Exception Handling

- **Exception**: An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions.
    - Examples include divide-by-zero errors, file not found errors, or null pointer exceptions.
- **Exception Handling**: It is a mechanism to handle runtime errors, ensuring the program continues to run smoothly without abrupt termination.

Types of Exceptions

Exceptions

Checked Exceptions

- These exceptions are checked at compile-time.
- If a method is likely to throw a checked exception, it must declare this fact using the throws keyword, or handle it using a try-catch block.
- Examples include `IOException`, `SQLException`,

Unchecked Exceptions

- These are not checked at compile-time.
- These exceptions occur due to programming errors, such as logic errors or improper use of an API.
- Examples include `NullPointerException`, `ArithmeticException`, `ArrayIndexOutOfBoundsException`, etc.

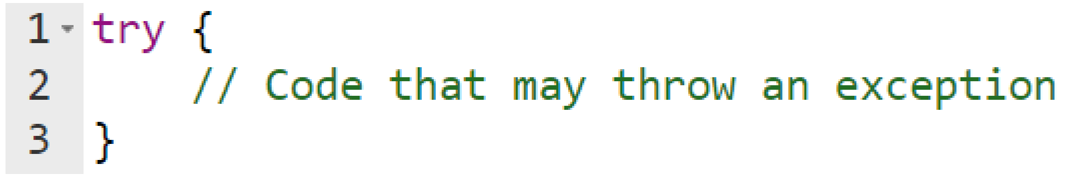
Error

- Errors are exceptions that occur due to environment issues.
- They are typically not handled in the program as they are beyond the control of the application.
- Examples include `OutOfMemoryError`, `VirtualMachineError`, etc.

Exception Handling Keywords in Java

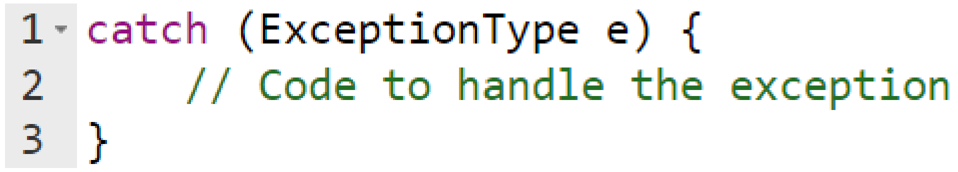
Java provides five keywords for exception handling: `try, catch, finally, throw, throws`

`try`: A block of code where exceptions can occur. This block is followed by either `catch` or `finally` blocks.



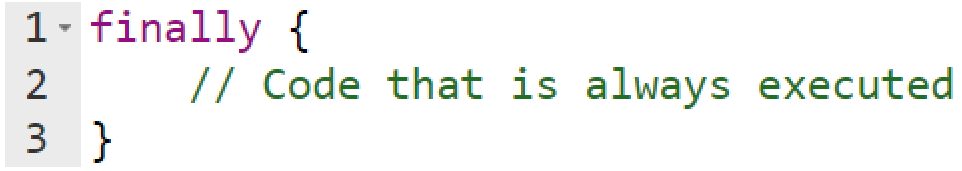
```java
try {
    // Code that may throw an exception
}
```

`catch`: A block of code that handles the exception thrown by the `try` block. It is used to handle specific exceptions and can be followed by multiple `catch` blocks for different exception types.



```java
catch (ExceptionType e) {
    // Code to handle the exception
}
```

`finally`: A block that is always executed after the `try` block, regardless of whether an exception was thrown or caught. It is typically used to close resources like files or database connections.



```java
finally {
    // Code that is always executed
}
```

Exception Handling Keywords in Java

Java provides five keywords for exception handling: `try`, `catch`, `finally`, `throw`, `throws`

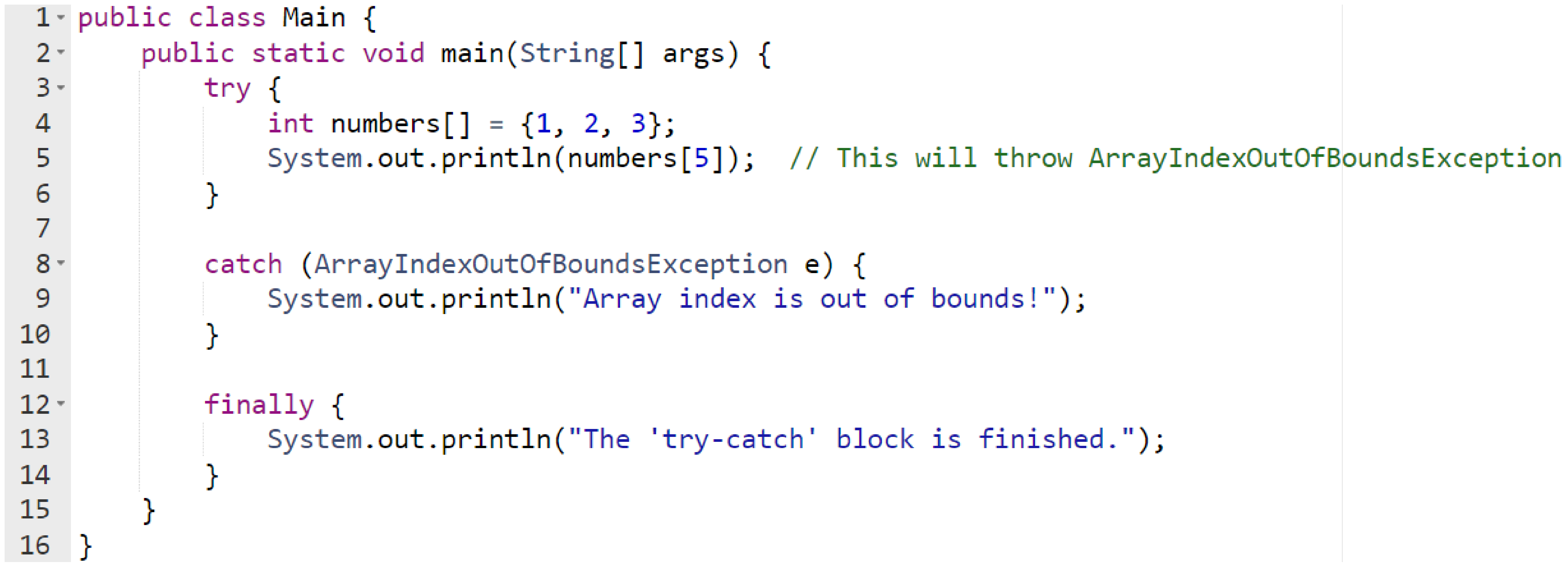
`throw`: Used to explicitly throw an exception. The `throw` keyword is followed by an instance of `Throwable` or its subclasses.

```java
throw new ExceptionType("Error message");

```

`throws`: Used in a method signature to indicate that this method may throw certain exceptions. It is followed by a comma-separated list of exceptions.

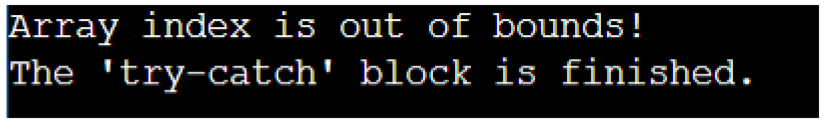
```java
public void myMethod() throws IOException, SQLException {
    // Code that may throw IOException or SQLException
}
```

Basic Example of Exception Handling



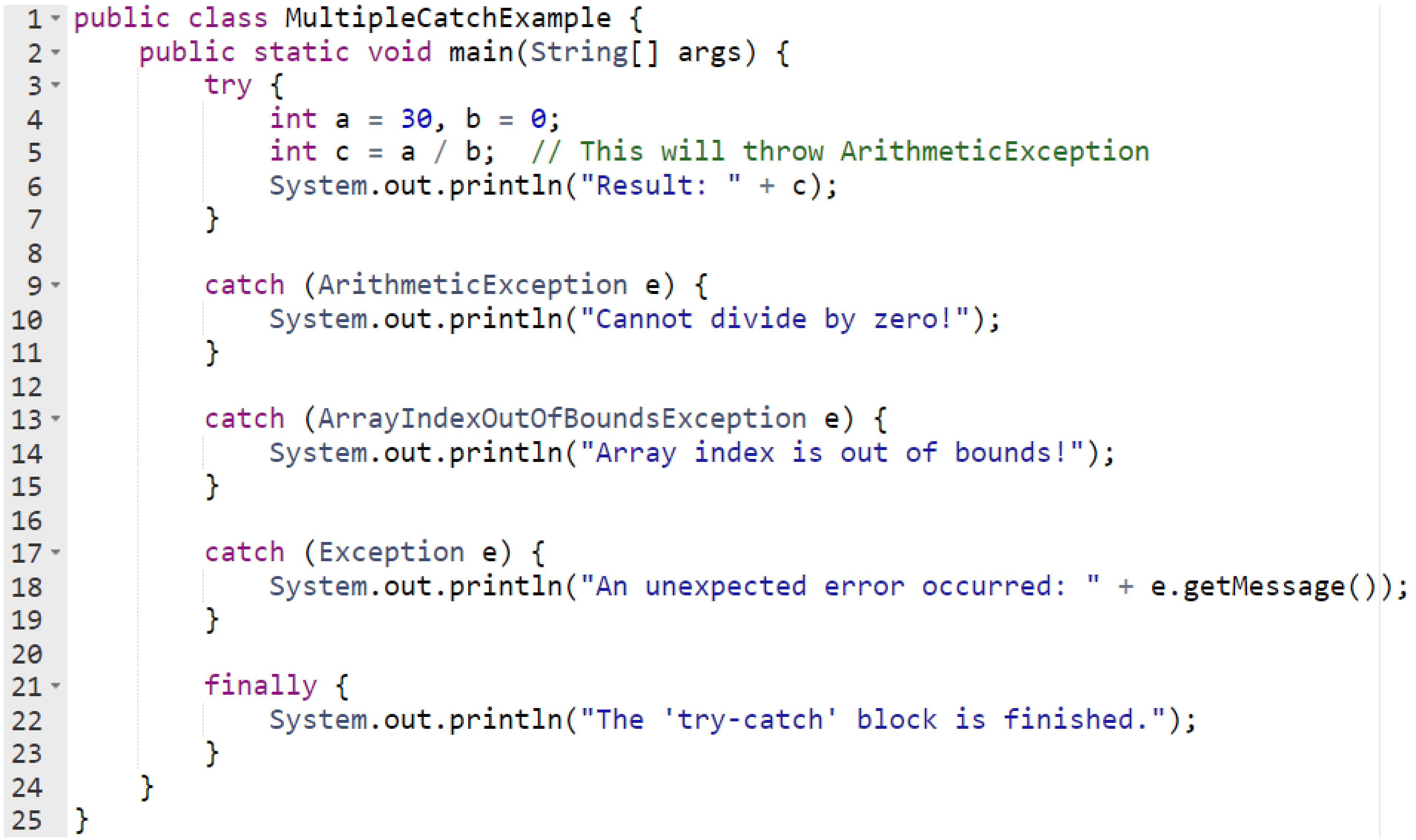
```java
public class Main {
    public static void main(String[] args) {
        try {
            int numbers[] = {1, 2, 3};
            System.out.println(numbers[5]);  // This will throw ArrayIndexOutOfBoundsException
        }

        catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index is out of bounds!");
        }

        finally {
            System.out.println("The 'try-catch' block is finished.");
        }
    }
}
```

Output:

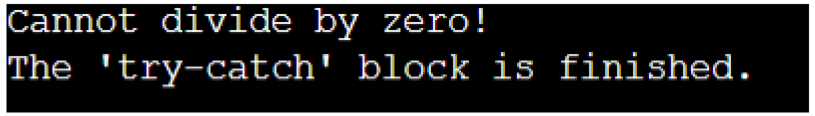


```
Array index is out of bounds!
The 'try-catch' block is finished.
```

Multiple Catch Blocks



```java
public class MultipleCatchExample {
    public static void main(String[] args) {
        try {
            int a = 30, b = 0;
            int c = a / b;  // This will throw ArithmeticException
            System.out.println("Result: " + c);
        }

        catch (ArithmeticException e) {
            System.out.println("Cannot divide by zero!");
        }

        catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index is out of bounds!");
        }

        catch (Exception e) {
            System.out.println("An unexpected error occurred: " + e.getMessage());
        }

        finally {
            System.out.println("The 'try-catch' block is finished.");
        }
    }
}
```



```
Cannot divide by zero!
The 'try-catch' block is finished.
```