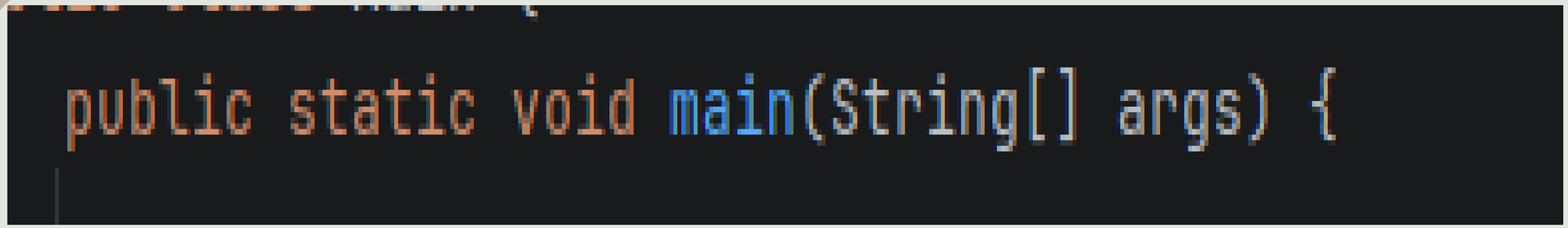
Java course

Basics

Package : is used to group classes to organize project.

Class : is a template used to create objects and it is where you define methods and attributes.

Main method : is the starting point of a Java program where execution begins, each program should have one main method .



Public : to be accessed everywhere.
Static :to run without creating an object of the class.
Void : means it does not return value.
Main : it’s name.
String[] : array of strings.
args : the name of the array (can be any valid name)
String[] args : is a parameter of the main method that is used to receive command-line arguments passed to a Java program when it starts.

Variables : think of it as a container where you store data in memory.
Every variable has a name, a data type, and a value.

```java
public static void main(String[] args) {

    int age ;
    age =10;
```

```java
public class Main {

    public static void main(String[] args) {

        int age = 10;
        System.out.println(age);

        age = 20;
        System.out.println(age);

    }
}
```

```
Run  Main
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:
10
20

Process finished with exit code 0
```

Note : you can initialize variables with the same type in many ways.

```
int age=10;
int size = 20;
int speed =30;
```

```
int age, size , speed;
age =10;
size = 20;
speed = 30;
```

```
int age=10, size =20, speed=30;
```

Primitive Data Types

Type	Size	Description / Example
byte	1 byte	Small integer from -128 to 127
short	2 bytes	Small integer from -32,768 to 32,767
int	4 bytes	Integer from -2,147,483,648 to 2,147,483,647
long	8 bytes	Large integer from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Decimal number (e.g., 3.14f)
double	8 bytes	Decimal number (e.g., 3.14159)
char	2 bytes	Single character (e.g. 'A')
boolean	1 bit	True or false only

Notes :

-Use L for long literals, f for float literals.

-char stores a single character in single quotes.

-boolean only has true or false, nothing else.

```java
int age=10;
byte by =125;
short sh = 130;
long ln = 30000L;
float fl = 30.5f;
double db =3000.54;
System.out.println(age);
System.out.println(by);
System.out.println(sh);
System.out.println(ln);
System.out.println(fl);
System.out.println(db);
```

Main

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ ID
10
125
130
30000
30.5
```

Primitive Data Types

```
//        char grade = "a"; invalid
//        String name = 'Aya'; invalid
//        String grade = 'a'; invalid
//        String grade = "C"; valid
```

```
    boolean condition = true;
    //boolean condition2 = "false"; invalid

    }
}
```

Operators

Arithmetic operators

Operator	Description	Example
+	Adds two operands.	A + B = 30
−	Subtracts second operand from the first.	A − B = -10
*	Multiplies both operands.	A * B = 200
/	Divides numerator by de-numerator.	B / A = 2
%	Modulus Operator and remainder of after an integer division.	B % A = 0
++	Increment operator increases the integer value by one.	A++ = 11
--	Decrement operator decreases the integer value by one.	A-- = 9

Operators
Arithmetic operators

Notes: Integer division removes the decimal part, use float/double to get the decimal part.

```java
public class Main {
    public static void main(String[] args) {

        int a = 10;
        int b = 3;

        System.out.println(a + b); // 13
        System.out.println(a - b); // 7
        System.out.println(a * b); // 30
        System.out.println(a / b); // 3
        System.out.println(a % b); // 1

    }
}
```

```
Main
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program F
13
7
30
3
1
```

Postfix and prefix operators:

postfix increment :variable ++(increment the variable in the next line).
postfix decrement :variable --(decrement the variable in the next line).
prefix increment :++variable (increment the variable in the same line).
prefix decrement : --variable (decrement the variable in the same line).

Postfix and prefix operators:

```java
public class Main {
    public static void main(String[] args) {


        int x = 5;

        int y = ++x;

        System.out.println(x); // 6
        System.out.println(y); // 6


    }
}
```

Main

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-java
6
6
```

```java
public class Main {
    public static void main(String[] args) {

        int x = 5;

        int y = x++;

        System.out.println(x); // 6
        System.out.println(y); // 5


    }
}
```

Main

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program F
6
5
```

Relational operators

Operator	Description	Example
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is true.

Relational operators

```java
public class Main {
    public static void main(String[] args) {

        int a = 10;
        int b = 20;
        System.out.println(a == b);  // false
        System.out.println(a != b);  // true
        System.out.println(a > b);   // false
        System.out.println(a < b);   // true
        System.out.println(a >= b);  // false
        System.out.println(a <= b);  // true
    }
}
```

Run Main

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\J
false
true
false
true
false
true

Process finished with exit code 0
```

Logical operators

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A && B) is true.

Logical operators

```java
public class Main {
    public static void main(String[] args) {

        int age = 20;

        System.out.println(age > 18 || age < 30); // true

    }
    }
```

```java
public class Main {
    public static void main(String[] args) {

        int age = 20;

        System.out.println(age > 18 && age < 30); // true

    }
    }
```

```java
public class Main {
    public static void main(String[] args) {

        int x = 10;
        int y = 5;

        System.out.println(x > y && y > 0); // true
        System.out.println(x < y || y > 0); // true
        System.out.println(!(x == y));      // true

    }
```

```java
public class Main {
    public static void main(String[] args) {

        boolean isStudent = false;

        System.out.println(!isStudent); // true

    }
    }
```

Logical operators

Notes :

-&& stops if the first condition is false

-|| stops if the first condition is true

```java
public class Main {
    public static void main(String[] args) {
        int x = 5;

        System.out.println(x < 10 || x++ > 3);
        System.out.println(x);



    }
    }
```

```
Main
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Prog
true
5

Process finished with exit code 0
```

What happens:

x < 10 → true

Because || needs ONLY ONE condition to be true ,

Java does NOT check x++ > 3

So x++ is never executed

```java
public class Main {
    public static void main(String[] args) {

        int x = 5;

        System.out.println(x > 10 && x++ > 3);
        System.out.println(x);



    }
}
```

```
Run  Main

"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetB
false
5

Process finished with exit code 0
```

What happens:

x > 10 → false

Because && needs BOTH conditions to be true.

Java does NOT check x++ > 3

So x++ is never executed

Assignment Operator

Operator	Meaning	Example	Same as
+=	Add and assign	x += 5	x = x + 5
-=	Subtract and assign	x -= 3	x = x - 3
*=	Multiply and assign	x *= 2	x = x * 2
/=	Divide and assign	x /= 4	x = x / 4
%=	Modulus and assign	x %= 3	x = x % 3

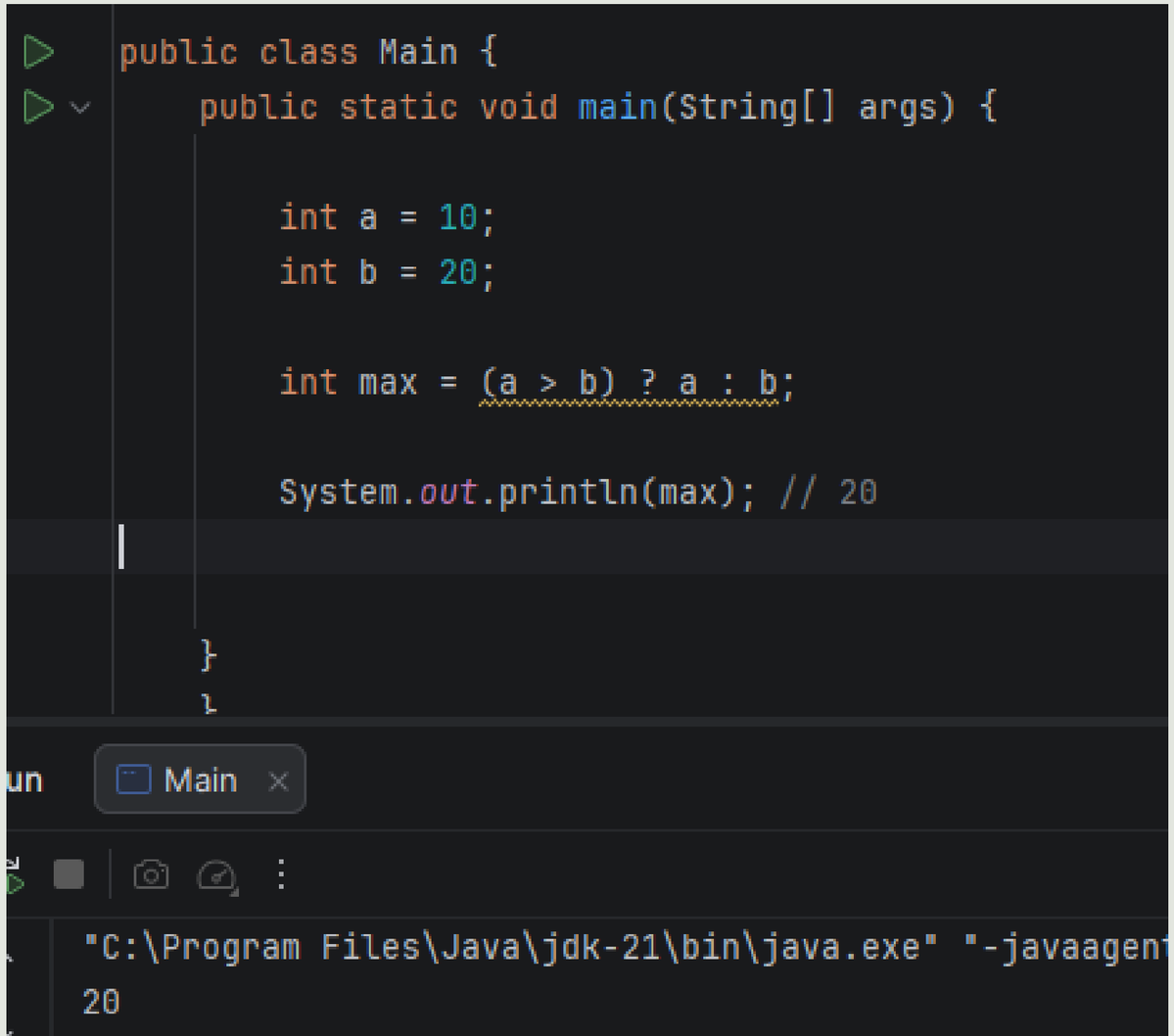
Assignment Operator

```
int x = 10;
x += 5;    // x = 15
x -= 3;    // x = 12
x *= 2;    // x = 24
x /= 4;    // x = 6
x %= 4;    // x = 2
```

Ternary Operator:

it is used to choose one value based on a condition.

condition ? value_if_true : value_if_false;



Control Statements

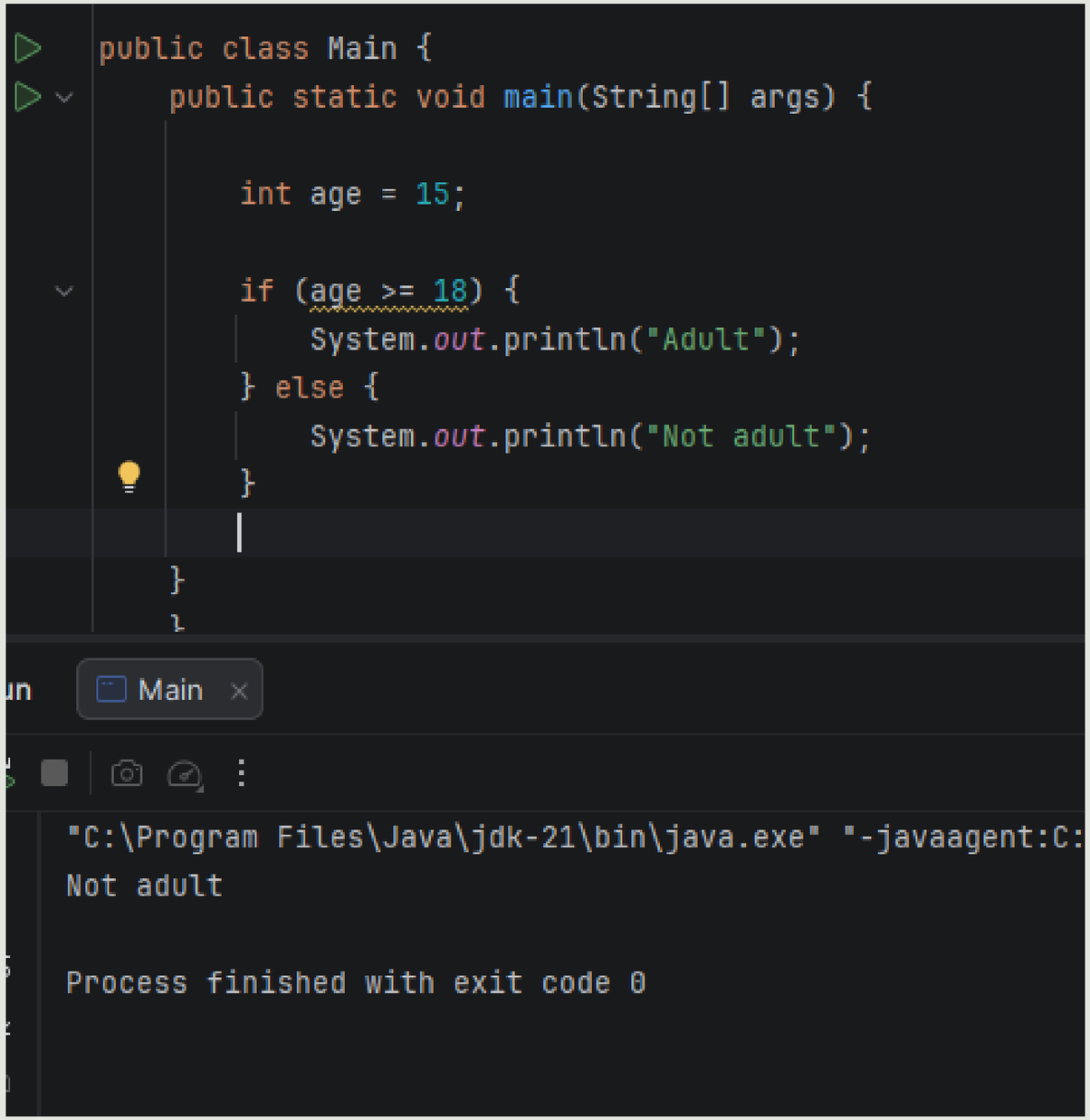
Conditional statements:
1-if
2-if else
3-nested if
4-switch case

The if statement is used to execute a block of code only if a condition is true.

```
if (condition) {
    // code runs if condition is true
}
```

Note : if it only single line of code , you can ignore braces.

if-else


public class Main {
public static void main(String[] args) {
int age = 15;
if (age >= 18) {
System.out.println("Adult");
} else {
System.out.println("Not adult");
}
}
Main
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:
Not adult
Process finished with exit code 0

if – else if – else

```java
public class Main {
    public static void main(String[] args) {

        int marks = 75;

        if (marks >= 90) {
            System.out.println("A");
        } else if (marks >= 75) {
            System.out.println("B");
        } else {
            System.out.println("C");
        }
```

Main

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C
B

Process finished with exit code 0
```

Tricky codes

```java
public class Main {
    public static void main(String[] args) {

        int x = 5;

        if (x = 10) {
            System.out.println("Inside if");
        }
    }
}
```

Build Output

va course: build failed At 1/8/2026 8:30 PM with 12 sec, 720 ms

Main.java src 1 error

incompatible types: int cannot be converted to boolean :7

```
java: incompatible types: int cannot be converted to boolean
```

Why?

= is an assignment operator .

if needs a boolean condition.

x = 10 returns an int, not Boolean.

Tricky codes

```java
public class Main {
    public static void main(String[] args) {
        boolean flag = false;

        if (flag = true) {
            System.out.println("Hello");
        }

        }
}
```

Main

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDE
Hello

Process finished with exit code 0
```

Why?

flag = true assigns true to flag
The expression becomes true
if block always runs

Tricky codes

```java
public class Main {
    public static void main(String[] args) {

        int x = 5;

        if (x > 10);
        {
            System.out.println("Hello");
        }


    }
}
```

```
Main

"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaa
Hello
```

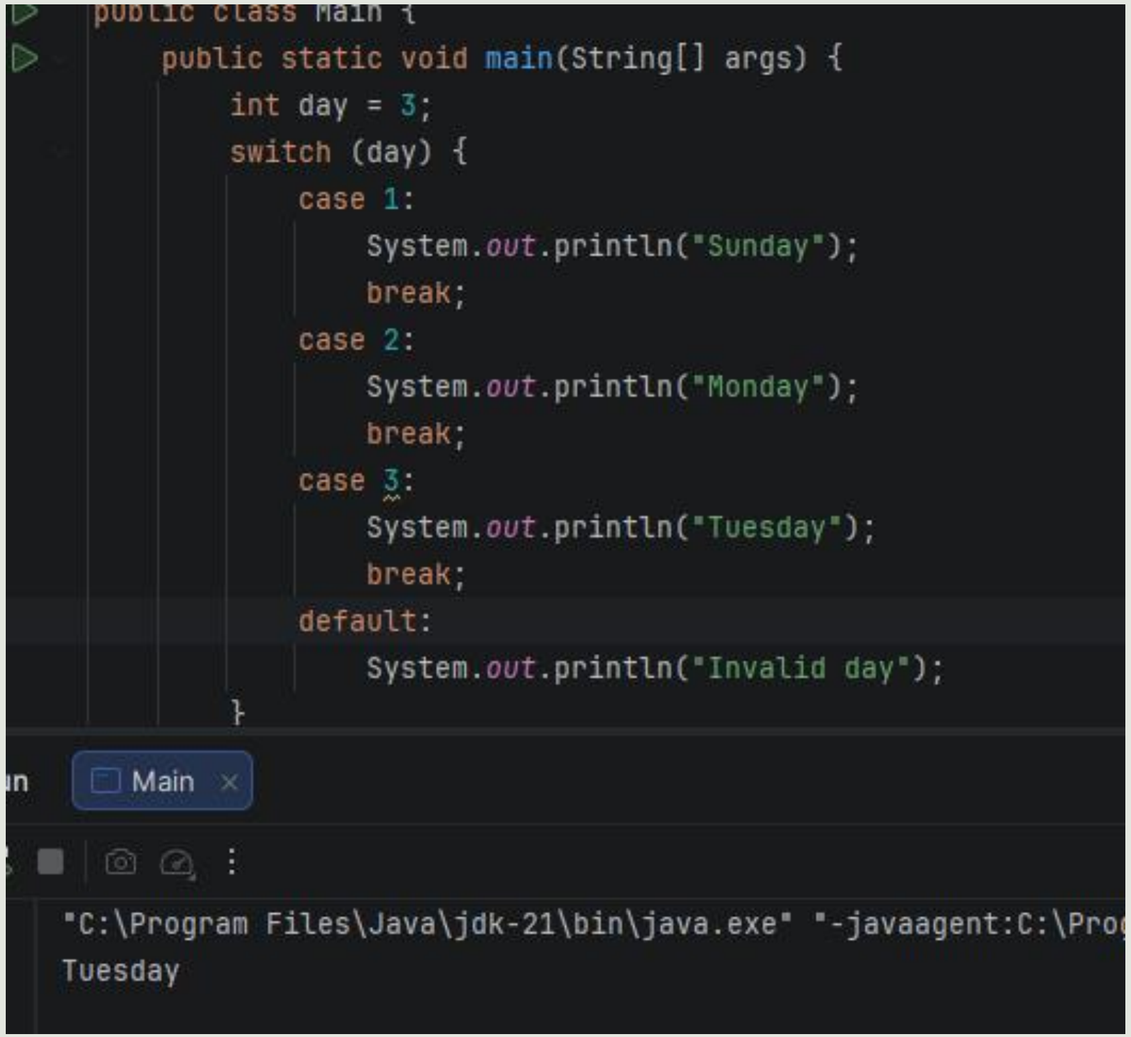
Why?

The ; ends the if statement
The block {} runs every time

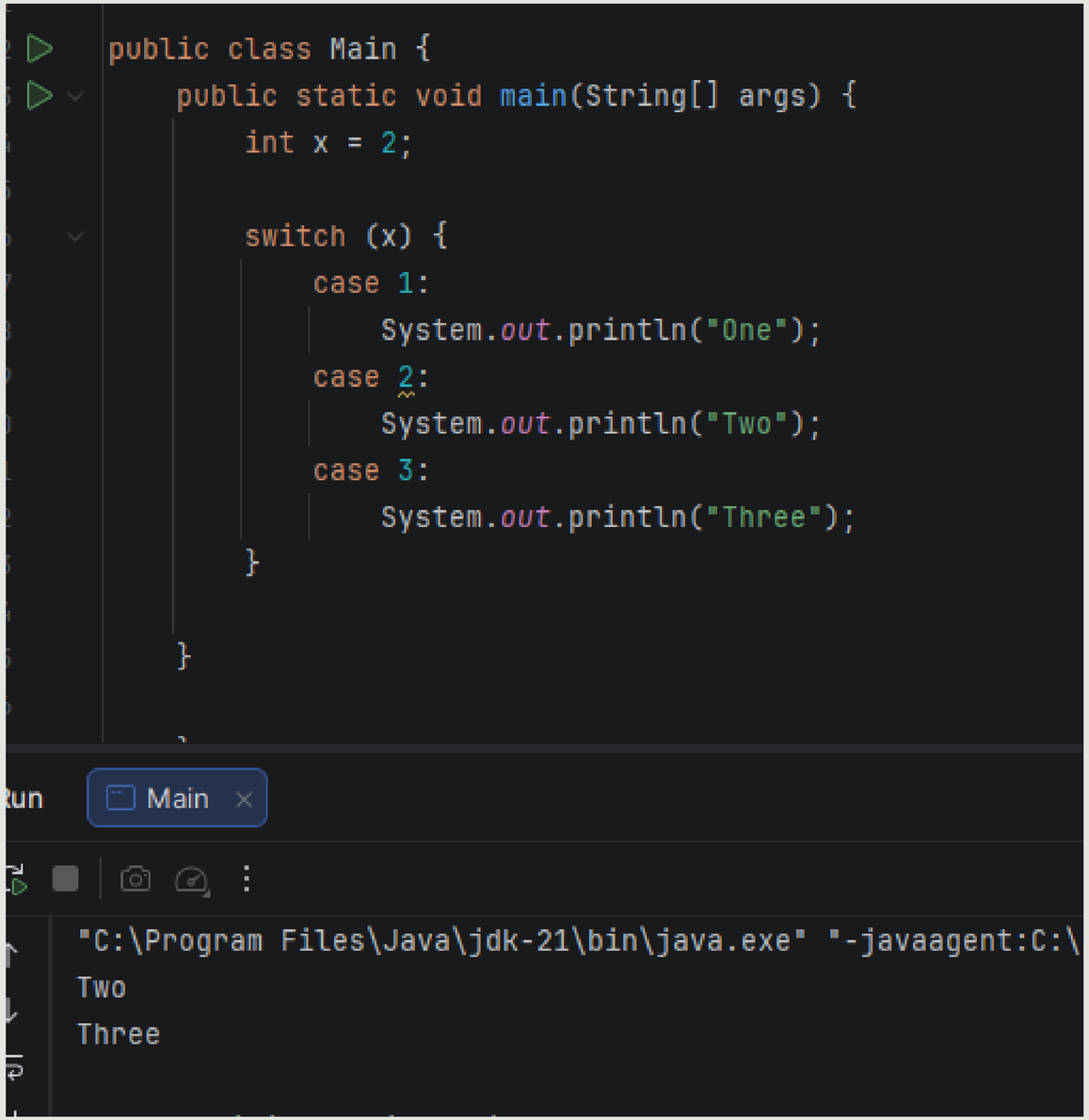
Switch

It is used to execute one block of code from multiple choices based on the value of an expression.

```
switch (expression) {
    case value1:
        // code
        break;
    case value2:
        // code
        break;
    default:
        // code
}
```



Tricky code



Because there is no break, execution continues.

Notes :

-switch expression can be : int, char, String, enum
-case values must be constant
-default is optional
-break stops execution

Loops

-The **while** loop is used to repeat a block of code while a condition is true.
-The condition is checked before each iteration.
-If the condition is **false** initially, the code **never executes**

```
while (condition) {
    // code to repeat
}
```


public class Main {
public static void main(String[] args) {
int i = 1;
while (i <= 5) {
System.out.println("i = " + i);
i++; // increment is important to avoid infinite loop
}
}
}
Run
Main
i = 1
i = 2
i = 3
i = 4
i = 5

Loop never stops → infinite loop

```java
int i = 1;

while (i <= 5) {
    System.out.println("Hello");
    // i is never incremented
}
```

```java
public class Main {
    public static void main(String[] args) {

        int i = 1;

        while (true) {  // infinite loop
            System.out.println(i);
            if (i == 5) {
                break;  // stops the loop
            }
            i++;
        }
    }
}
```

Run Main

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:
1
2
3
4
5
```

Note : when code is only one line you can ignore braces.

Tricky code

```java
public class Main {
    public static void main(String[] args) {
        int i = 1;

        while (i <= 3)
            System.out.println(i); //infinite loop printing 1
        i++;

    }
}
```

Why?

-Only System.out.println(i); is inside the loop.

-i++ is outside the loop

Condition never changes.

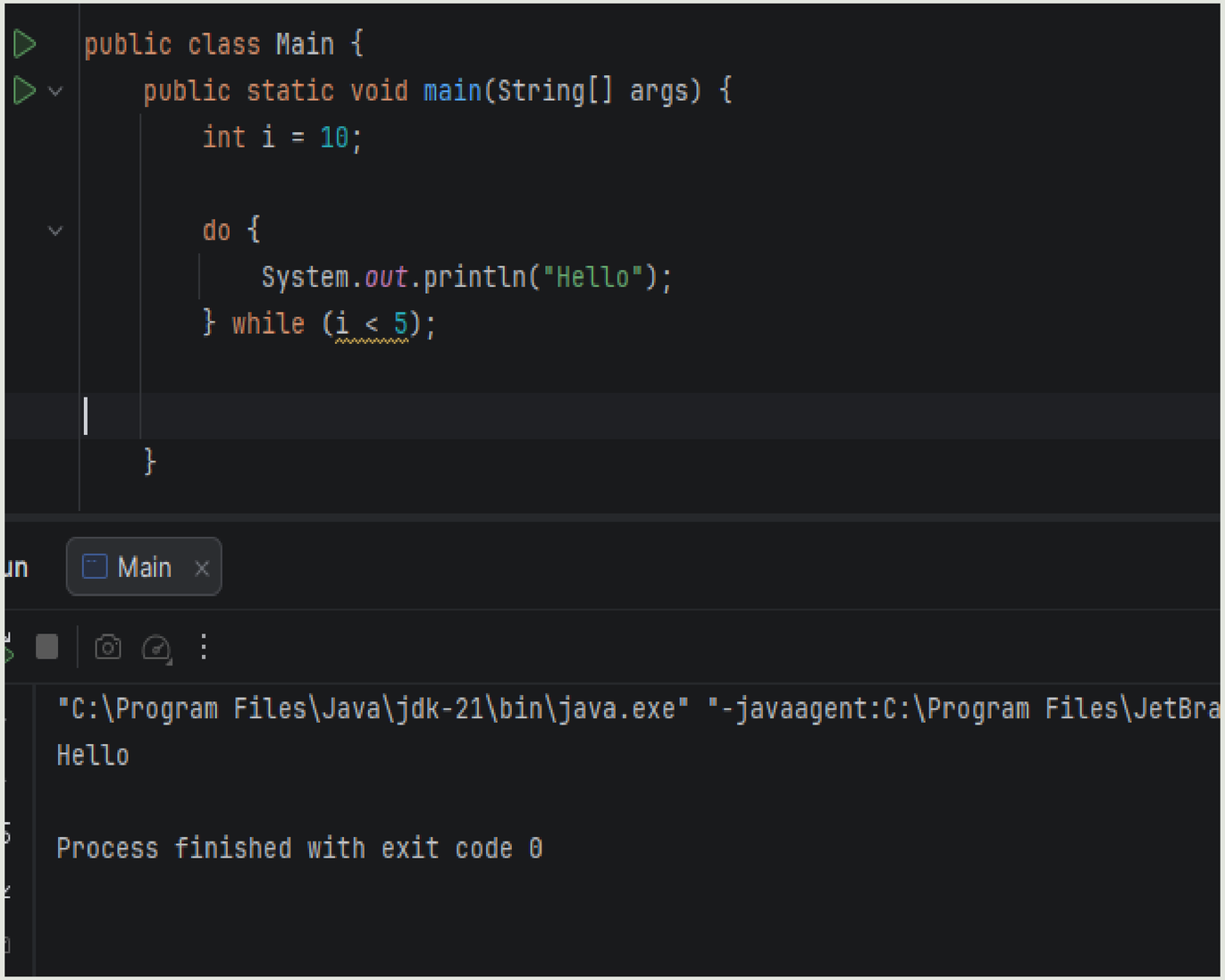
Do-while

-The do-while loop executes a block of code at least once and then repeats it while the condition is true.

```
do {
    // code to repeat
} while (condition);
```

-Take care of The semicolon ; after while(condition)
-The loop always runs at least once, even if the condition is false initially.


public class Main {
public static void main(String[] args) {
int i = 1;
do {
System.out.println("i = " + i);
i++;
} while (i <= 5);
}
Run
Main
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Pro
i = 1
i = 2
i = 3
i = 4
i = 5



When condition is false

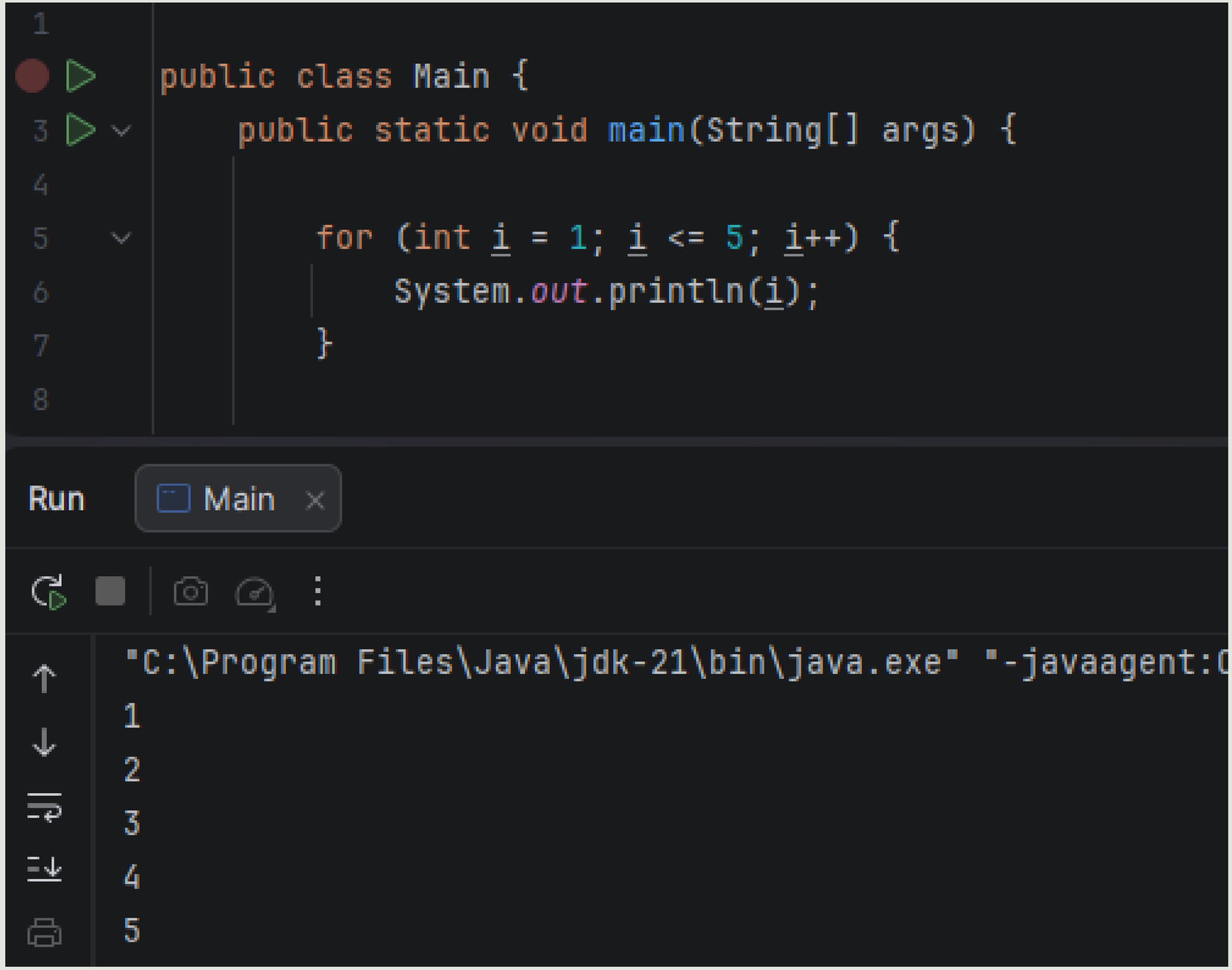
For Loop

-The for loop is used when you know how many times you want to repeat a block of code.

```
for (initialization; condition; update) {
    // code
}
```

Parts:

initialization → runs once at the start
condition → checked before every iteration
update → runs after each loop execution


1
public class Main {
3
public static void main(String[] args) {
4
5
for (int i = 1; i <= 5; i++) {
6
System.out.println(i);
7
}
8
Run
Main
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C
1
2
3
4
5

Tricky code

```
2   public class Main {
3       public static void main(String[] args) {
5           for (int i = 1; i <= 3; i++);
6           {
7               System.out.println("Hello");
8           }
9
10
```

Run Main

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Fil
Hello
```

Why?

The ; ends the for loop

The block {} runs once only

Break statement

```java
public class Main {
    public static void main(String[] args) {

        for (int i = 1; i <= 5; i++) {
            if (i == 3) {
                break;
            }
            System.out.println(i);
        }

    }
}
```

Run Main

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagen
1
2

Process finished with exit code 0
```

break → **exits loop**

continue statement

```java

public class Main {
    public static void main(String[] args) {

        for (int i = 1; i <= 5; i++) {
            if (i == 3) {
                continue;
            }
            System.out.println(i);
        }

```

Run Main

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program
1
2
4
5
```

continue → **skips current iteration**

Thank You