference(double r){
        return 2*3.142*r;
    }
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int r = scan.nextInt();
        System.out.println(circumference(r));
    }
}
```

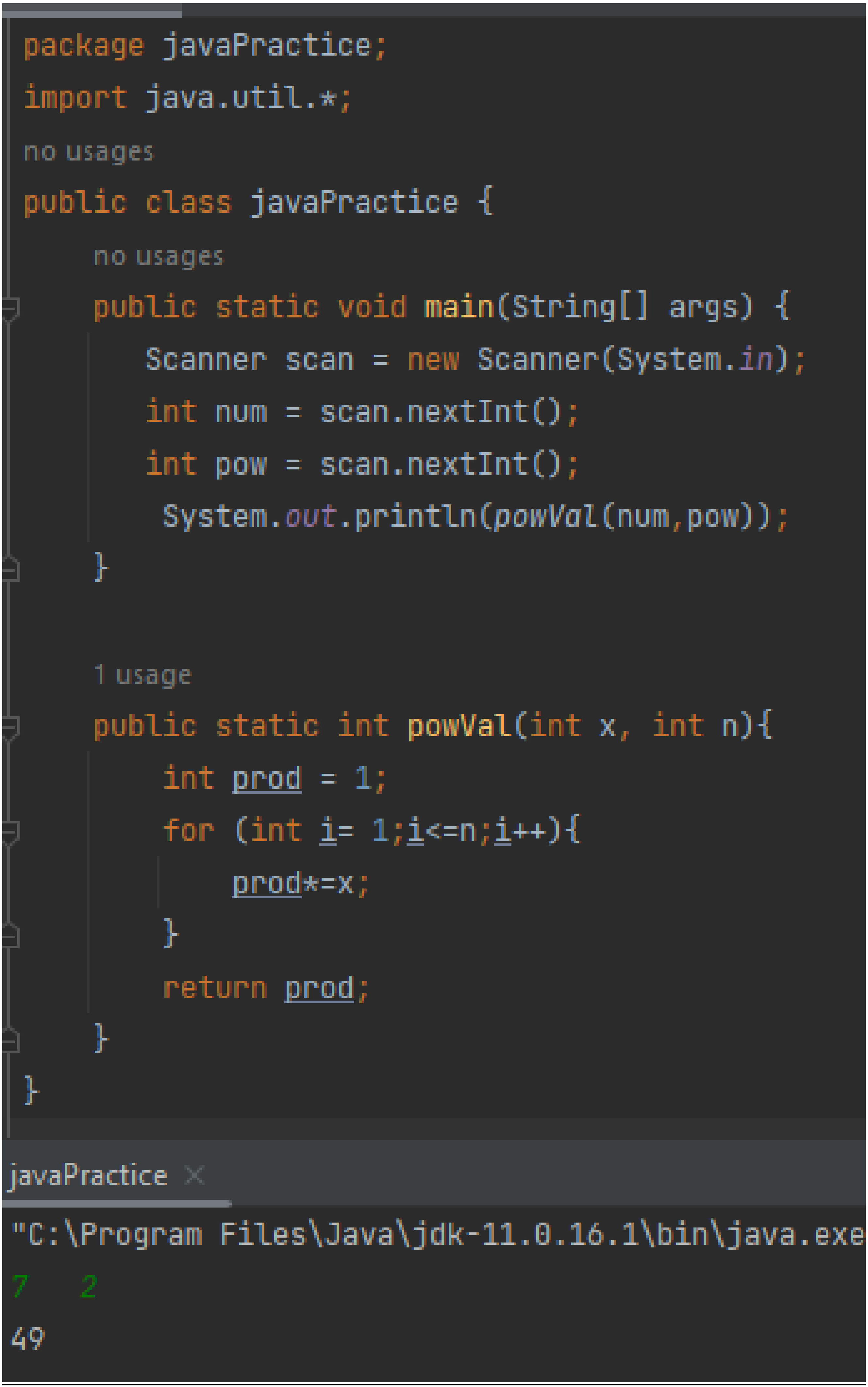
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "
10
62.839999999999996
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    1 usage
    public static void infiLoop(int num){
        do{
            System.out.println("Infinite Loop");
        }while(num>=1);

    }
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
        infiLoop(num);
    }
}
```

javaPractice

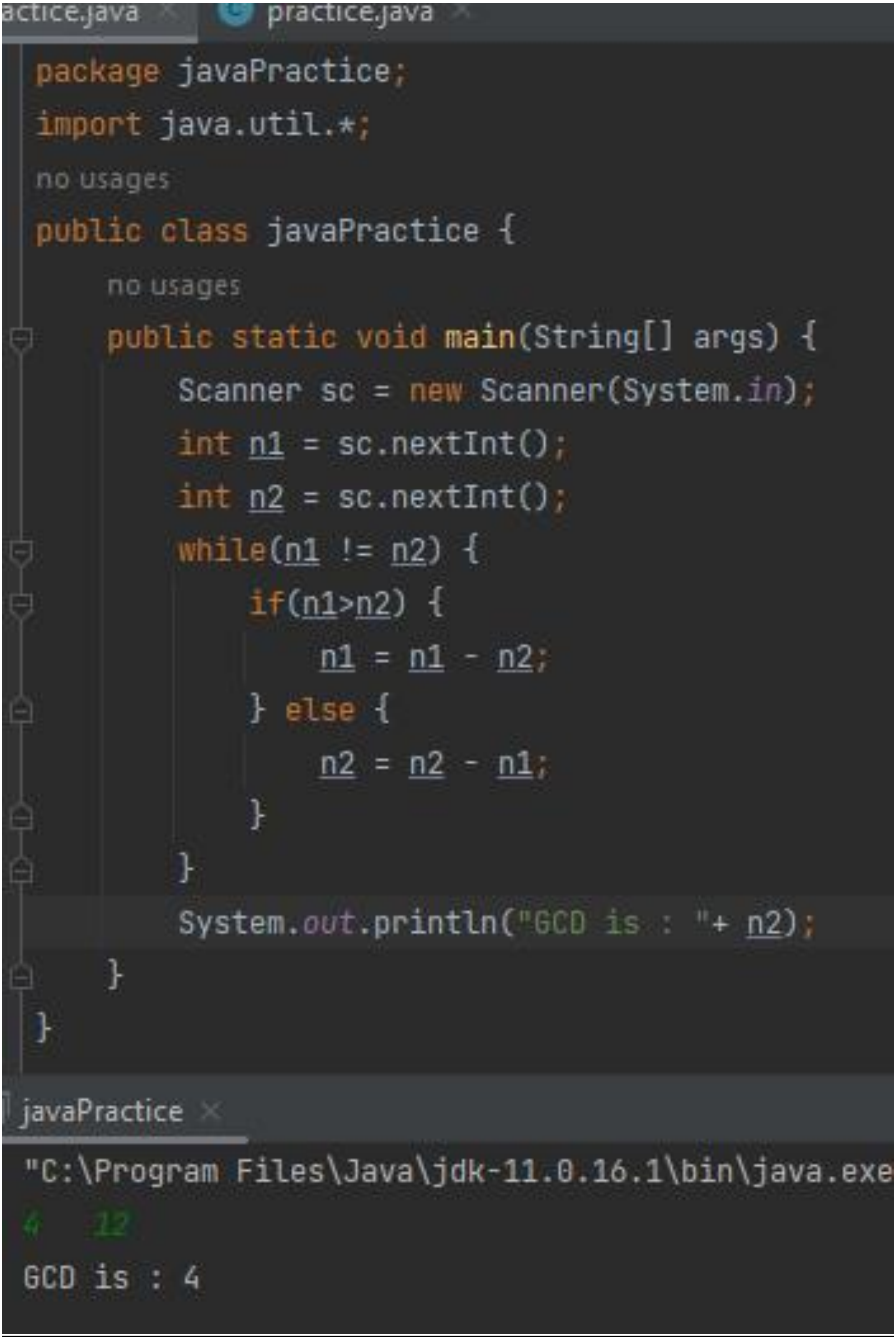
```
ite Loop
Infinite Loop
Infinite Loop
Infinite Loop
Infinite Loop
Infinite Loop
Infinite Loop
Infinite Loop
Infinite Loop
Infinite Loop
Infinite Loop
Infinite Loop
Infinite Loop
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
        int pow = scan.nextInt();
         System.out.println(powVal(num,pow));
    }

    1 usage
    public static int powVal(int x, int n){
        int prod = 1;
        for (int i= 1;i<=n;i++){
            prod*=x;
        }
        return prod;
    }
}
```

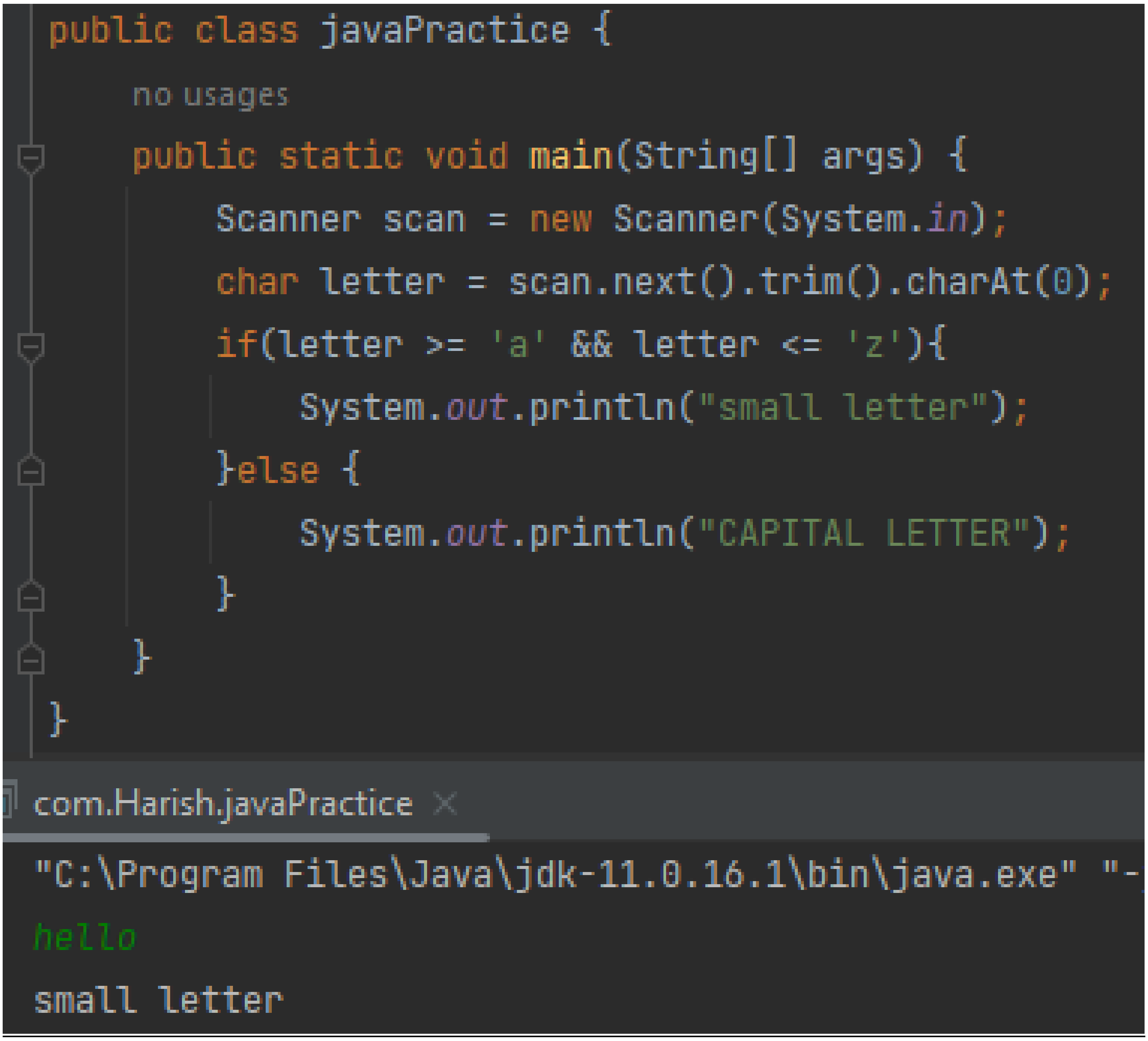
javaPractice

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
7   2
49
```



```java
package javaPractice;
import java.util.*;
public class javaPractice {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n1 = sc.nextInt();
        int n2 = sc.nextInt();
        while(n1 != n2) {
            if(n1>n2) {
                n1 = n1 - n2;
            } else {
                n2 = n2 - n1;
            }
        }
        System.out.println("GCD is : "+ n2);
    }
}
```

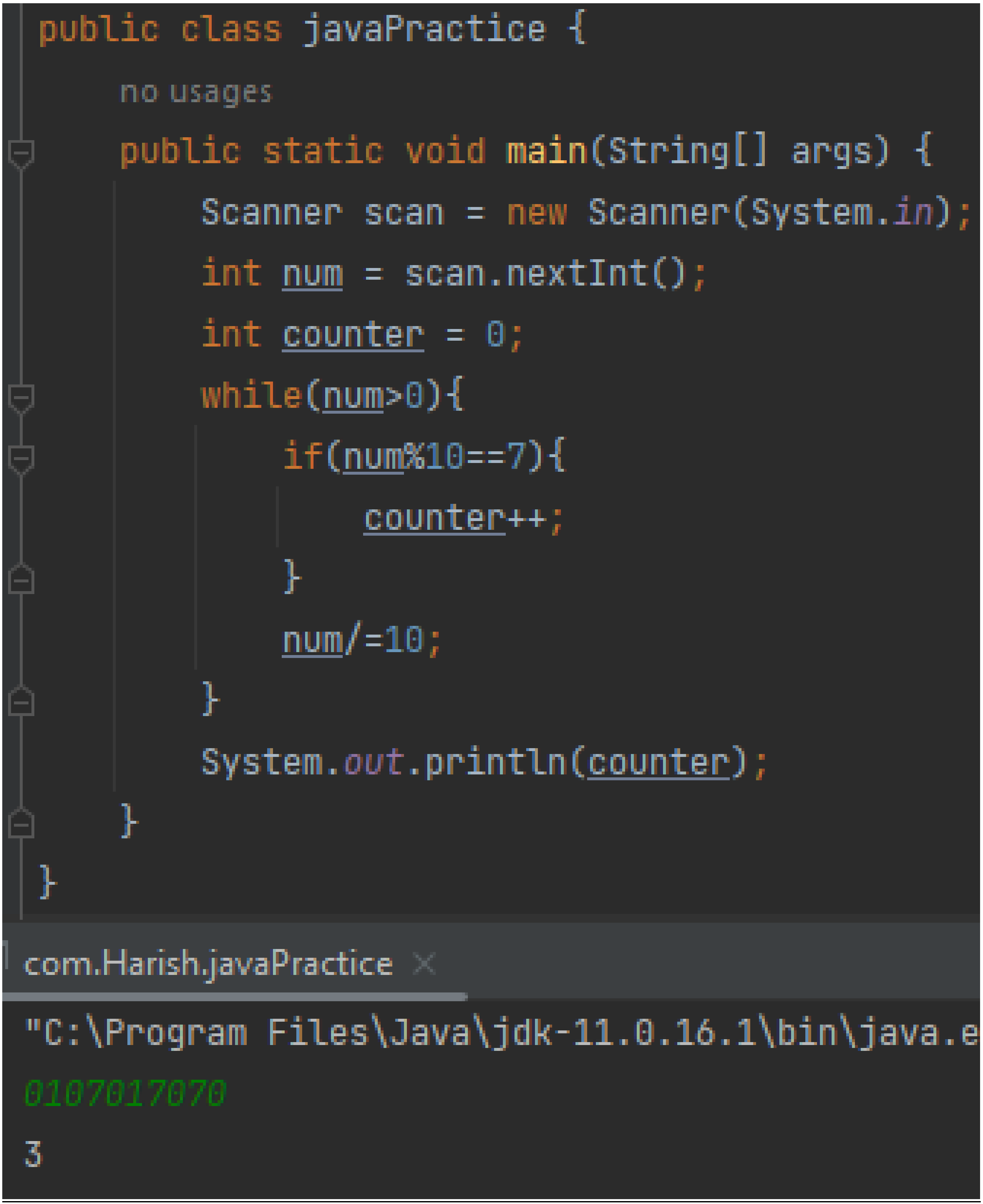
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
4  12
GCD is : 4
```



```java
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        char letter = scan.next().trim().charAt(0);
        if(letter >= 'a' && letter <= 'z'){
            System.out.println("small letter");
        }else {
            System.out.println("CAPITAL LETTER");
        }
    }
}
```

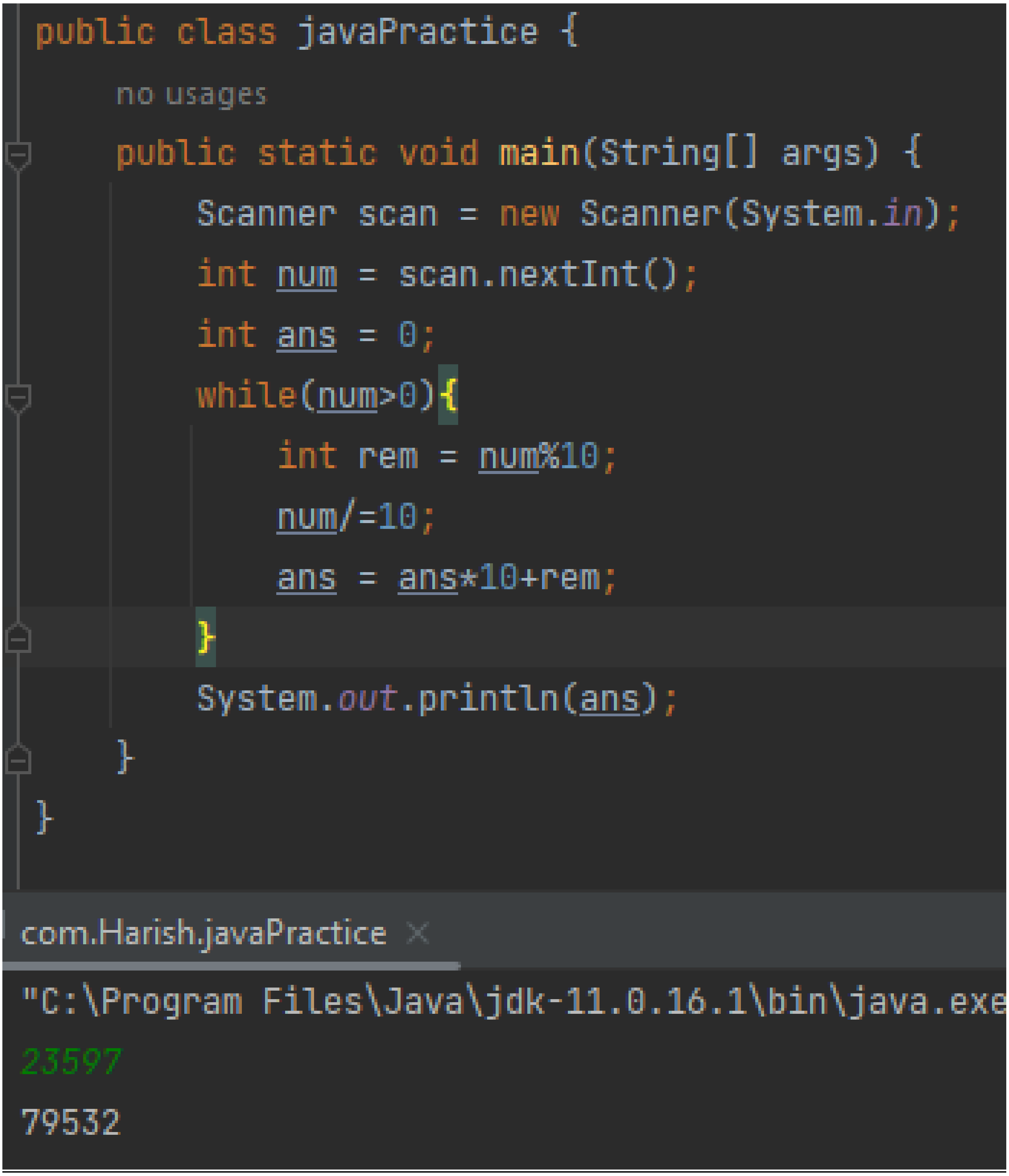
com.Harish.javaPractice

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "-
hello
small letter
```



```java
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
        int counter = 0;
        while(num>0){
            if(num%10==7){
                counter++;
            }
            num/=10;
        }
        System.out.println(counter);
    }
}
```

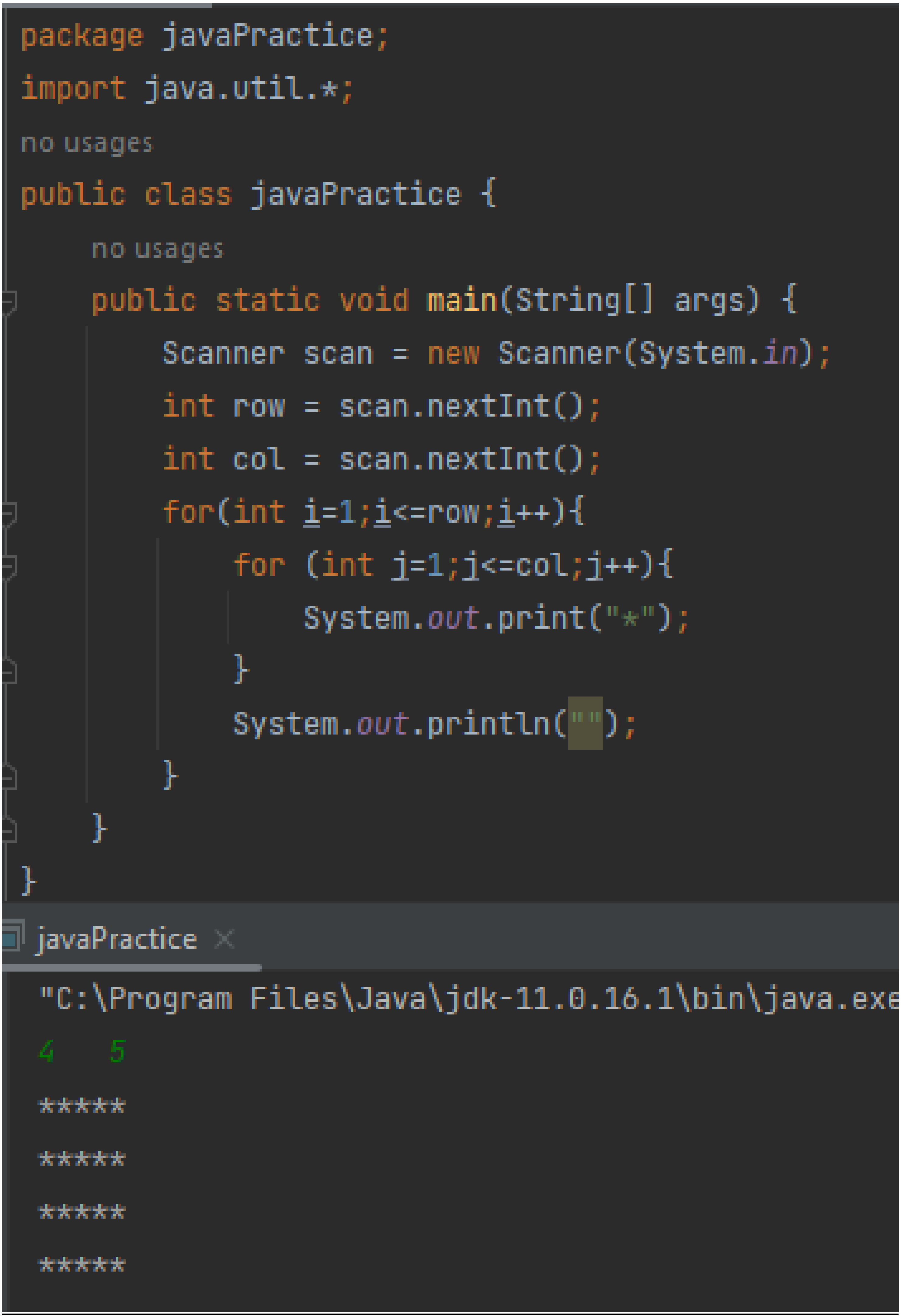
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.e
0107017070
3
```



```java
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
        int ans = 0;
        while(num>0){
            int rem = num%10;
            num/=10;
            ans = ans*10+rem;
        }
        System.out.println(ans);
    }
}
```

com.Harish.javaPractice

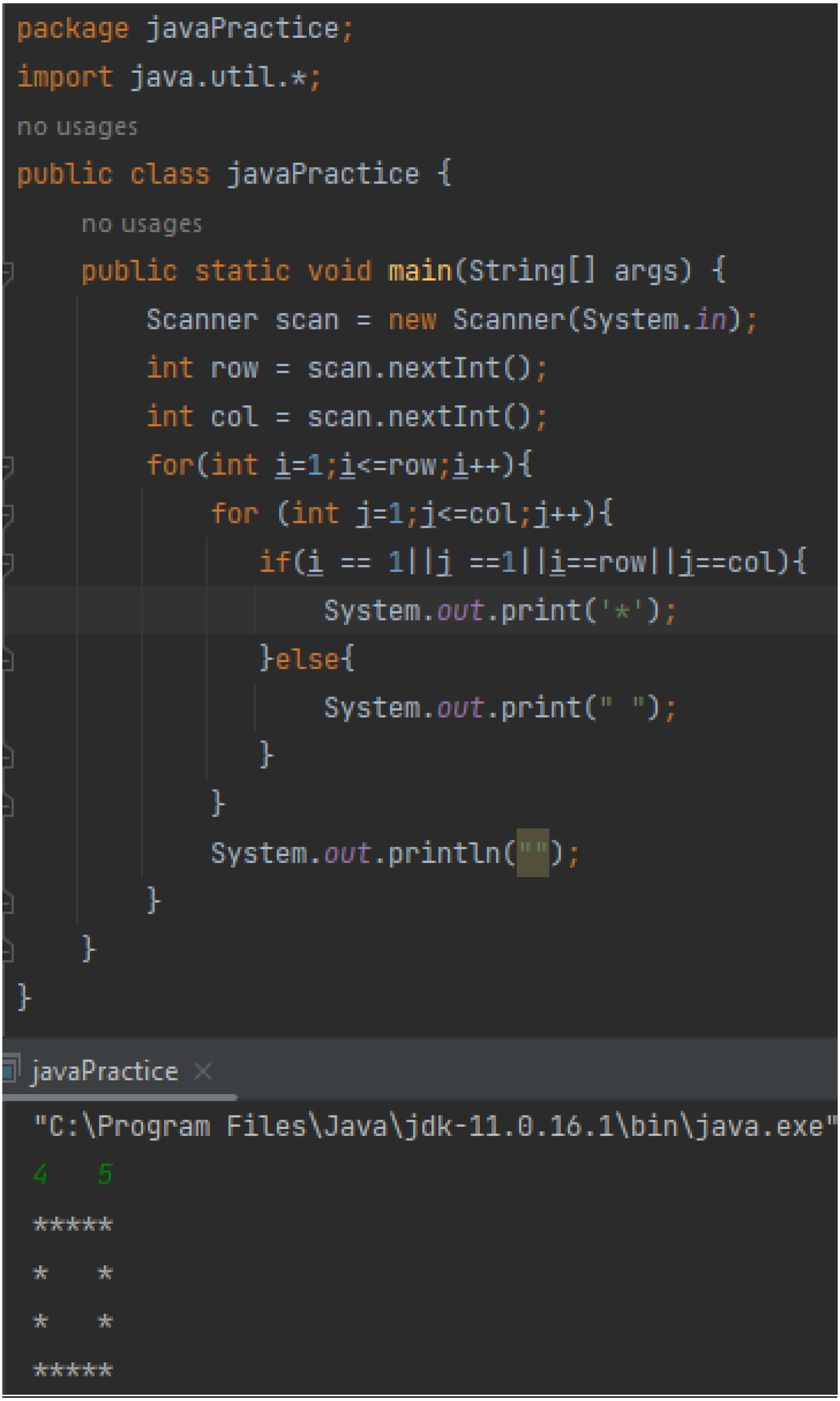
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
23597
79532
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int row = scan.nextInt();
        int col = scan.nextInt();
        for(int i=1;i<=row;i++){
            for (int j=1;j<=col;j++){
                System.out.print("*");
            }
            System.out.println("");
        }
    }
}
```

javaPractice

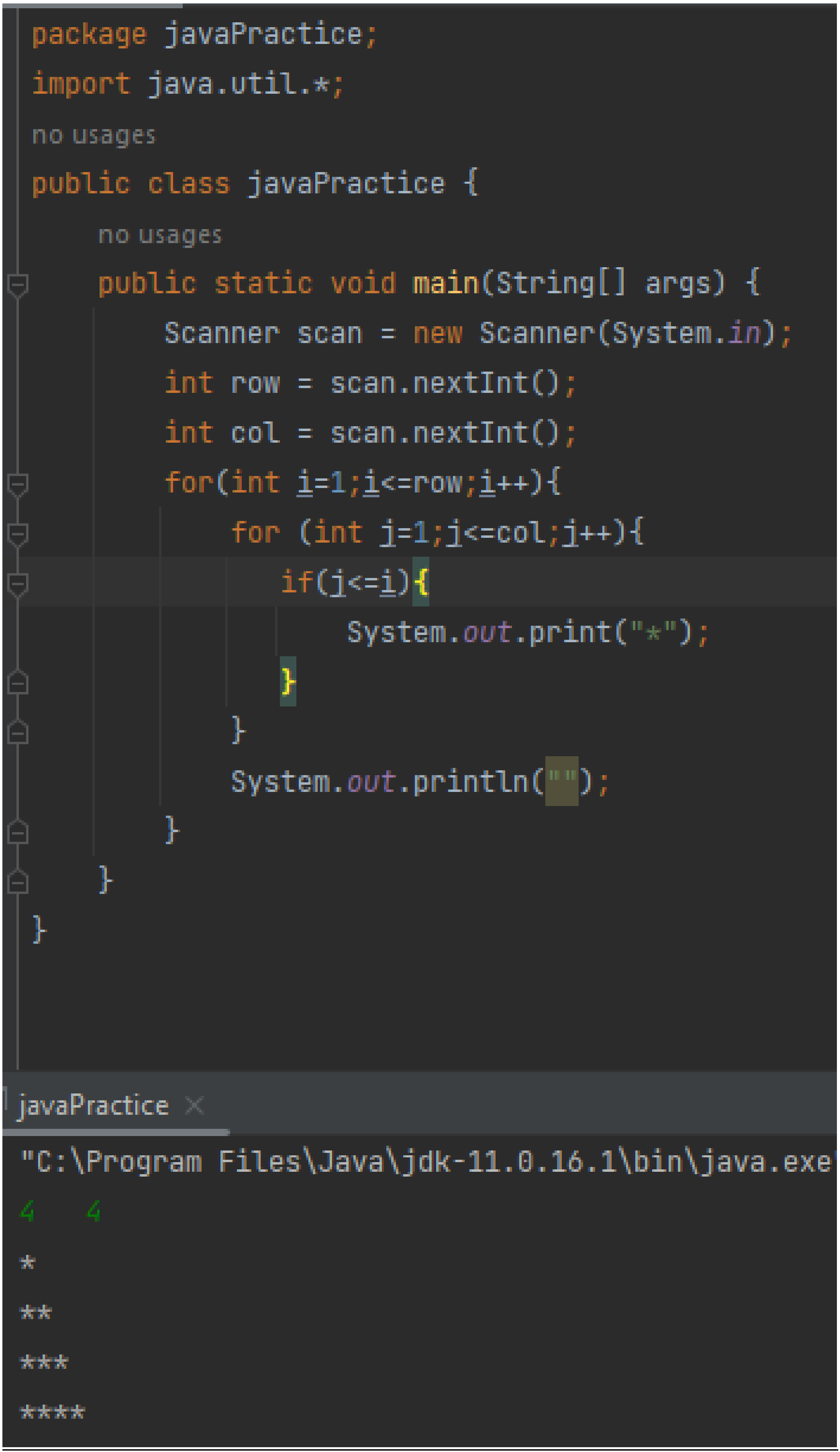
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
4   5
*****
*****
*****
*****
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int row = scan.nextInt();
        int col = scan.nextInt();
        for(int i=1;i<=row;i++){
            for (int j=1;j<=col;j++){
               if(i == 1||j ==1||i==row||j==col){
                   System.out.print('*');
               }else{
                   System.out.print(" ");
               }
            }
            System.out.println("");
        }
    }
}
```

javaPractice

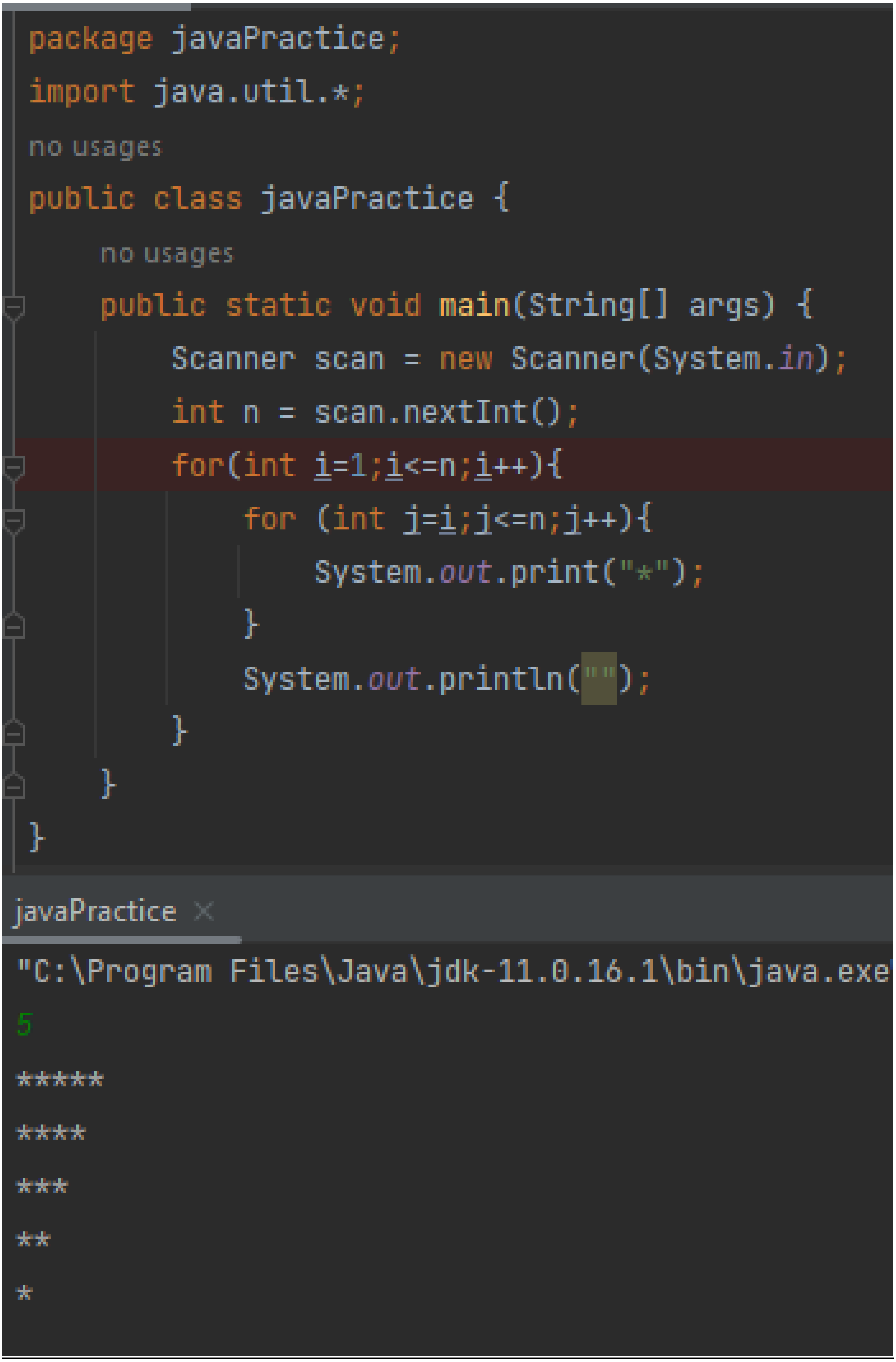
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe"
4   5
*****
*   *
*   *
*****
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int row = scan.nextInt();
        int col = scan.nextInt();
        for(int i=1;i<=row;i++){
            for (int j=1;j<=col;j++){
               if(j<=i){
                   System.out.print("*");
               }
            }
            System.out.println("");
        }
    }
}
```

javaPractice

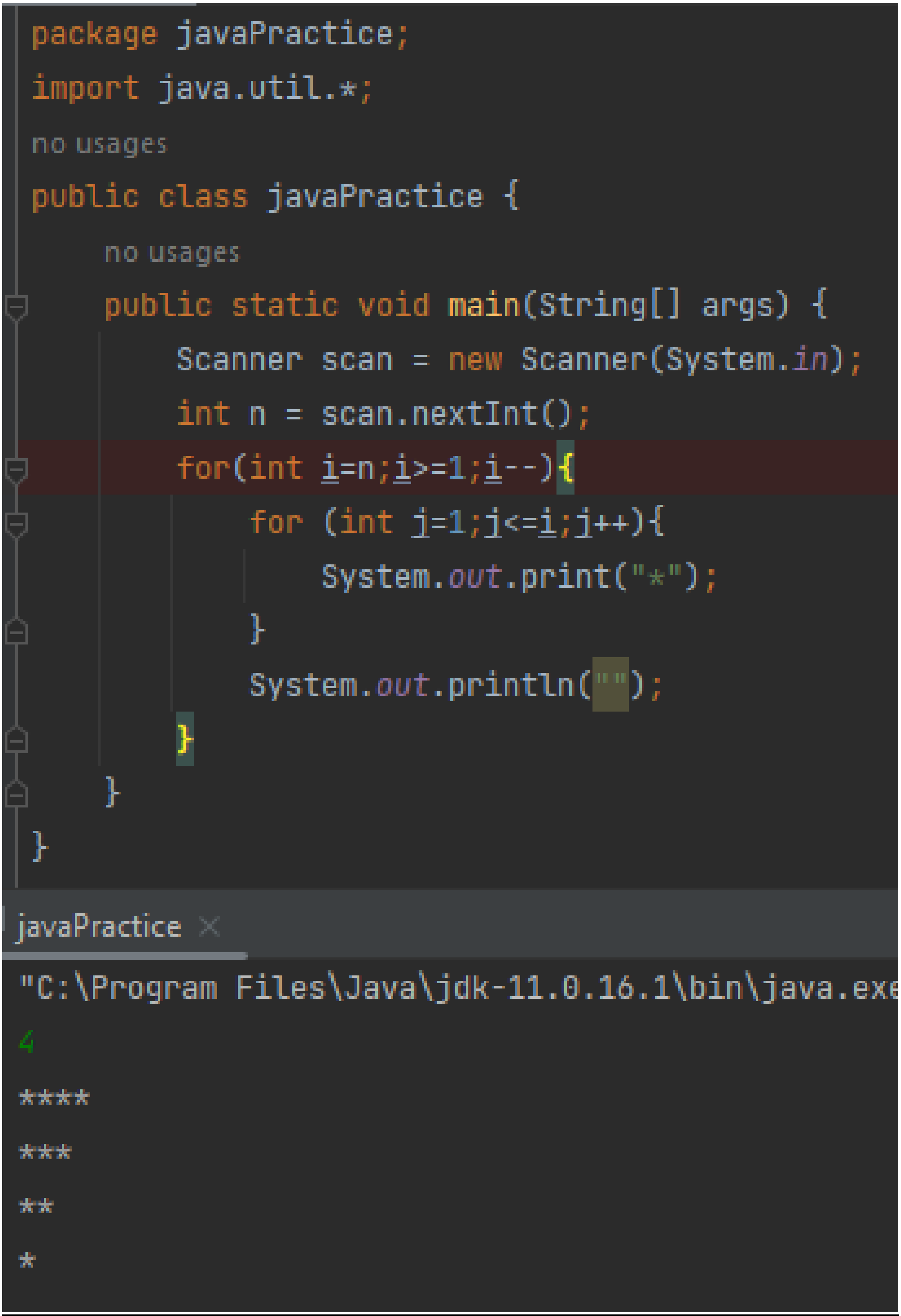
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
4   4
*
**
***
****
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for(int i=1;i<=n;i++){
            for (int j=i;j<=n;j++){
                System.out.print("*");
            }
            System.out.println("");
        }
    }
}
```

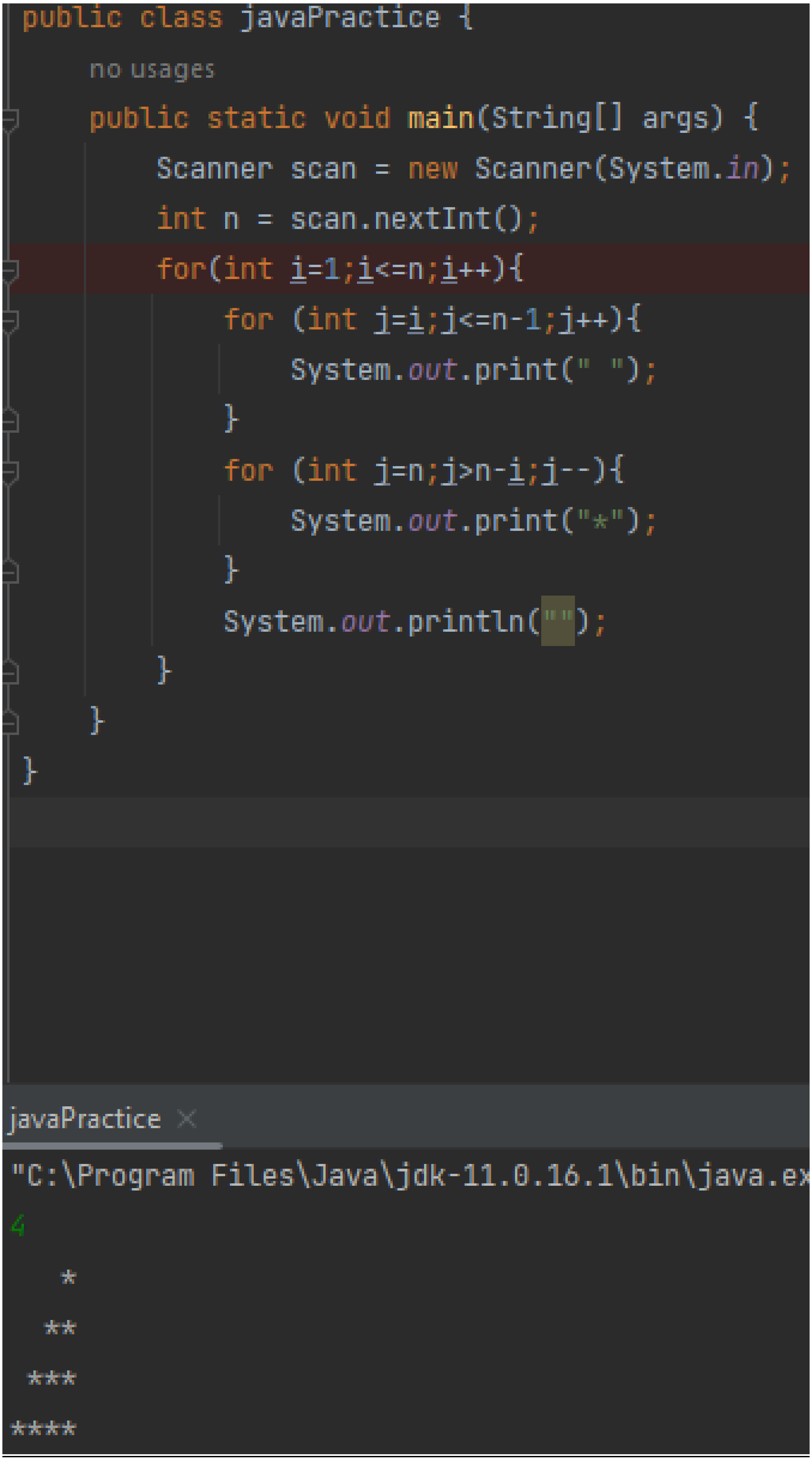
javaPractice

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
5
*****
****
***
**
*
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for(int i=n;i>=1;i--){
            for (int j=1;j<=i;j++){
                System.out.print("*");
            }
            System.out.println("");
        }
    }
}
```

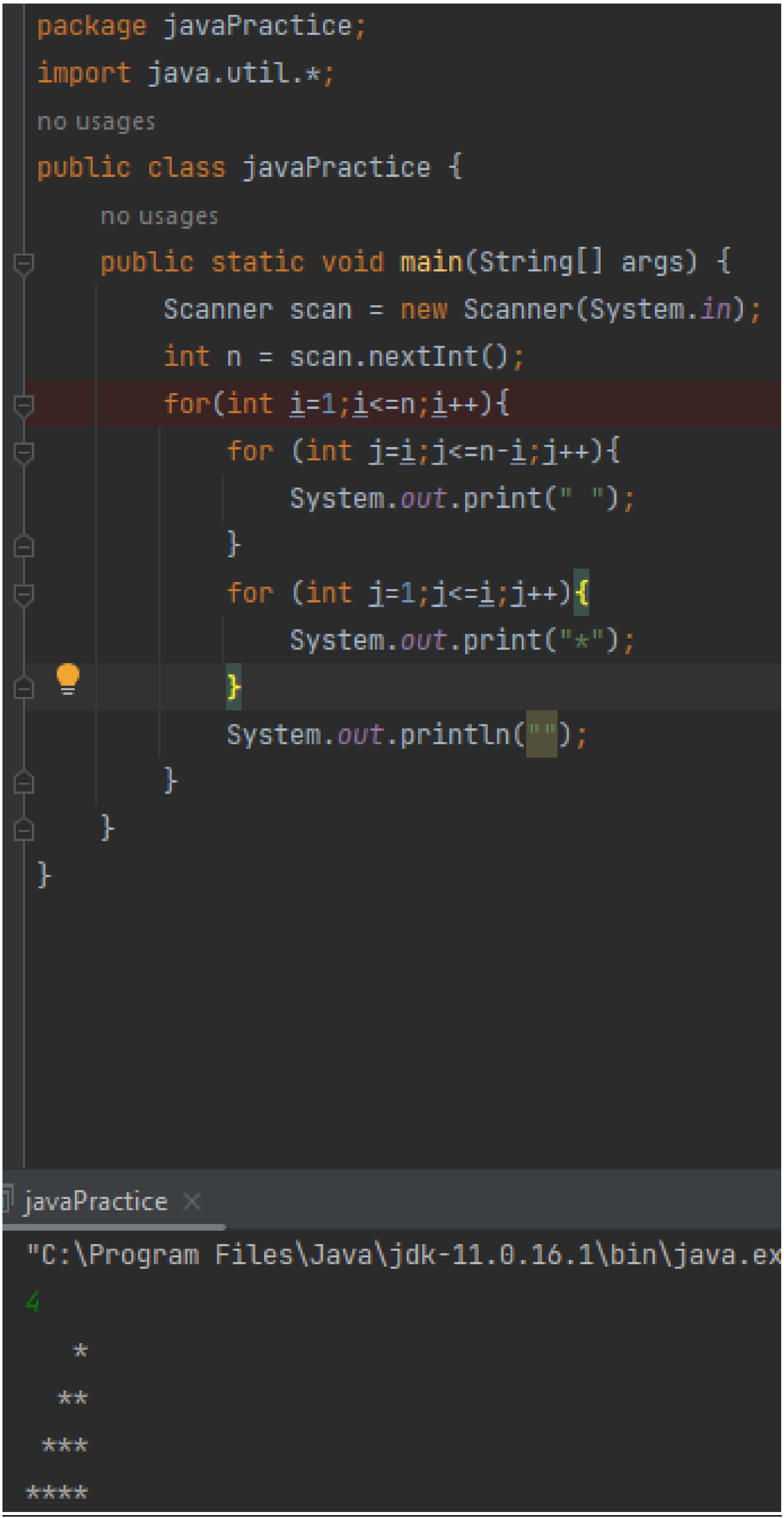
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
4
****
***
**
*
```



```java
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for(int i=1;i<=n;i++){
            for (int j=i;j<=n-1;j++){
                System.out.print(" ");
            }
            for (int j=n;j>n-i;j--){
                System.out.print("*");
            }
            System.out.println("");
        }
    }
}
```

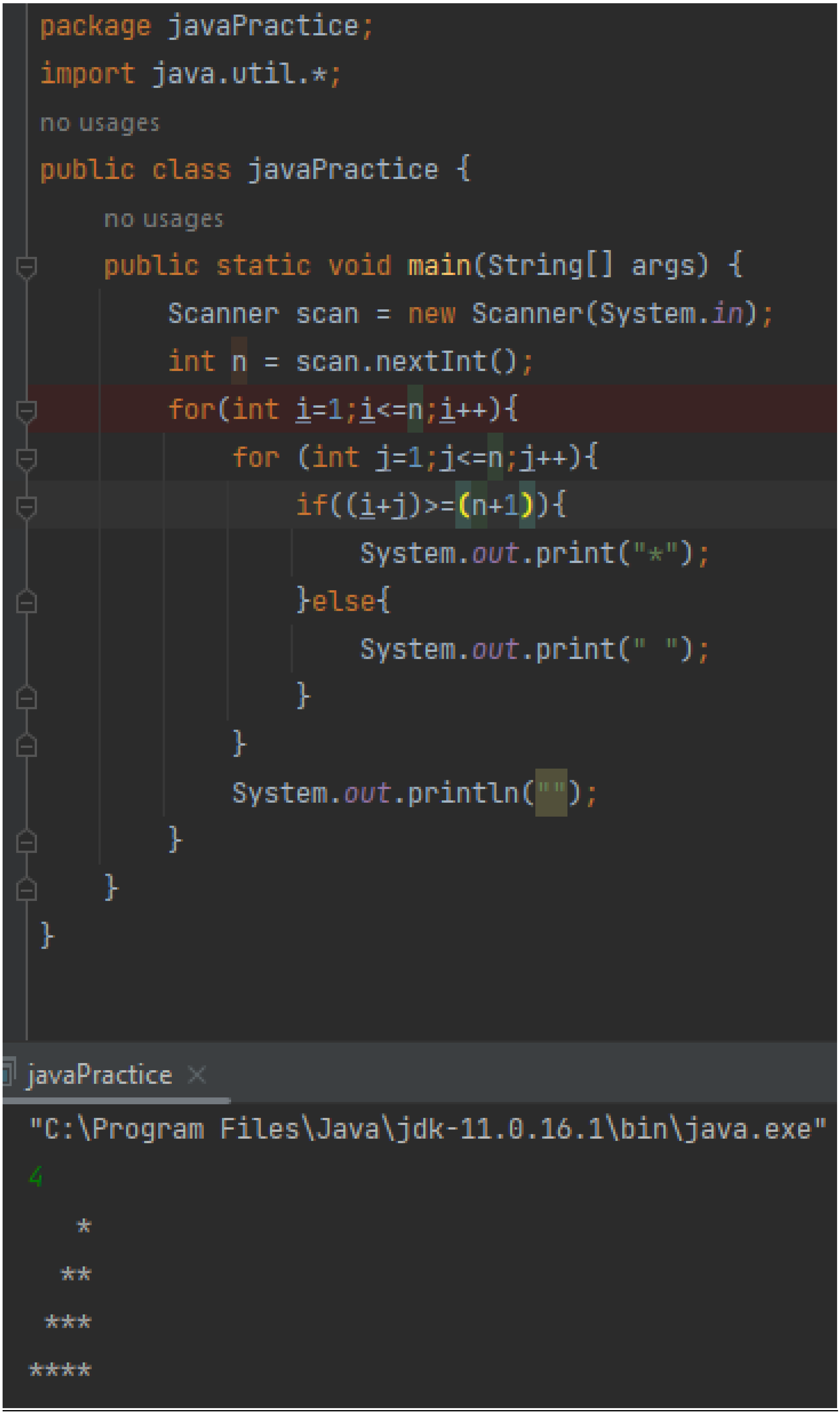
javaPractice

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.ex
4
   *
  **
 ***
****
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for(int i=1;i<=n;i++){
            for (int j=i;j<=n-i;j++){
                System.out.print(" ");
            }
            for (int j=1;j<=i;j++){
                System.out.print("*");
            }
            System.out.println("");
        }
    }
}
```

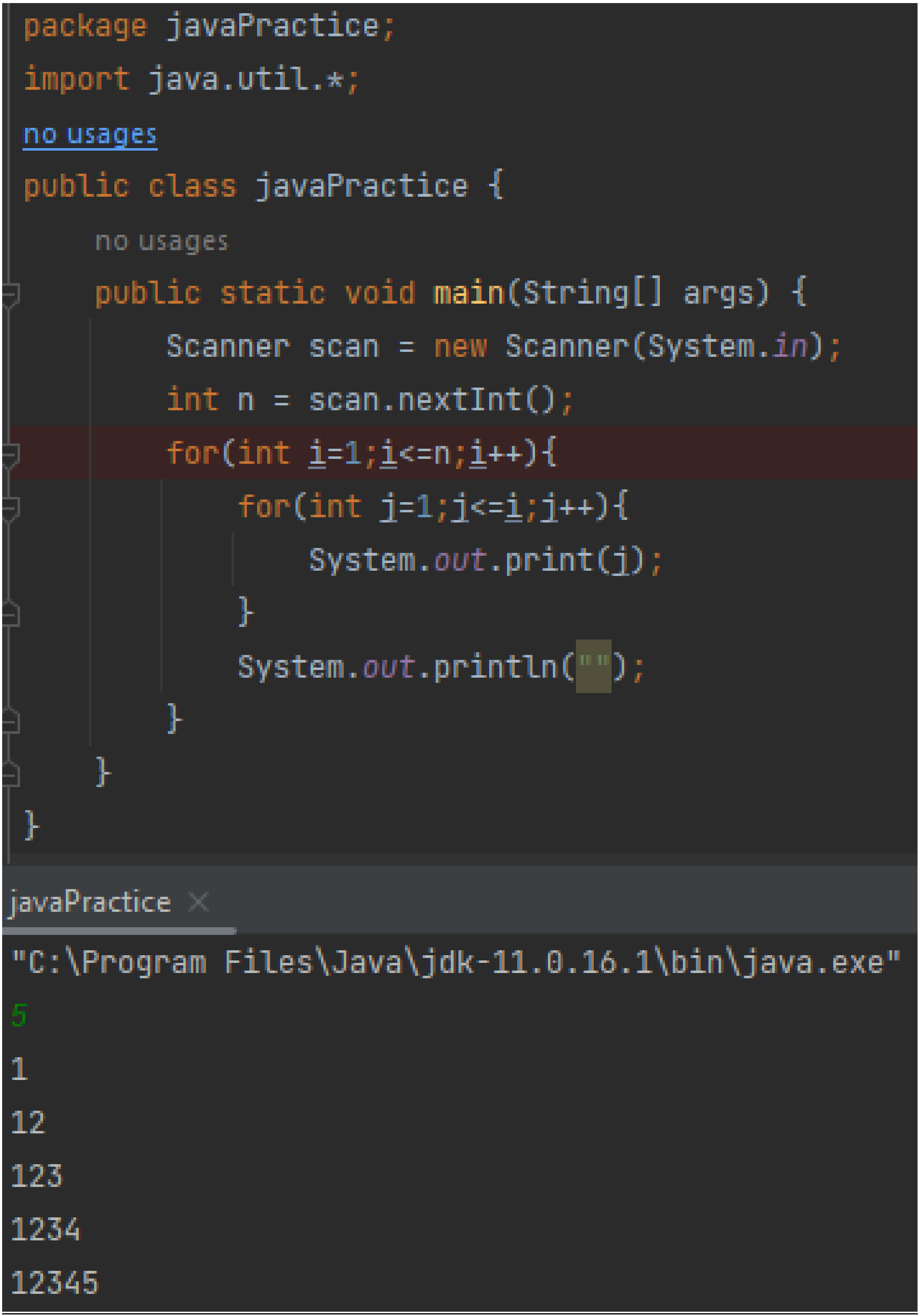
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.ex
4
   *
  **
 ***
****
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for(int i=1;i<=n;i++){
            for (int j=1;j<=n;j++){
                if((i+j)>=(n+1)){
                    System.out.print("*");
                }else{
                    System.out.print(" ");
                }
            }
            System.out.println("");
        }
    }
}
```

javaPractice

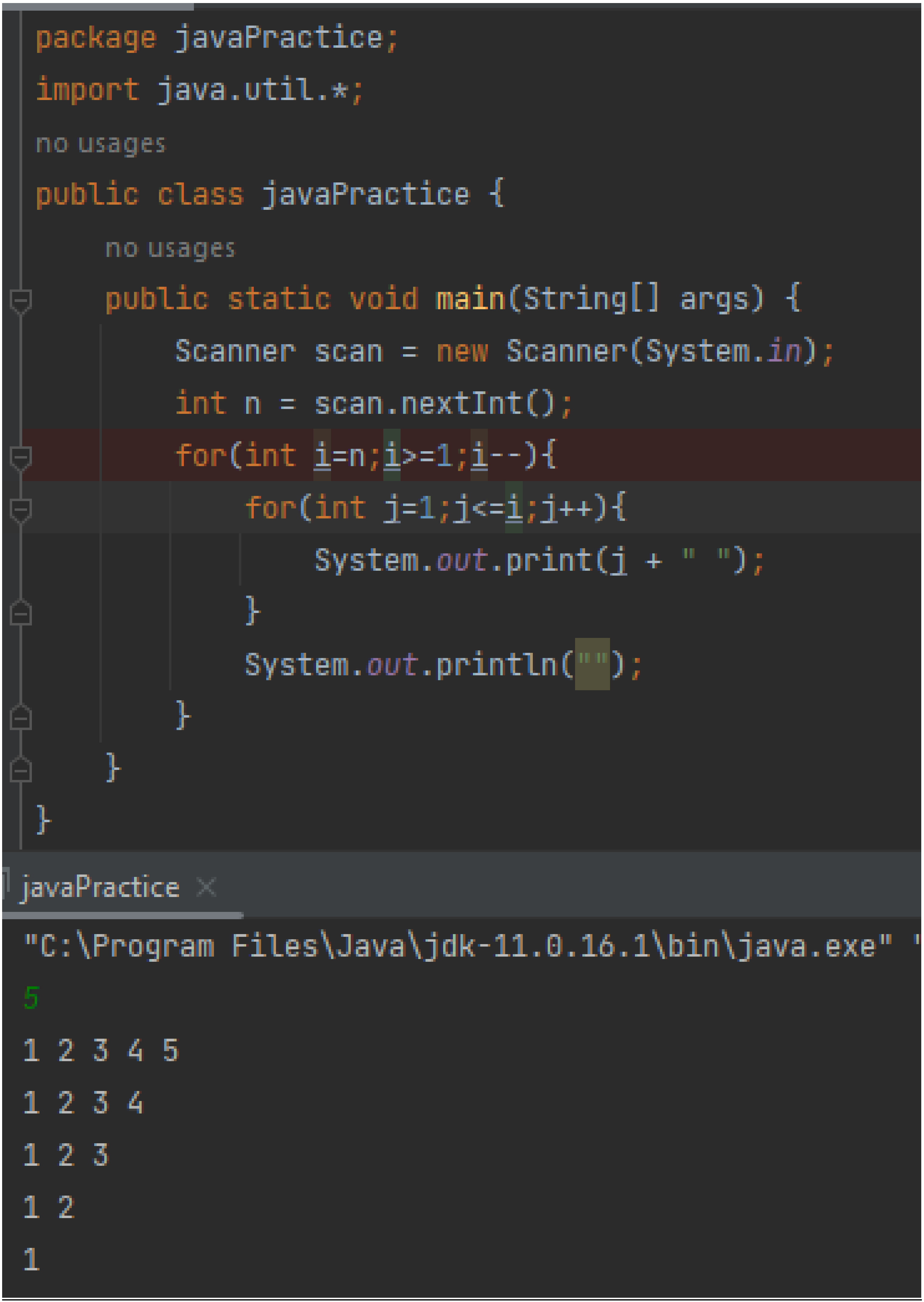
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe"
4
   *
  **
 ***
****
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for(int i=1;i<=n;i++){
            for(int j=1;j<=i;j++){
                System.out.print(j);
            }
            System.out.println("");
        }
    }
}
```

javaPractice

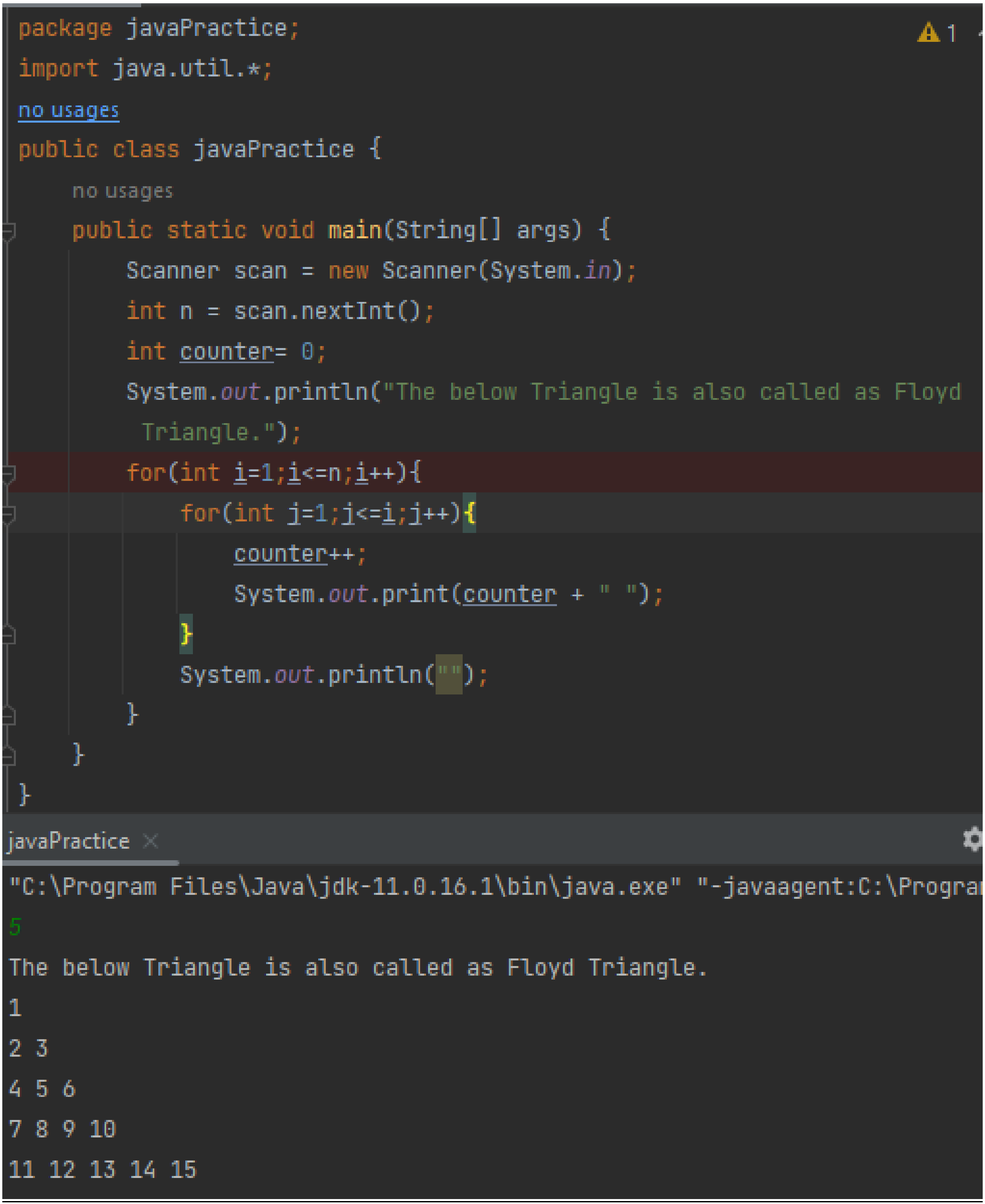
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe"
5
1
12
123
1234
12345
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for(int i=n;i>=1;i--){
            for(int j=1;j<=i;j++){
                System.out.print(j + " ");
            }
            System.out.println("");
        }
    }
}
```

javaPractice

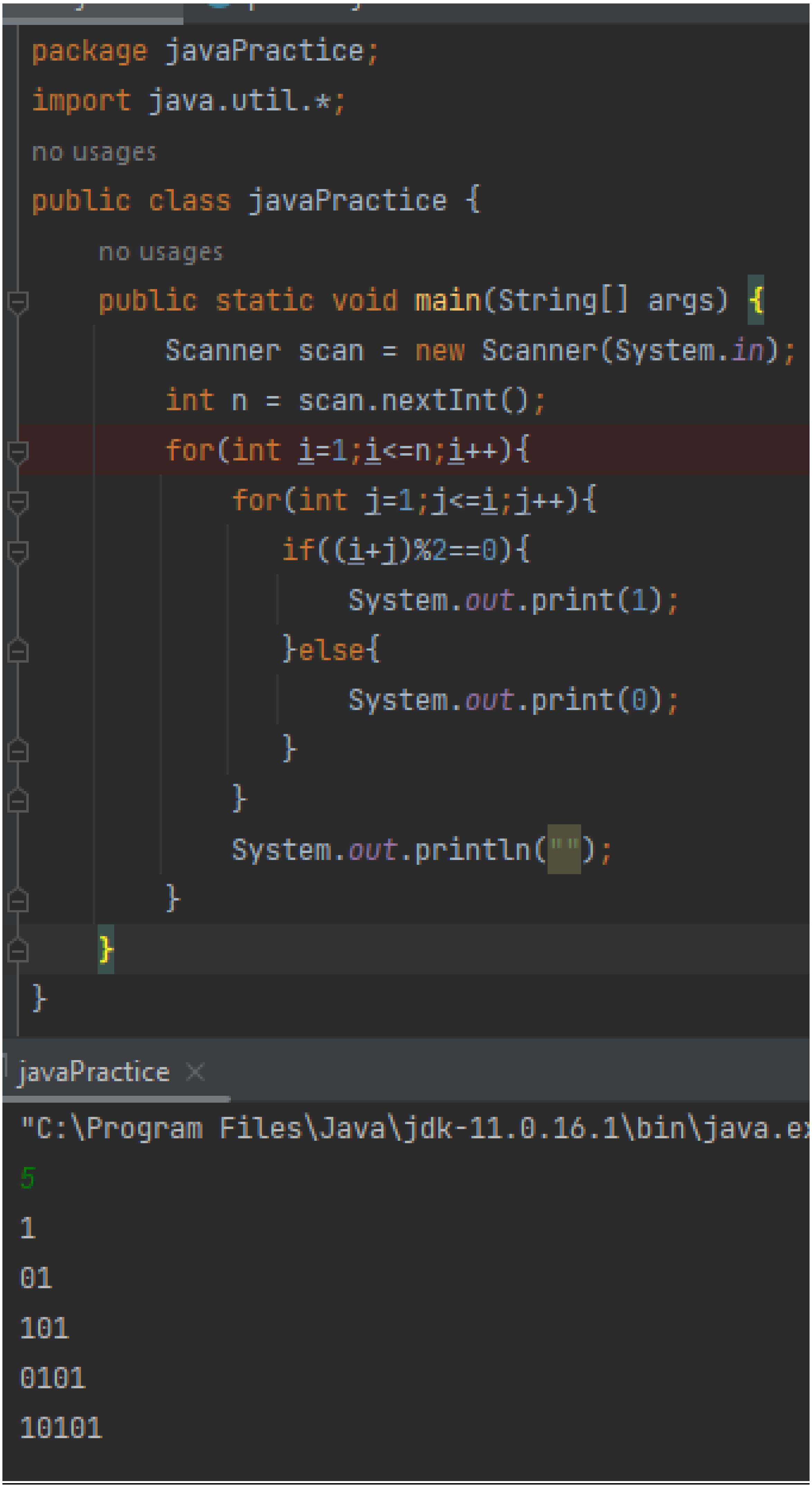
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" '
5
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        int counter= 0;
        System.out.println("The below Triangle is also called as Floyd
         Triangle.");
        for(int i=1;i<=n;i++){
            for(int j=1;j<=i;j++){
                counter++;
                System.out.print(counter + " ");
            }
            System.out.println("");
        }
    }
}
```

javaPractice

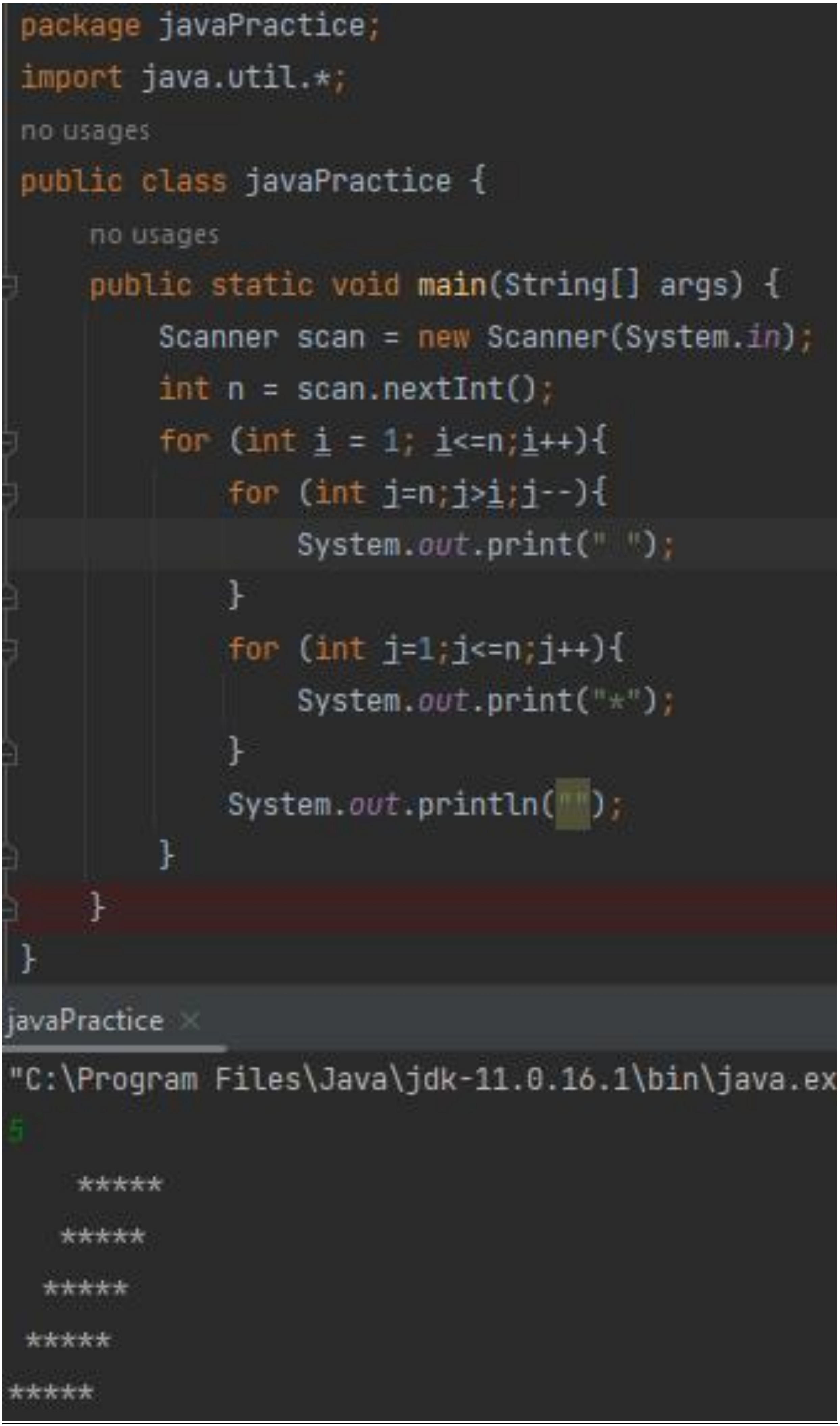
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "-javaagent:C:\Progra
5
The below Triangle is also called as Floyd Triangle.
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```



```java
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for(int i=1;i<=n;i++){
            for(int j=1;j<=i;j++){
               if((i+j)%2==0){
                   System.out.print(1);
               }else{
                   System.out.print(0);
               }
            }
            System.out.println("");
        }
    }
}
```

javaPractice

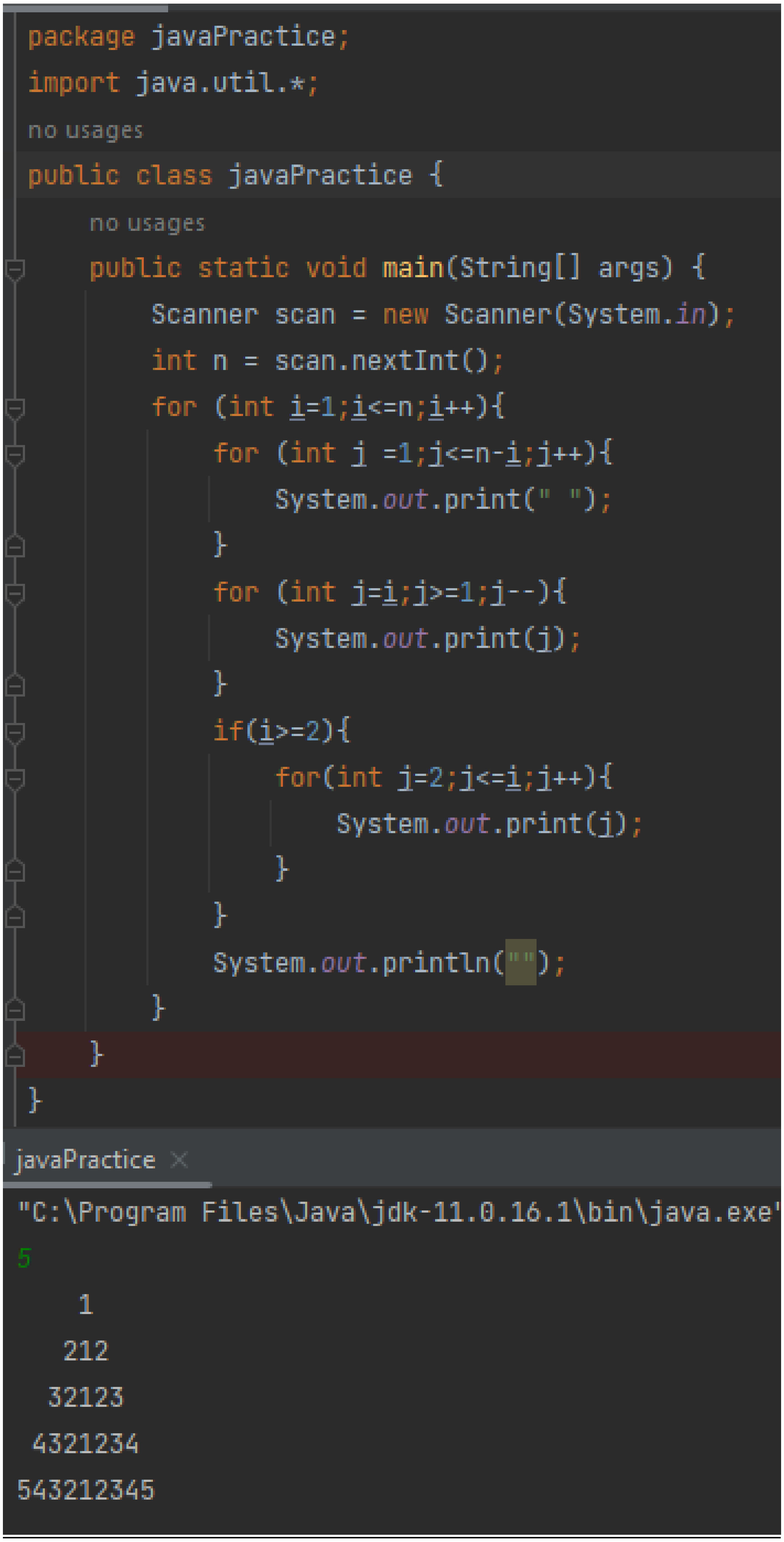
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.ex
5
1
01
101
0101
10101
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for (int i = 1; i<=n;i++){
            for (int j=n;j>i;j--){
                System.out.print(" ");
            }
            for (int j=1;j<=n;j++){
                System.out.print("*");
            }
            System.out.println("");
        }
    }
}
```

javaPractice

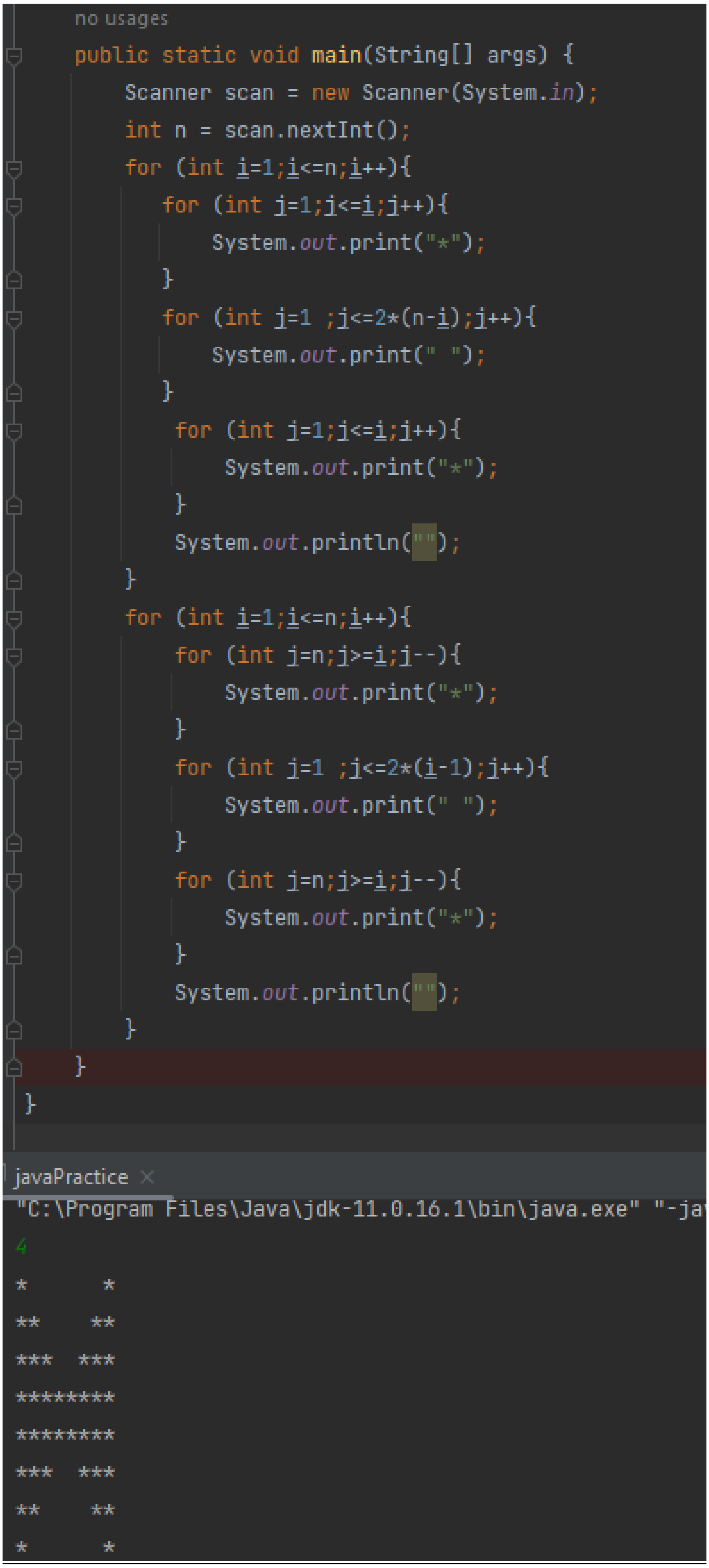
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.ex
5
    *****
   *****
  *****
 *****
*****
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for (int i=1;i<=n;i++){
            for (int j =1;j<=n-i;j++){
                System.out.print(" ");
            }
            for (int j=i;j>=1;j--){
                System.out.print(j);
            }
            if(i>=2){
                for(int j=2;j<=i;j++){
                    System.out.print(j);
                }
            }
            System.out.println("");
        }
    }
}
```

javaPractice ×

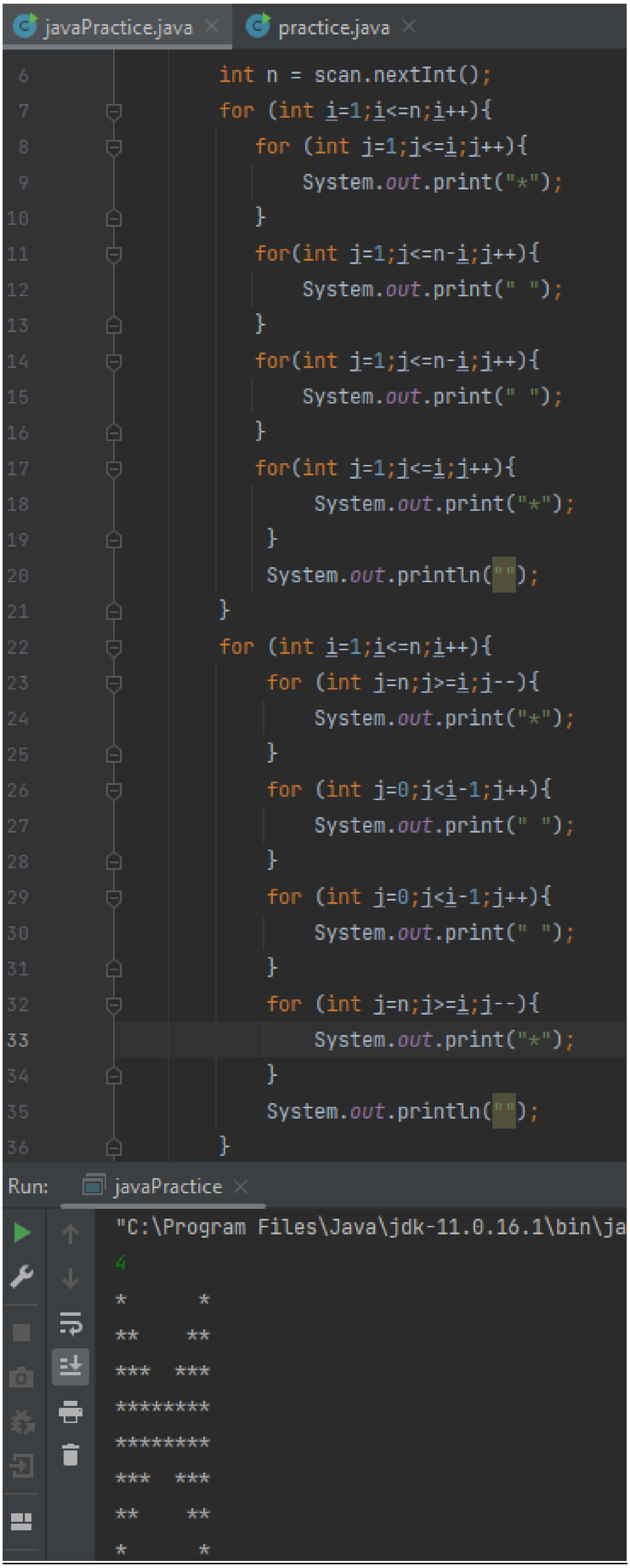
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe'
5
    1
   212
  32123
 4321234
543212345
```



```
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for (int i=1;i<=n;i++){
            for (int j=1;j<=i;j++){
                System.out.print("*");
            }
            for (int j=1 ;j<=2*(n-i);j++){
                System.out.print(" ");
            }
             for (int j=1;j<=i;j++){
                 System.out.print("*");
             }
             System.out.println("");
        }
        for (int i=1;i<=n;i++){
             for (int j=n;j>=i;j--){
                 System.out.print("*");
             }
             for (int j=1 ;j<=2*(i-1);j++){
                 System.out.print(" ");
             }
             for (int j=n;j>=i;j--){
                 System.out.print("*");
             }
             System.out.println("");
        }
    }
}
```

javaPractice

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "-ja
4
*      *
**    **
***  ***
********
********
***  ***
**    **
*      *
```

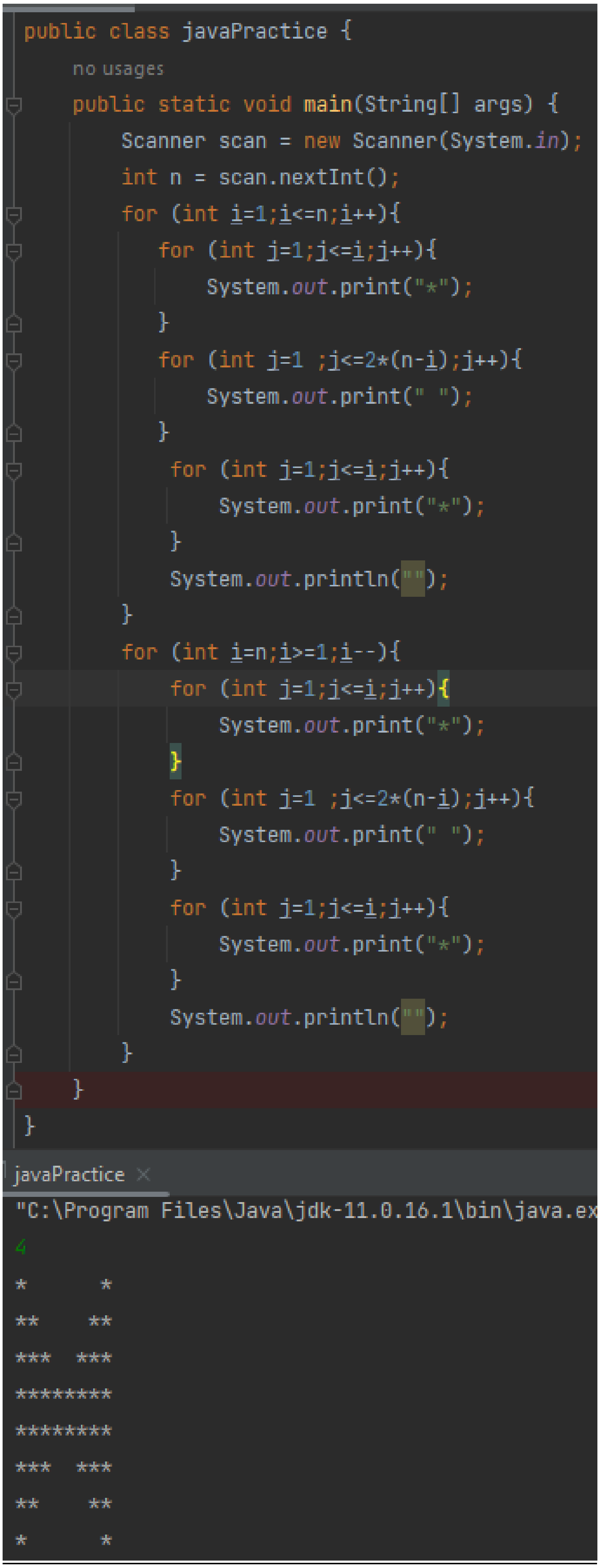


javaPractice.java   practice.java

```java
        int n = scan.nextInt();
        for (int i=1;i<=n;i++){
            for (int j=1;j<=i;j++){
                System.out.print("*");
            }
            for(int j=1;j<=n-i;j++){
                System.out.print(" ");
            }
            for(int j=1;j<=n-i;j++){
                System.out.print(" ");
            }
            for(int j=1;j<=i;j++){
                System.out.print("*");
             }
            System.out.println("");
        }
        for (int i=1;i<=n;i++){
             for (int j=n;j>=i;j--){
                 System.out.print("*");
             }
             for (int j=0;j<i-1;j++){
                 System.out.print(" ");
             }
             for (int j=0;j<i-1;j++){
                 System.out.print(" ");
             }
             for (int j=n;j>=i;j--){
                 System.out.print("*");
             }
            System.out.println("");
        }
```

Run: javaPractice

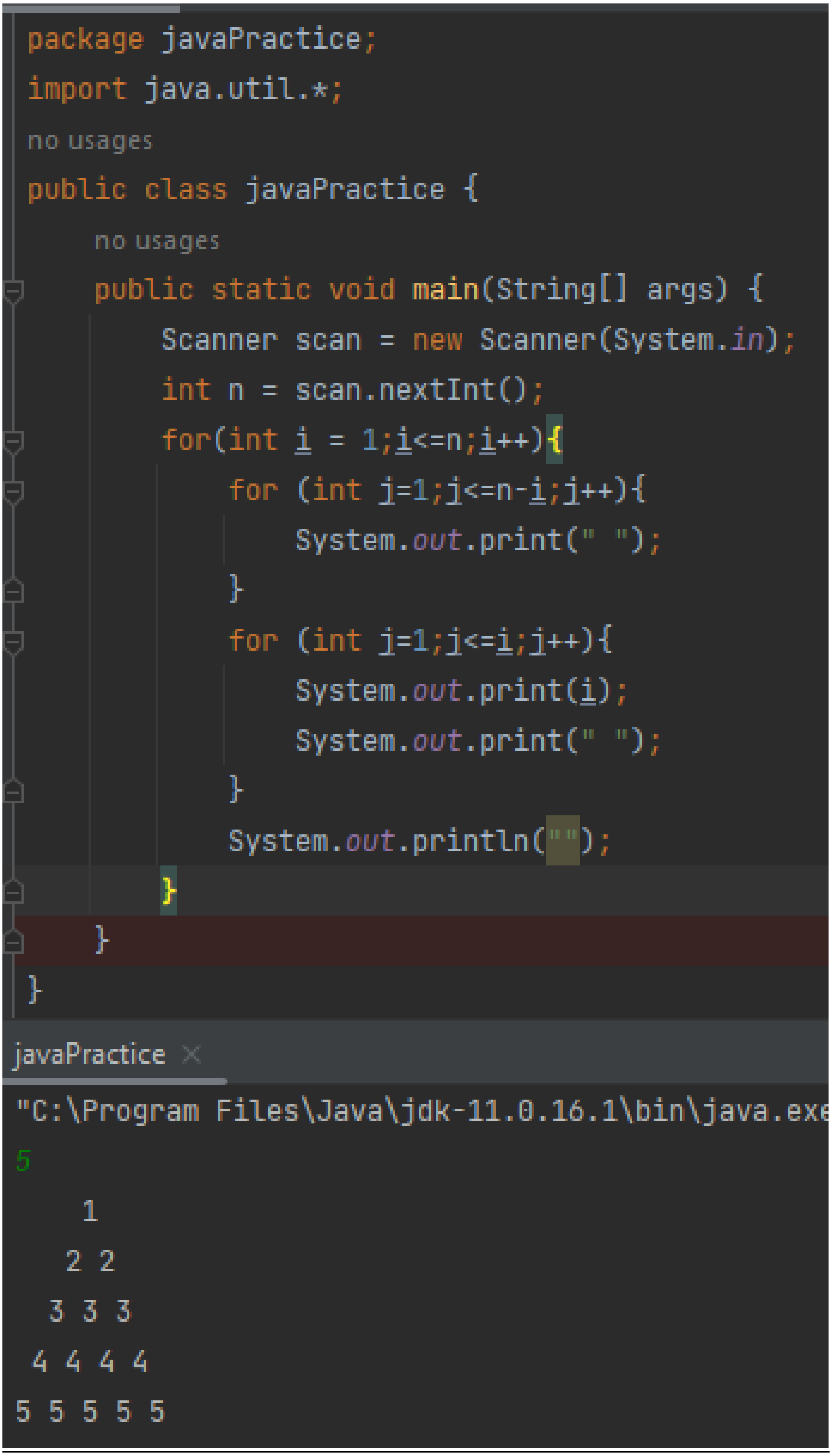
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\ja
4
*      *
**    **
***  ***
********
********
***  ***
**    **
*      *
```



```
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for (int i=1;i<=n;i++){
            for (int j=1;j<=i;j++){
                System.out.print("*");
            }
            for (int j=1 ;j<=2*(n-i);j++){
                System.out.print(" ");
            }
             for (int j=1;j<=i;j++){
                 System.out.print("*");
             }
             System.out.println("");
        }
        for (int i=n;i>=1;i--){
             for (int j=1;j<=i;j++){
                 System.out.print("*");
             }
             for (int j=1 ;j<=2*(n-i);j++){
                 System.out.print(" ");
             }
             for (int j=1;j<=i;j++){
                 System.out.print("*");
             }
             System.out.println("");
        }
    }
}
```

javaPractice

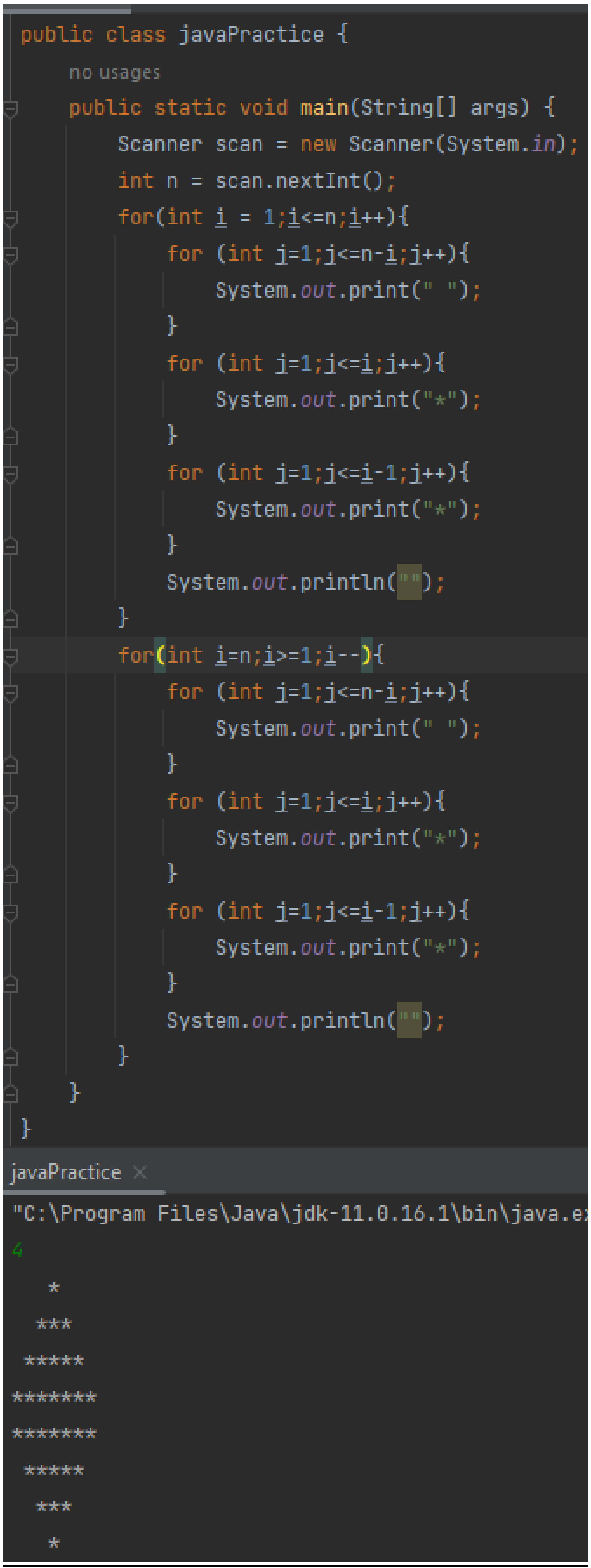
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.ex
4
*      *
**    **
***  ***
********
********
***  ***
**    **
*      *
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for(int i = 1;i<=n;i++){
            for (int j=1;j<=n-i;j++){
                System.out.print(" ");
            }
            for (int j=1;j<=i;j++){
                System.out.print(i);
                System.out.print(" ");
            }
            System.out.println("");
        }
    }
}
```

javaPractice

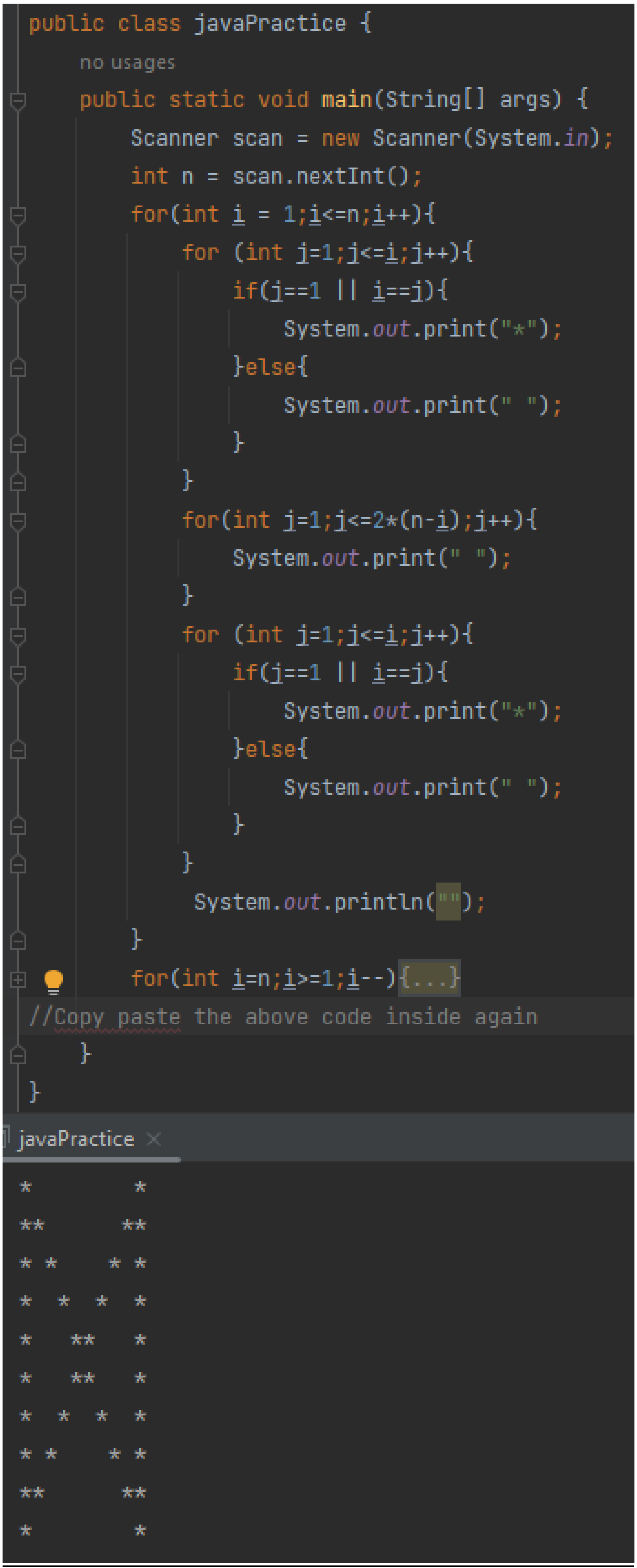
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
5
    1
   2 2
  3 3 3
 4 4 4 4
5 5 5 5 5
```



```
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for(int i = 1;i<=n;i++){
            for (int j=1;j<=n-i;j++){
                System.out.print(" ");
            }
            for (int j=1;j<=i;j++){
                System.out.print("*");
            }
            for (int j=1;j<=i-1;j++){
                System.out.print("*");
            }
            System.out.println("");
        }
        for(int i=n;i>=1;i--){
            for (int j=1;j<=n-i;j++){
                System.out.print(" ");
            }
            for (int j=1;j<=i;j++){
                System.out.print("*");
            }
            for (int j=1;j<=i-1;j++){
                System.out.print("*");
            }
            System.out.println("");
        }
    }
}
```

javaPractice ×

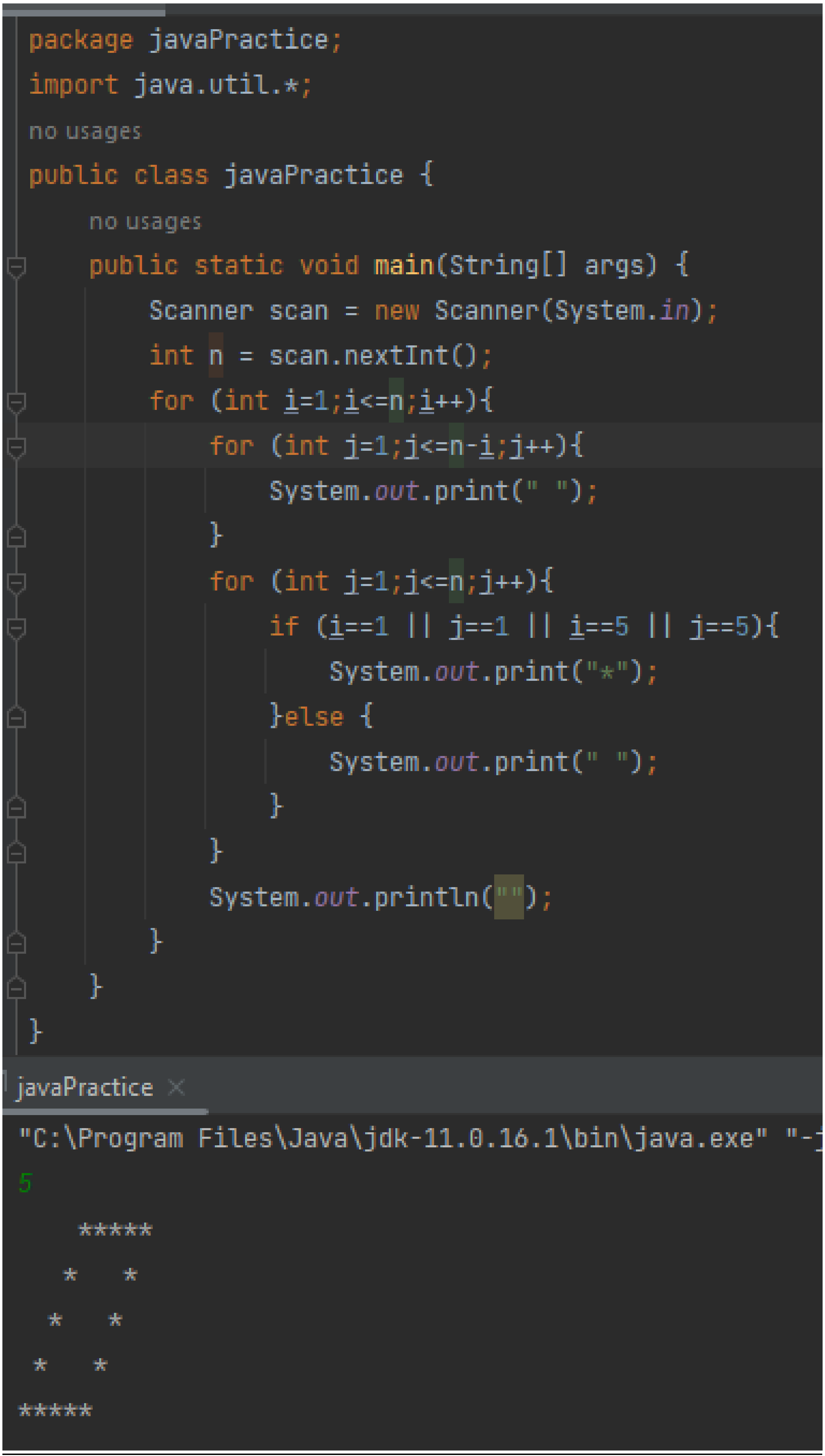
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.ex
4
   *
  ***
 *****
*******
*******
 *****
  ***
   *
```



```java
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for(int i = 1;i<=n;i++){
            for (int j=1;j<=i;j++){
                if(j==1 || i==j){
                    System.out.print("*");
                }else{
                    System.out.print(" ");
                }
            }
            for(int j=1;j<=2*(n-i);j++){
                System.out.print(" ");
            }
            for (int j=1;j<=i;j++){
                if(j==1 || i==j){
                    System.out.print("*");
                }else{
                    System.out.print(" ");
                }
            }
             System.out.println("");
        }
        for(int i=n;i>=1;i--){...}
//Copy paste the above code inside again
    }
}
```

javaPractice

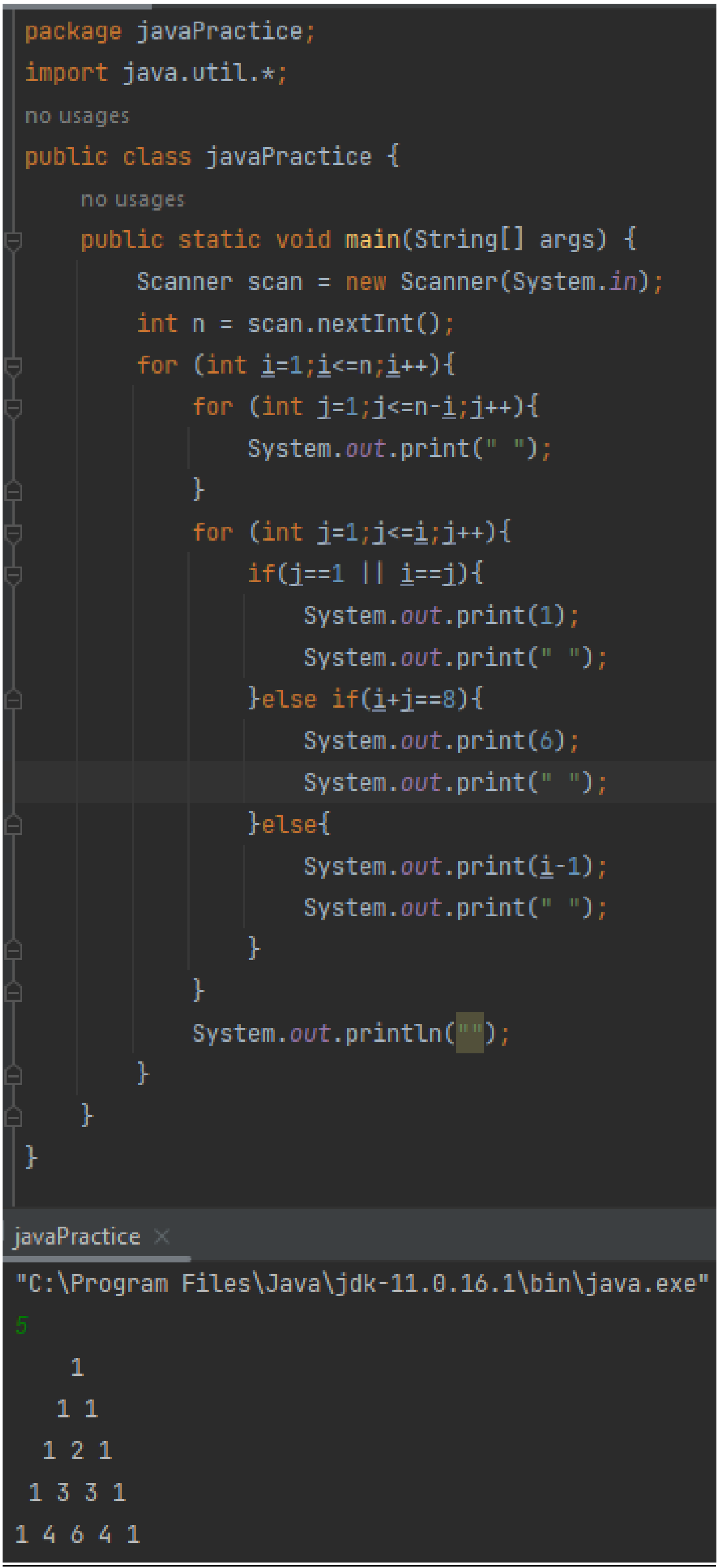
```
*        *
**      **
* *    * *
*  *  *  *
*   **   *
*   **   *
*  *  *  *
* *    * *
**      **
*        *
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for (int i=1;i<=n;i++){
            for (int j=1;j<=n-i;j++){
                System.out.print(" ");
            }
            for (int j=1;j<=n;j++){
                if (i==1 || j==1 || i==5 || j==5){
                    System.out.print("*");
                }else {
                    System.out.print(" ");
                }
            }
            System.out.println("");
        }
    }
}
```

javaPractice

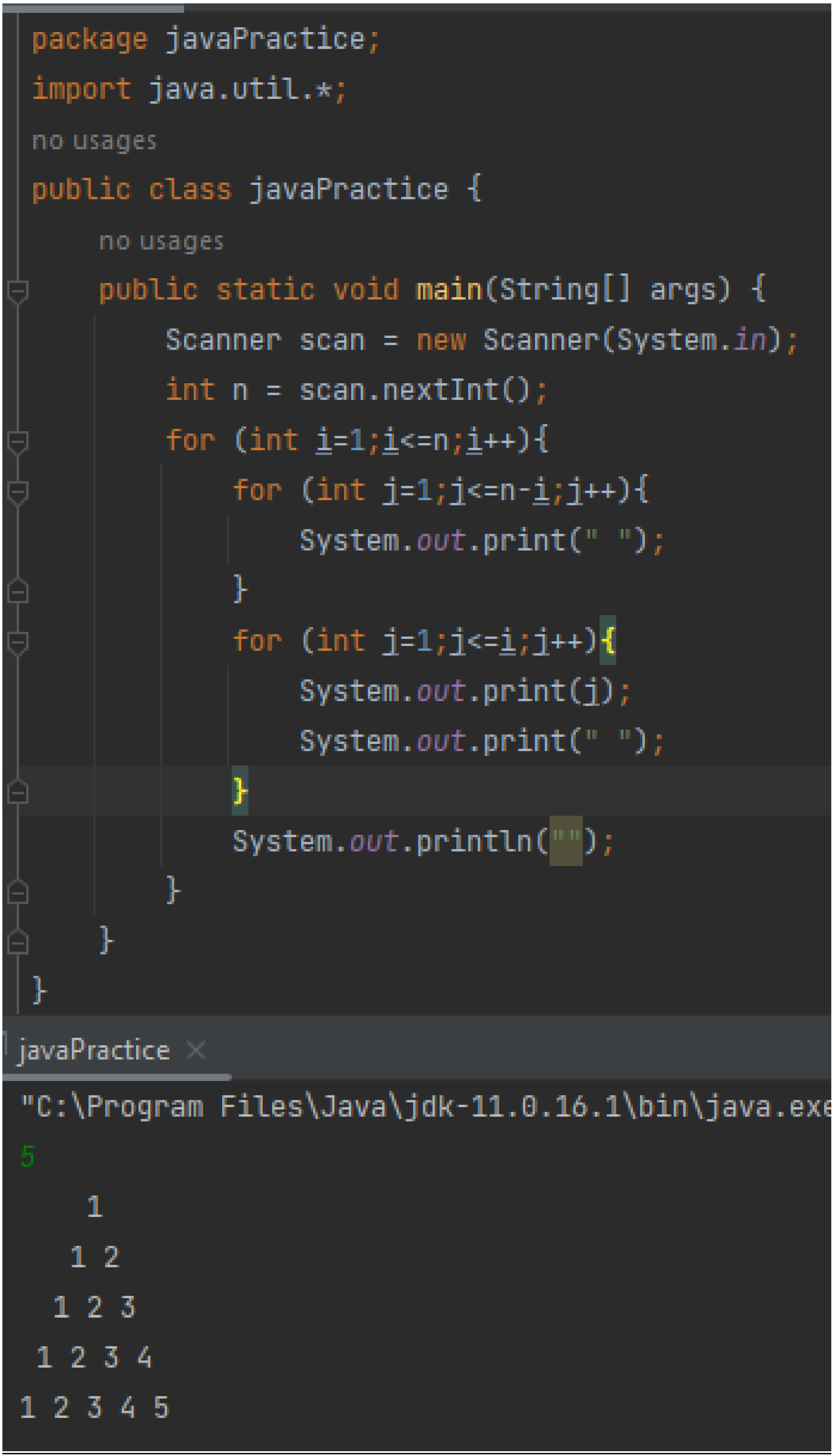
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "-j
5
    *****
   *   *
  *   *
 *   *
*****
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for (int i=1;i<=n;i++){
            for (int j=1;j<=n-i;j++){
                System.out.print(" ");
            }
            for (int j=1;j<=i;j++){
                if(j==1 || i==j){
                    System.out.print(1);
                    System.out.print(" ");
                }else if(i+j==8){
                    System.out.print(6);
                    System.out.print(" ");
                }else{
                    System.out.print(i-1);
                    System.out.print(" ");
                }
            }
            System.out.println("");
        }
    }
}
```

javaPractice

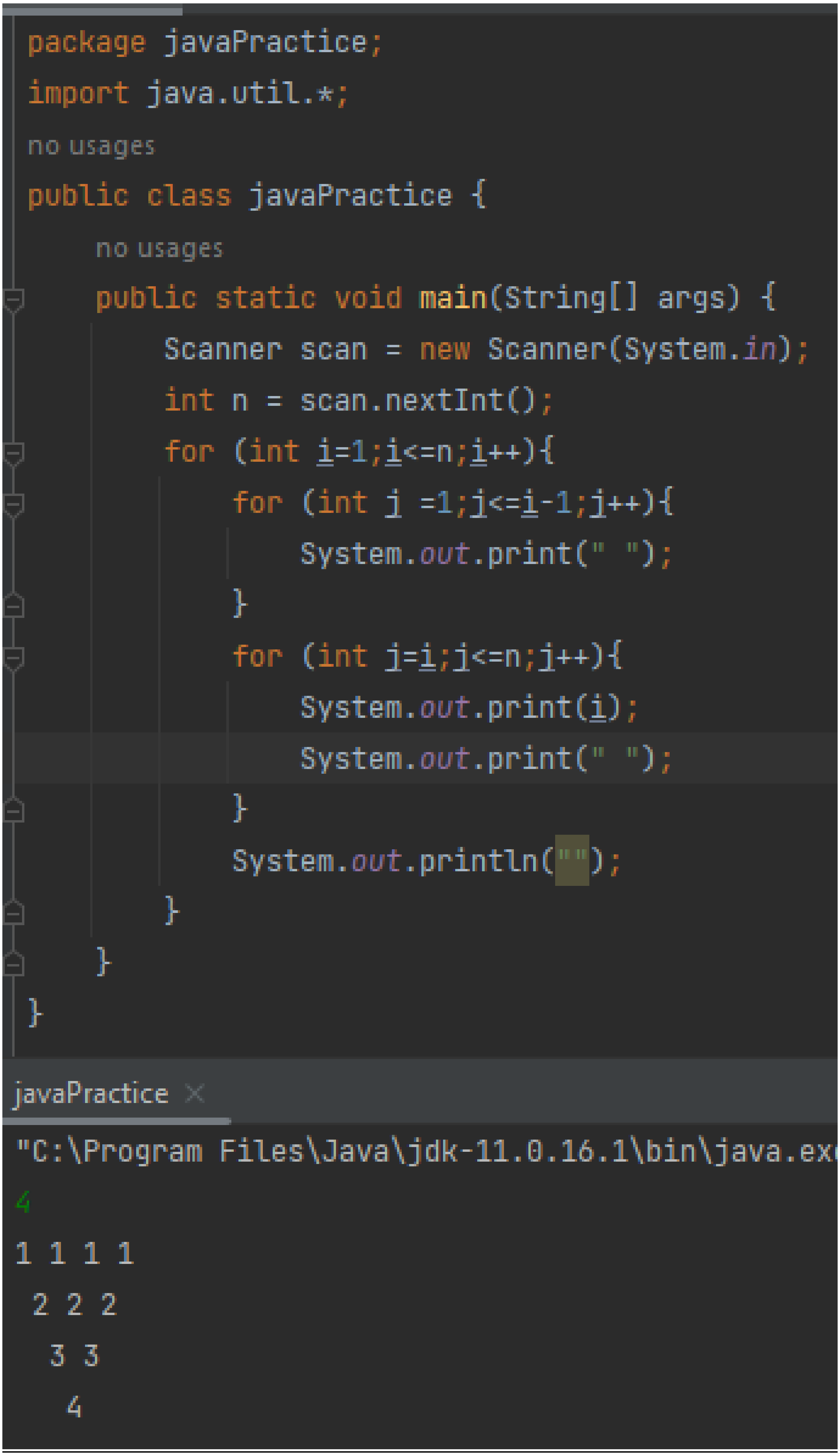
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe"
5
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```



```java
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for (int i=1;i<=n;i++){
            for (int j=1;j<=n-i;j++){
                System.out.print(" ");
            }
            for (int j=1;j<=i;j++){
                System.out.print(j);
                System.out.print(" ");
            }
            System.out.println("");
        }
    }
}
```

javaPractice

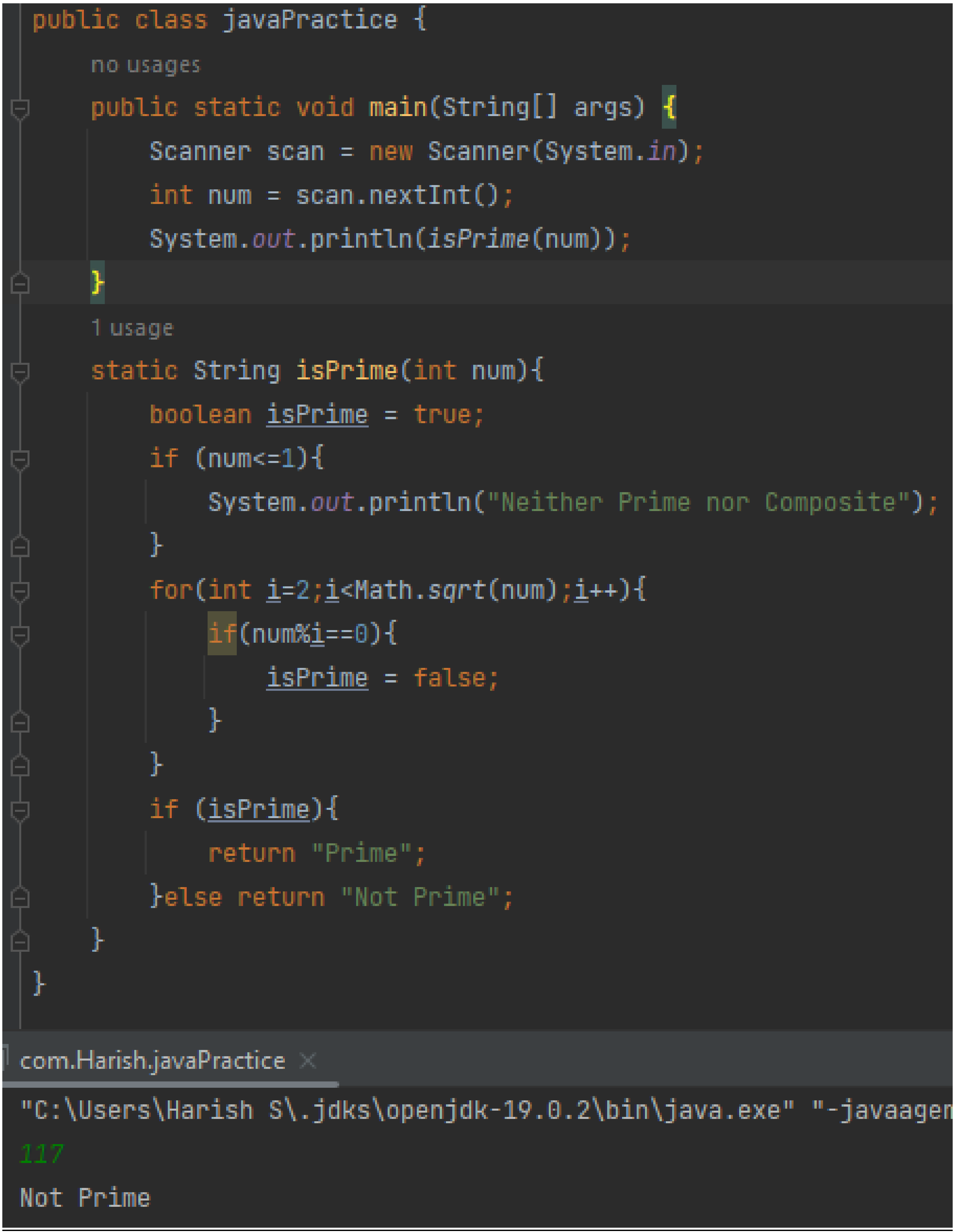
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
5
    1
   1 2
  1 2 3
 1 2 3 4
1 2 3 4 5
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        for (int i=1;i<=n;i++){
            for (int j =1;j<=i-1;j++){
                System.out.print(" ");
            }
            for (int j=i;j<=n;j++){
                System.out.print(i);
                System.out.print(" ");
            }
            System.out.println("");
        }
    }
}
```

javaPractice

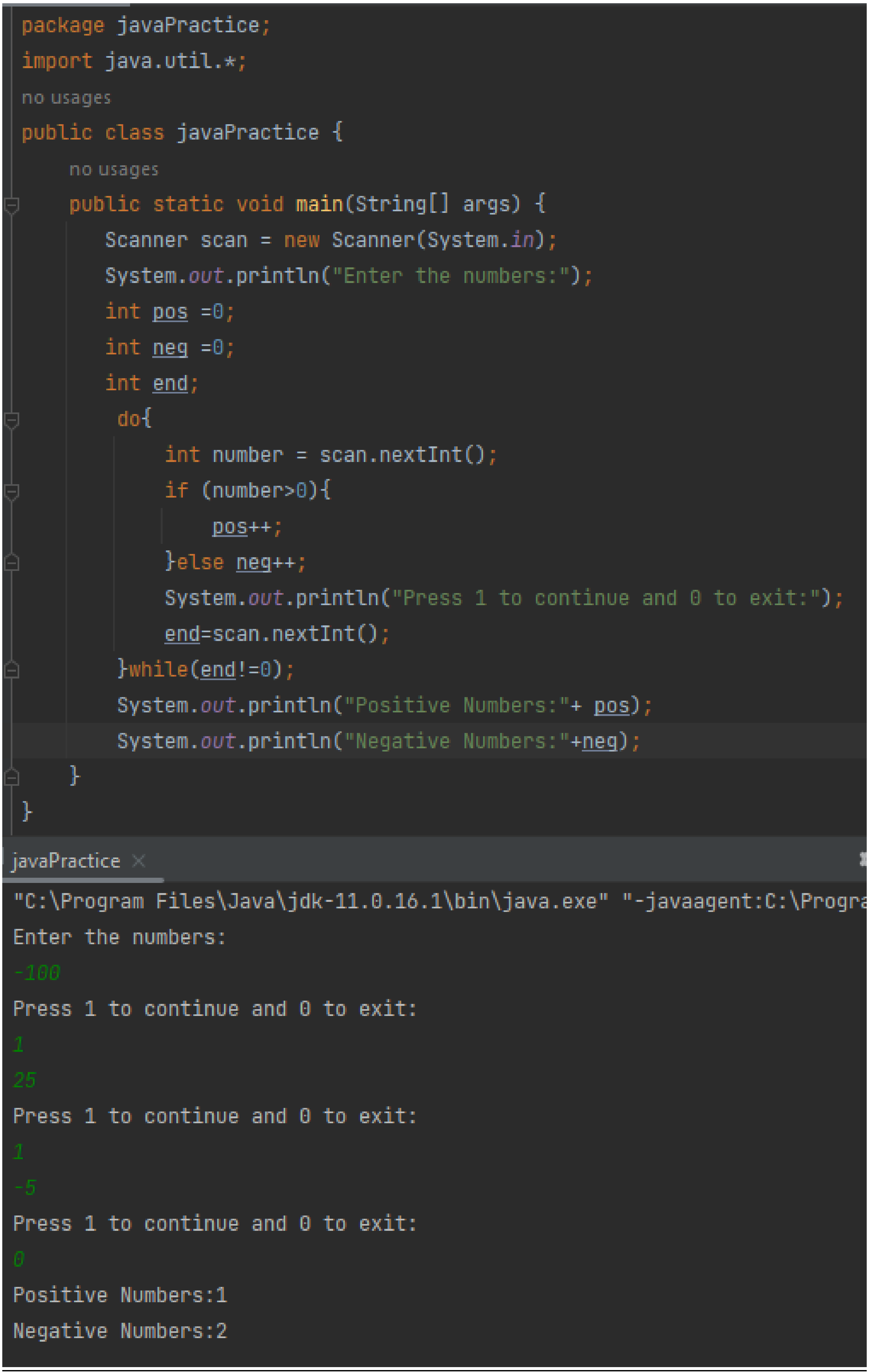
```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
4
1 1 1 1
 2 2 2
  3 3
   4
```



```java
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
        System.out.println(isPrime(num));
    }
    1 usage
    static String isPrime(int num){
        boolean isPrime = true;
        if (num<=1){
            System.out.println("Neither Prime nor Composite");
        }
        for(int i=2;i<Math.sqrt(num);i++){
            if(num%i==0){
                isPrime = false;
            }
        }
        if (isPrime){
            return "Prime";
        }else return "Not Prime";
    }
}
```

com.Harish.javaPractice

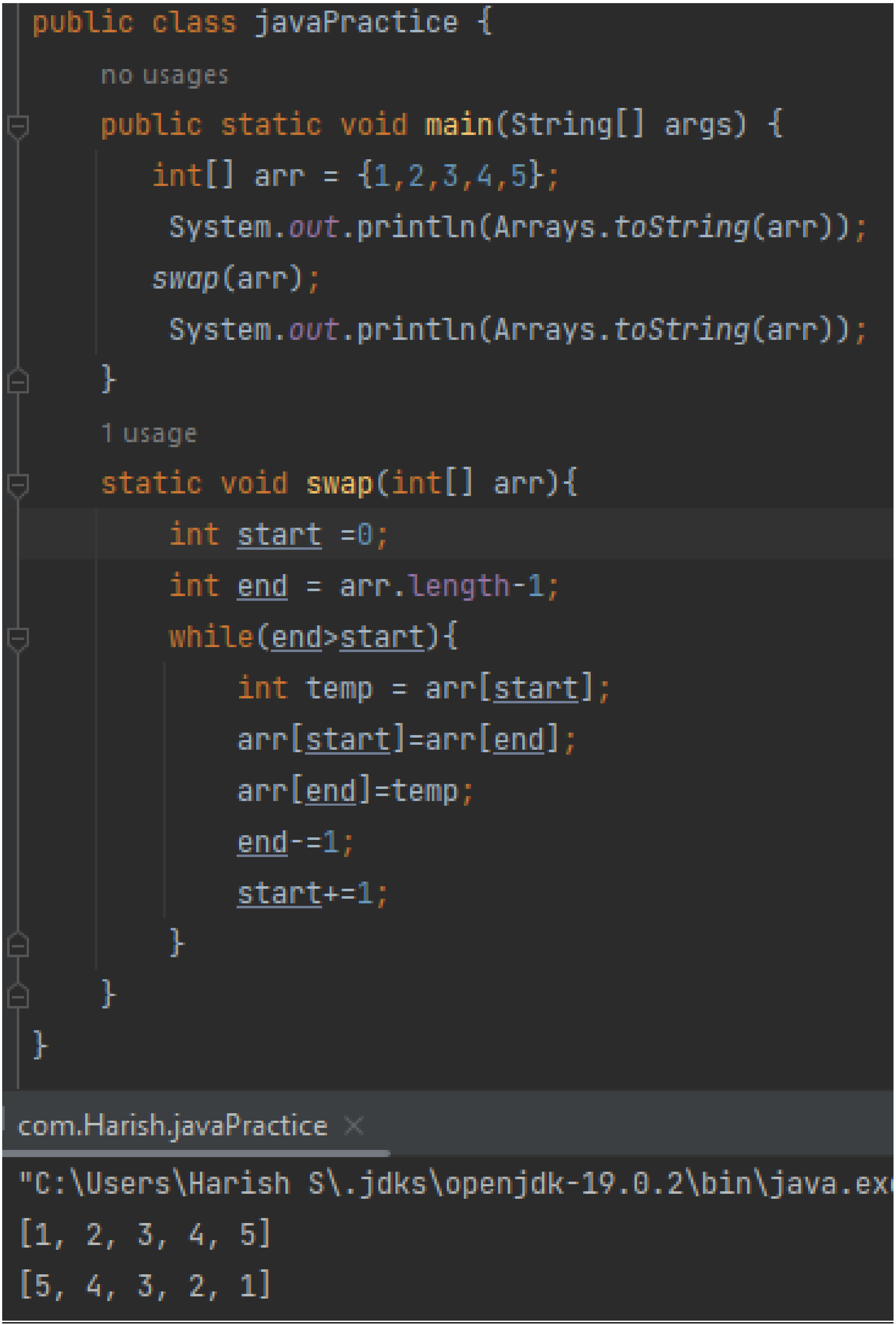
```
"C:\Users\Harish S\.jdks\openjdk-19.0.2\bin\java.exe" "-javaagen
117
Not Prime
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter the numbers:");
        int pos =0;
        int neg =0;
        int end;
         do{
             int number = scan.nextInt();
             if (number>0){
                 pos++;
             }else neg++;
             System.out.println("Press 1 to continue and 0 to exit:");
             end=scan.nextInt();
         }while(end!=0);
         System.out.println("Positive Numbers:"+ pos);
         System.out.println("Negative Numbers:"+neg);
    }
}
```

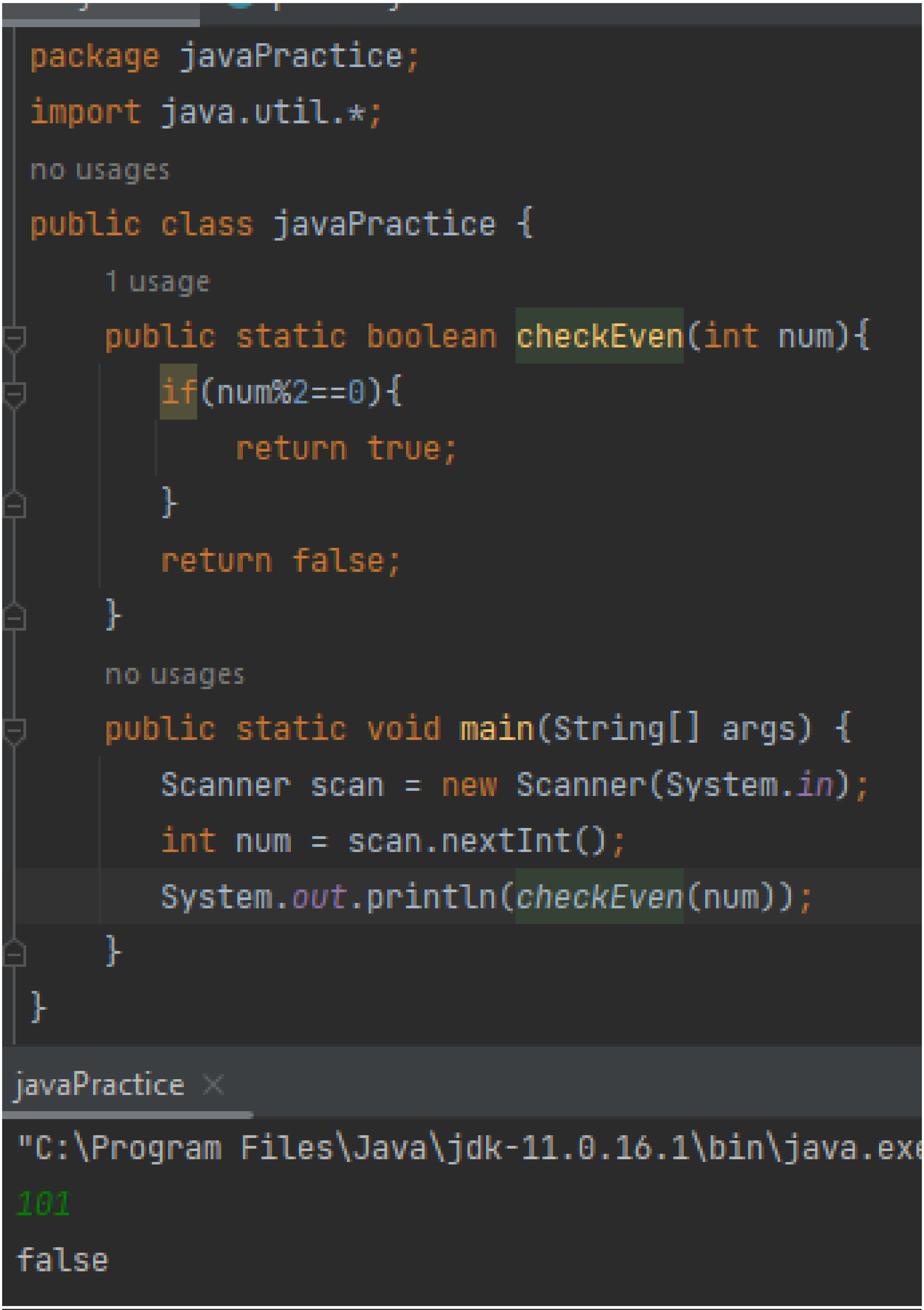
javaPractice ×

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe" "-javaagent:C:\Progra
Enter the numbers:
-100
Press 1 to continue and 0 to exit:
1
25
Press 1 to continue and 0 to exit:
1
-5
Press 1 to continue and 0 to exit:
0
Positive Numbers:1
Negative Numbers:2
```
]



```
public class javaPractice {
    no usages
    public static void main(String[] args) {
        int[] arr = {1,2,3,4,5};
         System.out.println(Arrays.toString(arr));
        swap(arr);
         System.out.println(Arrays.toString(arr));
    }
    1 usage
    static void swap(int[] arr){
         int start =0;
         int end = arr.length-1;
         while(end>start){
             int temp = arr[start];
             arr[start]=arr[end];
             arr[end]=temp;
             end-=1;
             start+=1;
         }
    }
}
```

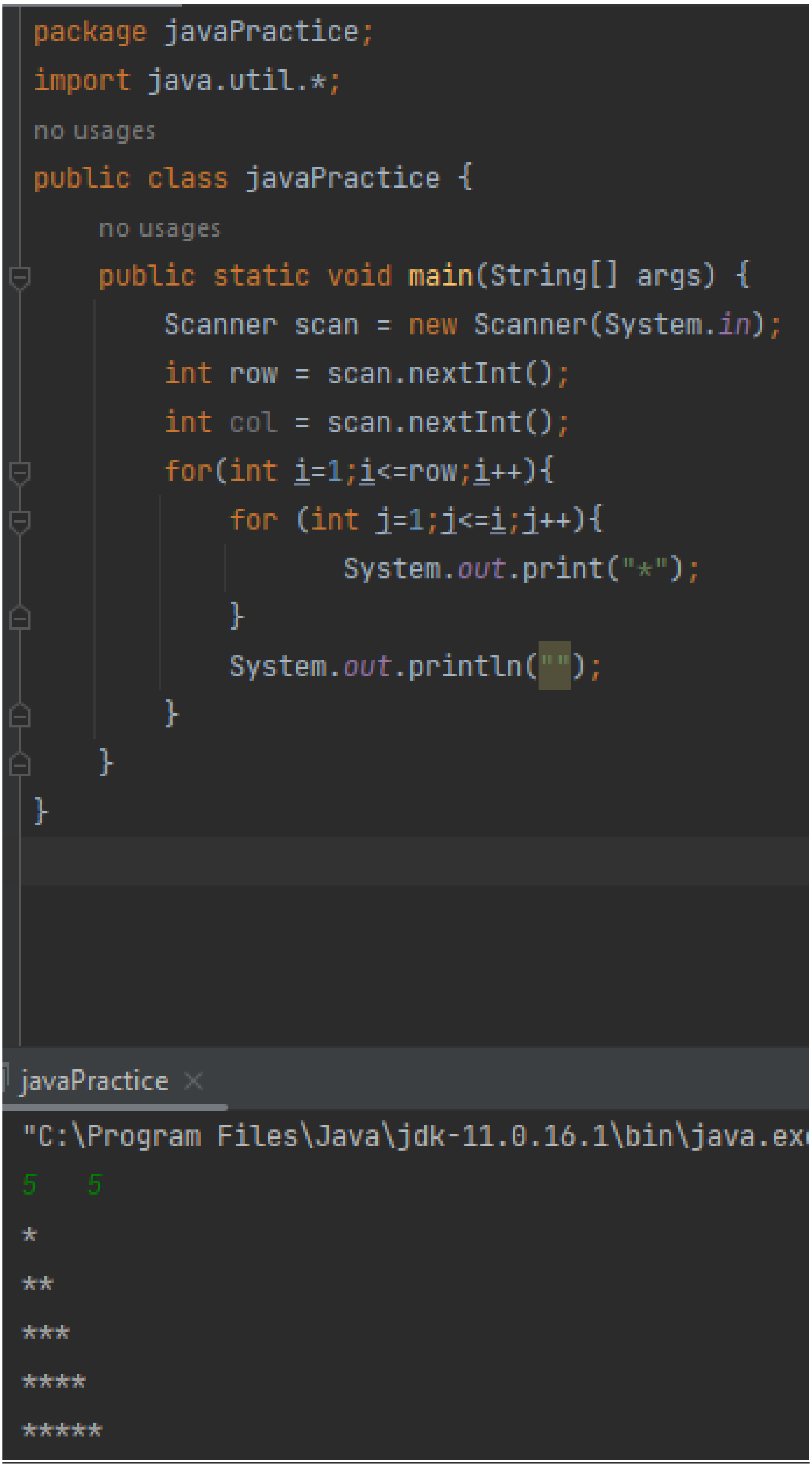
```
"C:\Users\Harish S\.jdks\openjdk-19.0.2\bin\java.exe
[1, 2, 3, 4, 5]
[5, 4, 3, 2, 1]
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    1 usage
    public static boolean checkEven(int num){
        if(num%2==0){
            return true;
        }
        return false;
    }
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int num = scan.nextInt();
        System.out.println(checkEven(num));
    }
}
```

javaPractice

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
101
false
```



```
package javaPractice;
import java.util.*;
no usages
public class javaPractice {
    no usages
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int row = scan.nextInt();
        int col = scan.nextInt();
        for(int i=1;i<=row;i++){
            for (int j=1;j<=i;j++){
                    System.out.print("*");
            }
            System.out.println("");
        }
    }
}
```

```
"C:\Program Files\Java\jdk-11.0.16.1\bin\java.exe
5   5
*
**
***
****
*****
```