Implement Exception Handling

Menu

00:01:00:00:39 01/ 39

Slide Note

Think and Tell

- There could be multiple reasons for a washing machine to stop working abruptly. Power failure, insufficient water supply, or something getting stuck in the machine could be a few reasons.
- How do you think the machine should respond in such situations?

Slide Note

Let us Discuss

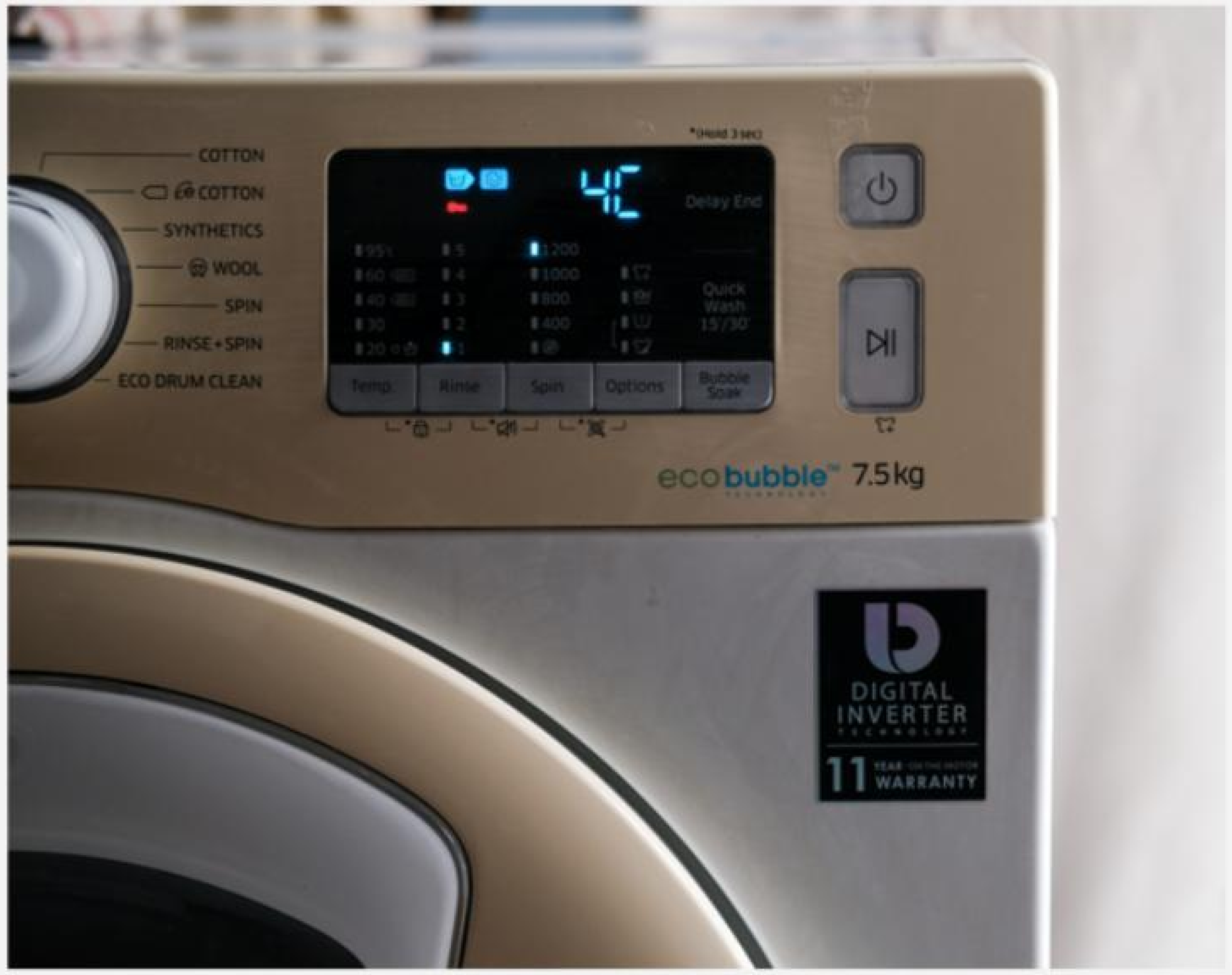
- Would you like your machine to start the wash cycle all over again or resume it from the point where it stopped?
- Would it help if your machine could remember the temperature, water level, and other settings?

Menu

00:03:00:00:39 03/ 39

Slide Note

Error Codes on a Washing Machine



- Have you ever seen this code on your washing machine?

Menu

00:04 00:00:39 04/ 39

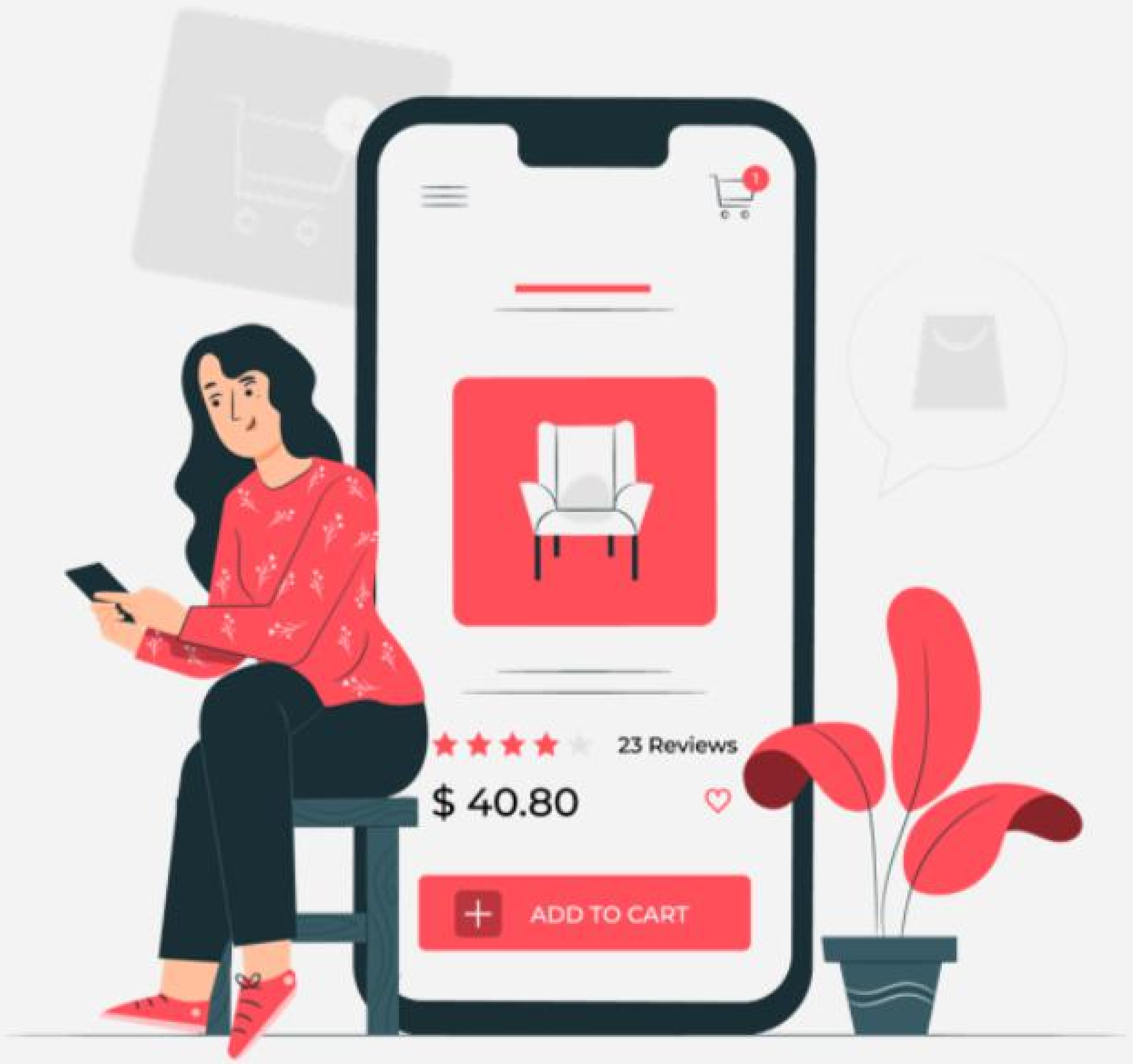
Slide Note

Online Purchase

- After spending an hour browsing the web and selecting a product to be purchased online, you try to make the payment.

- Imagine a scenario where the product is out of stock.

- What would have been a better way for the system to respond?



Menu

00:05 00:00:39 05/ 39

Slide Note

Let's Discuss

- What message would you, as a user, wish to see in such situations?
- When do you think receiving such messages would help you the most? Give reasons.

Menu

Slide Note

Learning Objective

- Define errors in coding
- Introduction to Exception
- Describe and list the types of exceptions
- Define Unchecked Exception
- Use try, catch and finally blocks
- Explain try with multiple catch

Menu

00:07:00:00:39 07/ 39

Slide Note

Define Errors in Coding

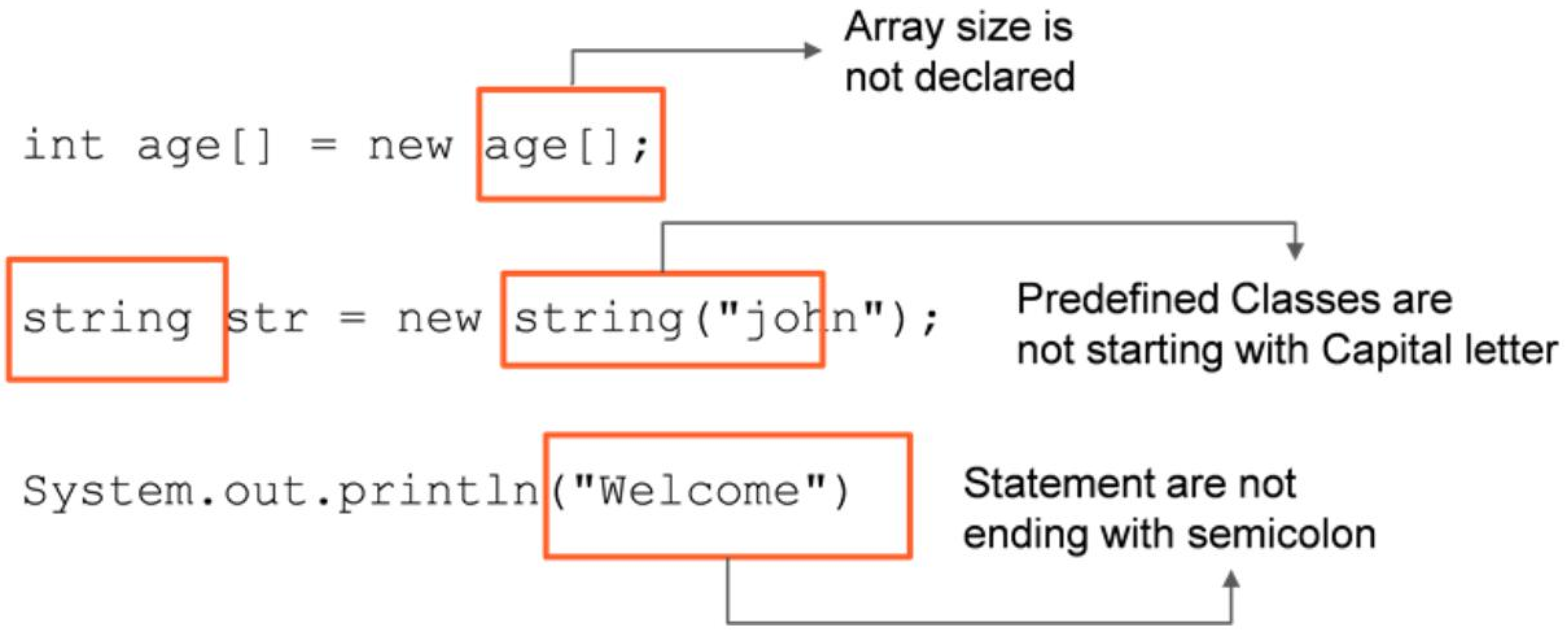
Menu

00:08:00:00:39 08/ 39

Slide Note

Syntax Errors While Coding

- Syntax errors usually occur when language rules are not followed.



```
int age[] = new age[];

string str = new string("john");

System.out.println("Welcome")
```

Array size is not declared
Predefined Classes are not starting with Capital letter
Statement are not ending with semicolon

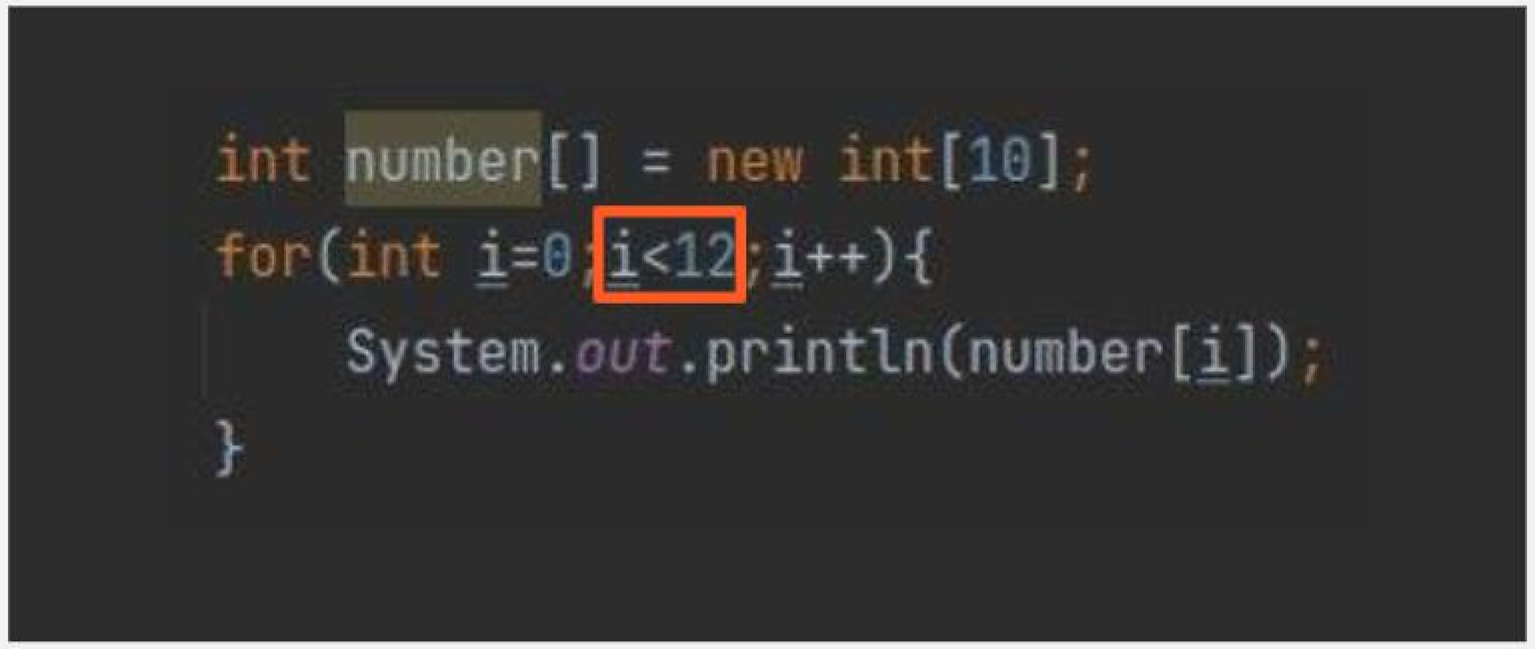
- Syntax errors are compile-time errors.
- The program will not run until syntax errors are resolved.

Slide Note

Logical Error While Coding

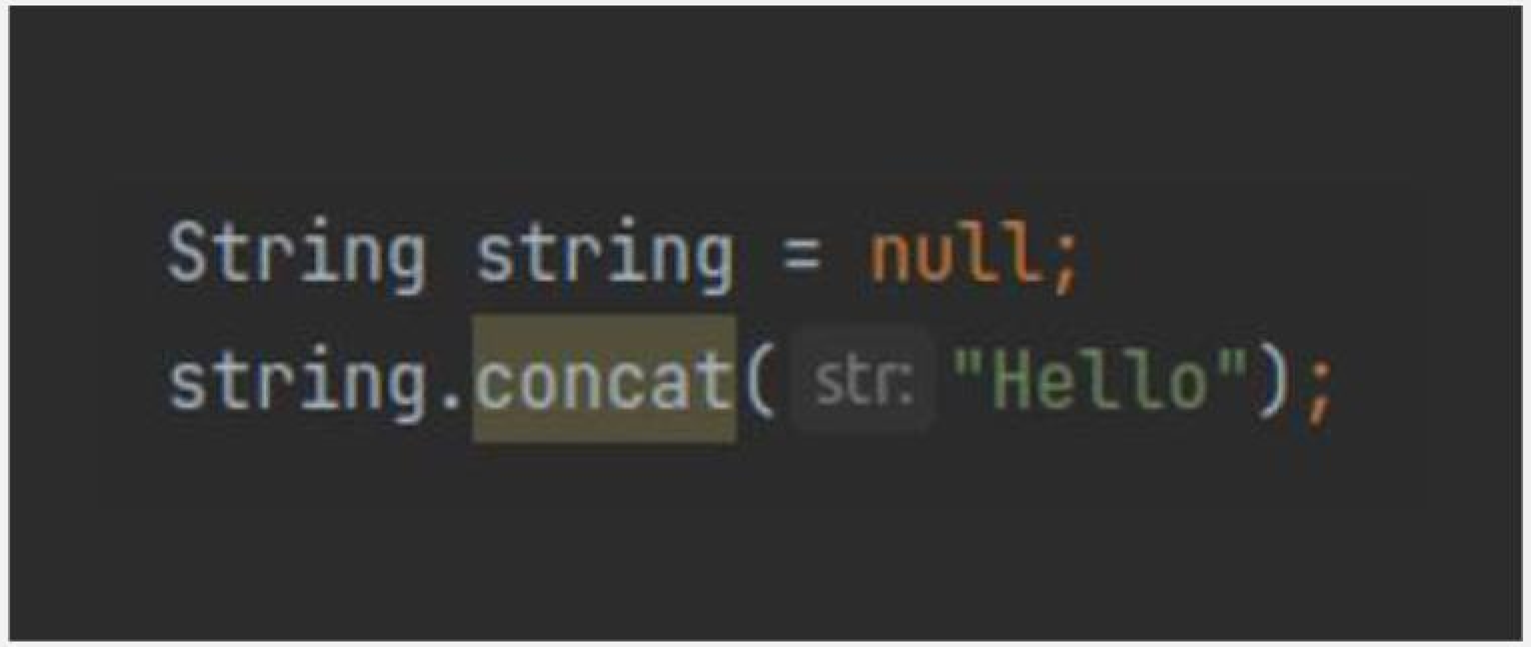
- Logical errors do not allow a program to behave as expected.
- Logical errors are programmer-specific errors.
- A program with a logical error might not give any errors during compile time.
- Logical errors will cause problems during runtime.
- Since logical errors are detected at runtime or execution time, they must be handled appropriately.
- Java provides an exception-handling mechanism to handle such runtime errors.

The `for` loop iteration condition exceeds the actual size of the array.



```
int number[] = new int[10];
for(int i=0;i<12;i++){
    System.out.println(number[i]);
}
```

The `String` Object is not initialized, but the reference variable `string` is calling the method of the `String` class.



```
String string = null;
string.concat( str: "Hello");
```

Menu

00:10 00:00:39 10/ 39

Slide Note

Introduction to Exception

Menu

00:11 00:00:39 11/ 39

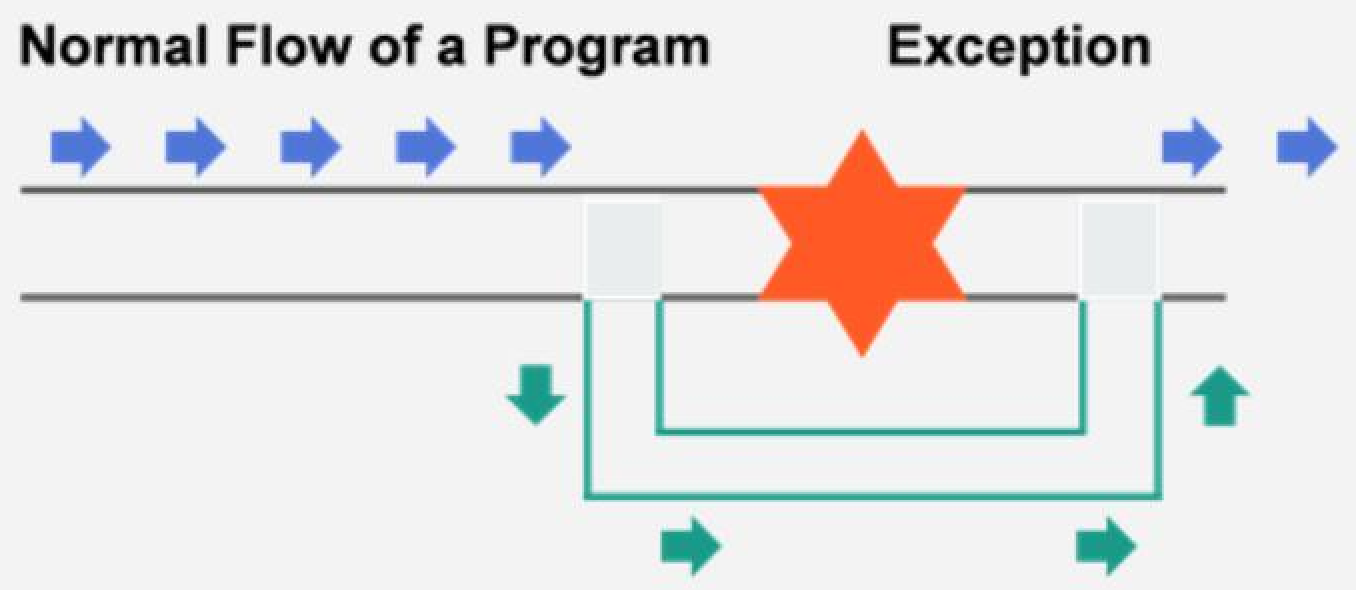
Slide Note

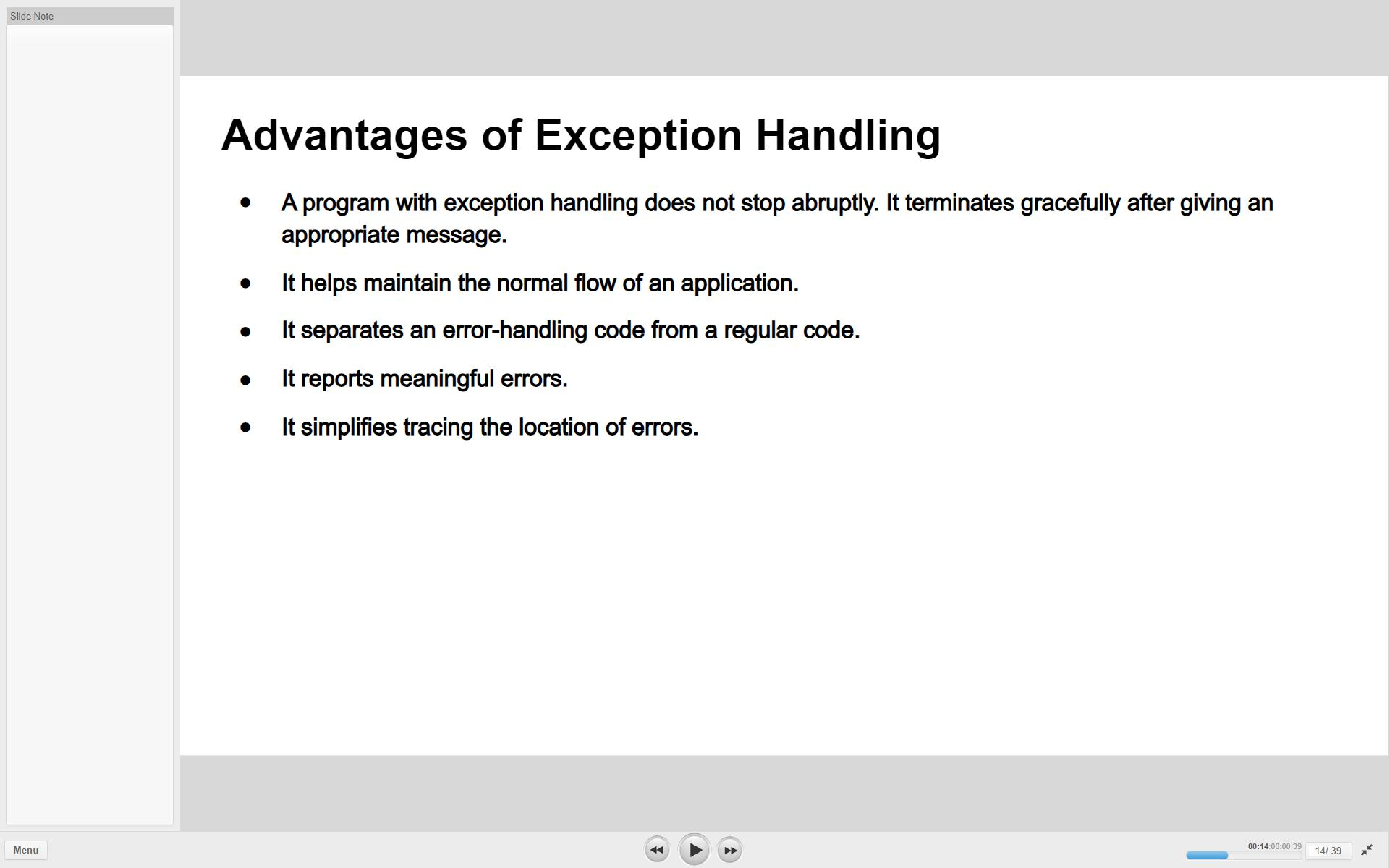
What Are Exceptions?

- An exception is an unwanted event that occurs during the execution of a program and disrupts the normal flow of the program.
- An Exception describes an exceptional condition that occurs in a piece of code.
- An exception can happen either during the compilation or execution phases.

Exception Handling

- Exception handling is a mechanism that helps a programmer bypass an exceptional situation, allowing the program to flow instead of abruptly terminating.





Advantages of Exception Handling

- A program with exception handling does not stop abruptly. It terminates gracefully after giving an appropriate message.
- It helps maintain the normal flow of an application.
- It separates an error-handling code from a regular code.
- It reports meaningful errors.
- It simplifies tracing the location of errors.

Slide Note

Exception Handling in Java

- Java provides a rich set of classes for exception handling.
- All Java programs that generate an exception will throw an object of the parent class of the exception or its sub-classes.
- The parent class is `Throwable`, with multiple subclasses like `Exception`, `Error`, etc. It resides in `java.lang` package.

Slide Note

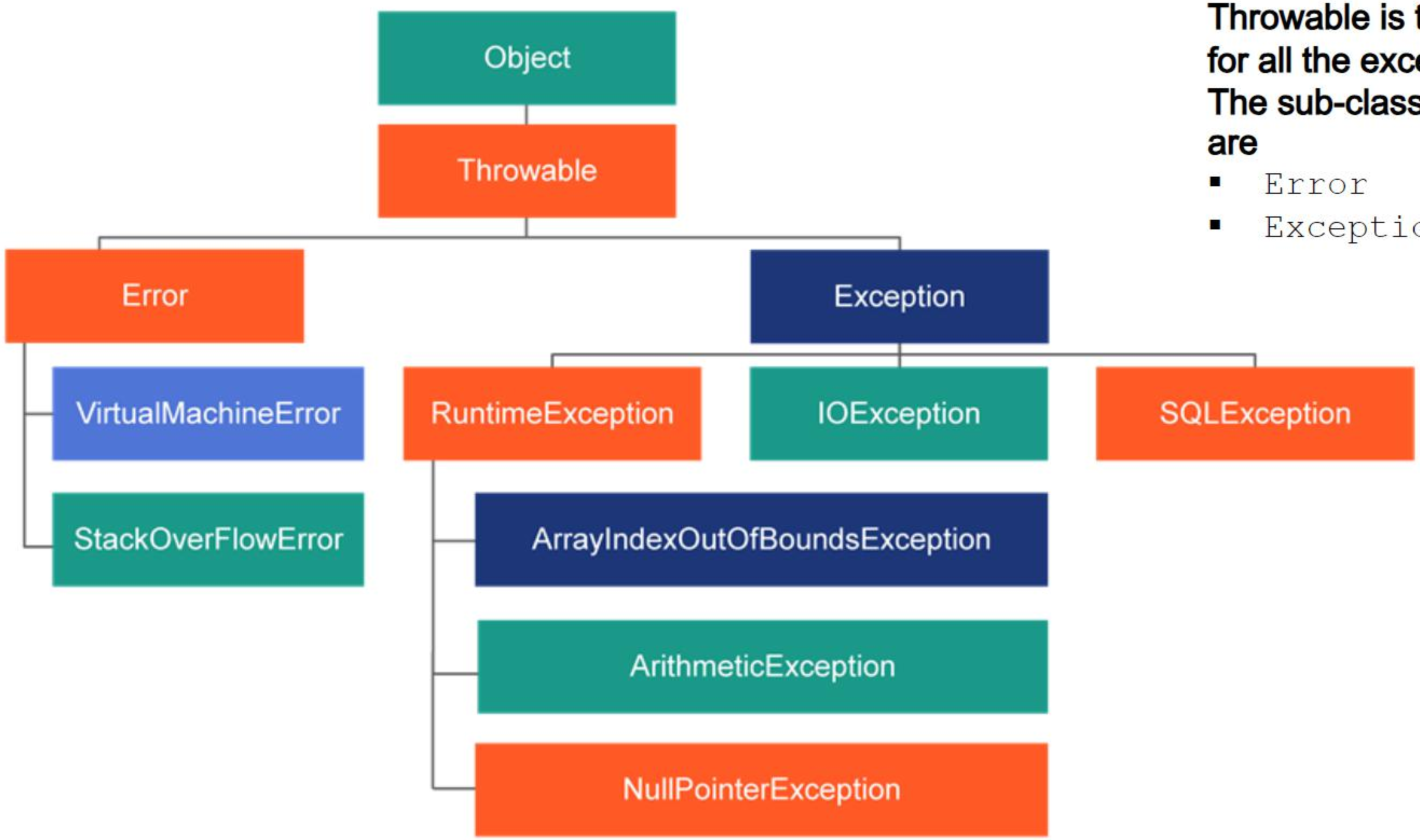
Describe and List The Types of Exceptions

Menu

00:16 00:00:39 16/ 39

Slide Note

The Exception Class Hierarchy



Throwable is the super class for all the exceptions.
The sub-classes of Throwable are

- `Error`
- `Exception`

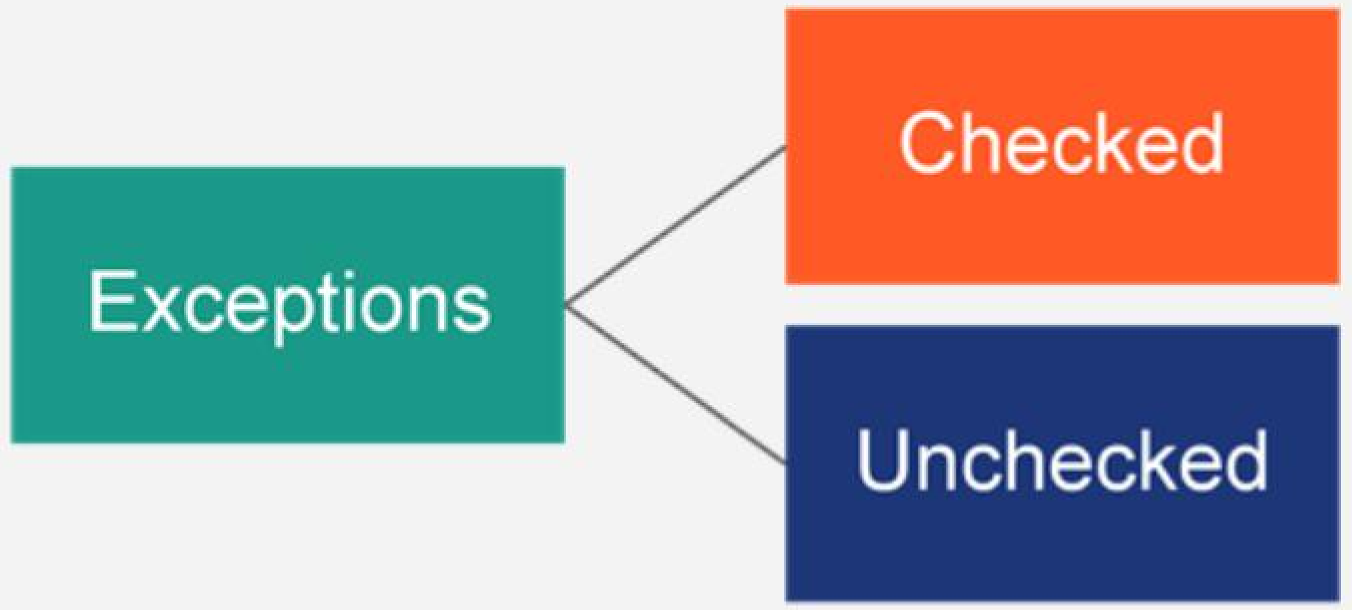
Slide Note

Errors in Exception Hierarchy

- Errors are conditions during the execution of a program that terminates the execution of the program.
- Error situations like memory overflow or a system failure are out of the control of the programmer and cannot be handled.
- The programmer must not handle errors.
- Some examples of error situations are:
  - VirtualMachineError
  - StackOverFlowError

Menu

Slide Note



Types of Exceptions

- Exceptions must be handled by the programmer.
- There are two types of exceptions in Java:
  - Checked or Compile-time exceptions like `IOException`, `SQLException`, etc.
  - Unchecked or Runtime exceptions like `NullPointerException`, `ArrayIndexOutOfBoundException`, etc.

Note: Checked exceptions will be discussed in the next Sprint.

Slide Note

Define An Unchecked Exception

Menu

00:20 00:00:39 20/ 39

Slide Note

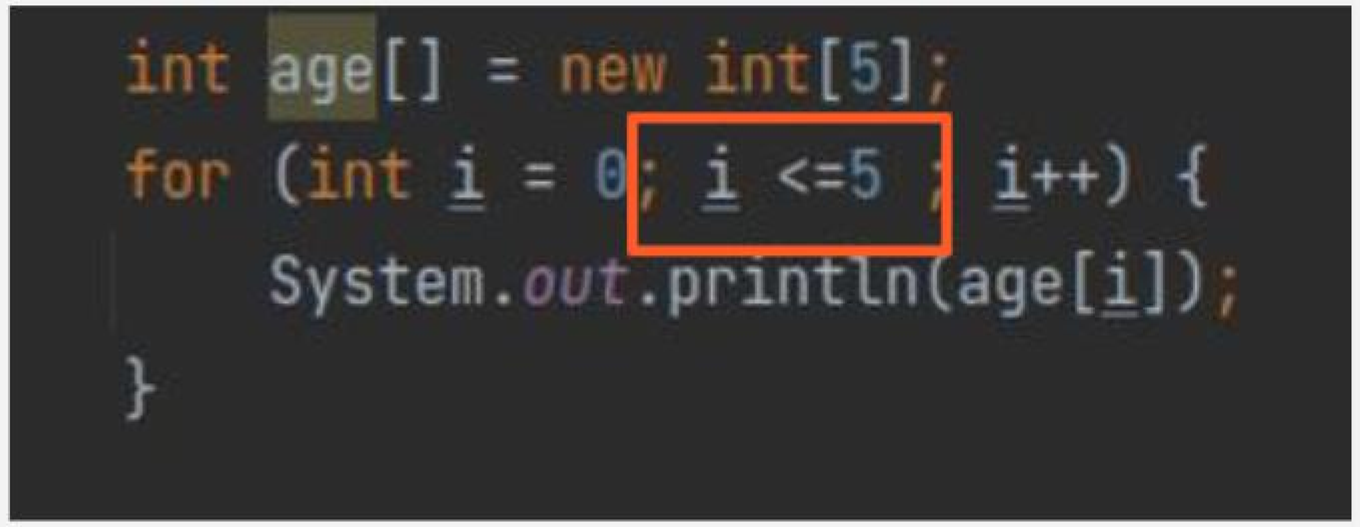
Unchecked Exceptions

- Runtime exceptions or unchecked exceptions occur due to bad programming.
- For example, while trying to retrieve an element from an array, the length of the array must be checked first before trying to retrieve the element; otherwise, it might throw an `ArrayIndexOutOfBoundException` at runtime or execution time.
- Runtime exceptions can be avoided with better programming.

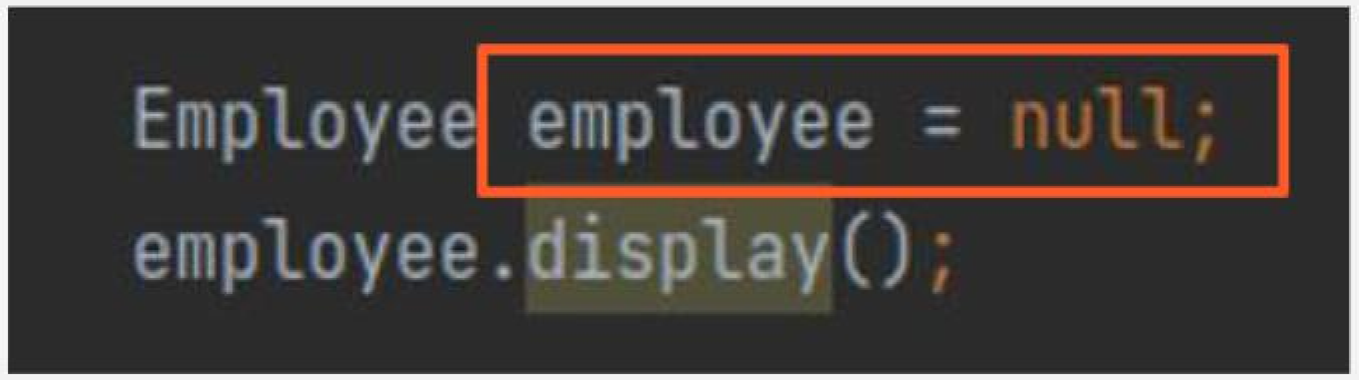
Menu

Slide Note

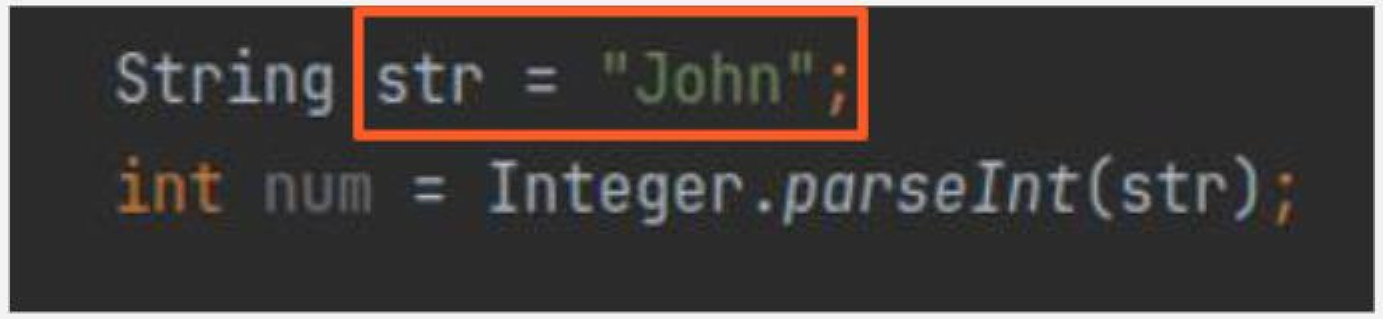
Unchecked Exceptions (Cont'd)



```
int age[] = new int[5];
for (int i = 0; i <=5 ; i++) {
    System.out.println(age[i]);
}
```



```
Employee employee = null;
employee.display();
```



```
String str = "John";
int num = Integer.parseInt(str);
```

- Iterating an array more than its size will give an `ArrayIndexOutofBoundException`, a runtime exception.
- Creating an employee reference without initializing the object will result in a `NullPointerException`.
- Parsing a string that contains characters instead of integers will throw a `NumberFormatException`.
- All these are examples of bad programming practices.

Menu

00:22 00:00:39 22/ 39

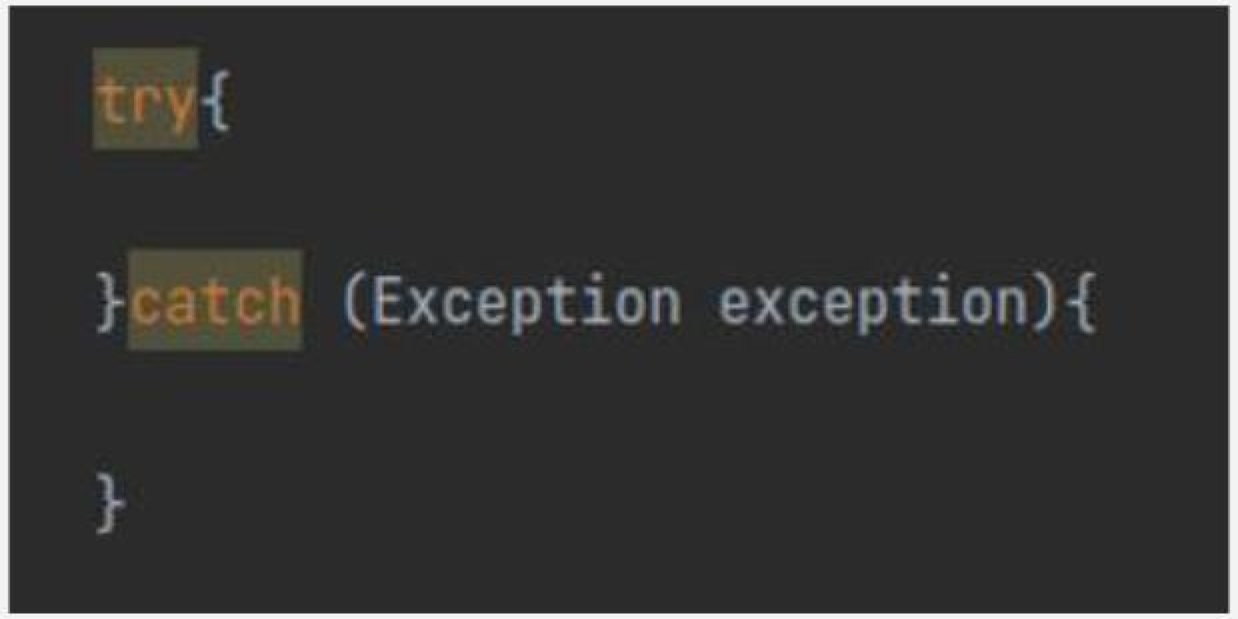
Slide Note

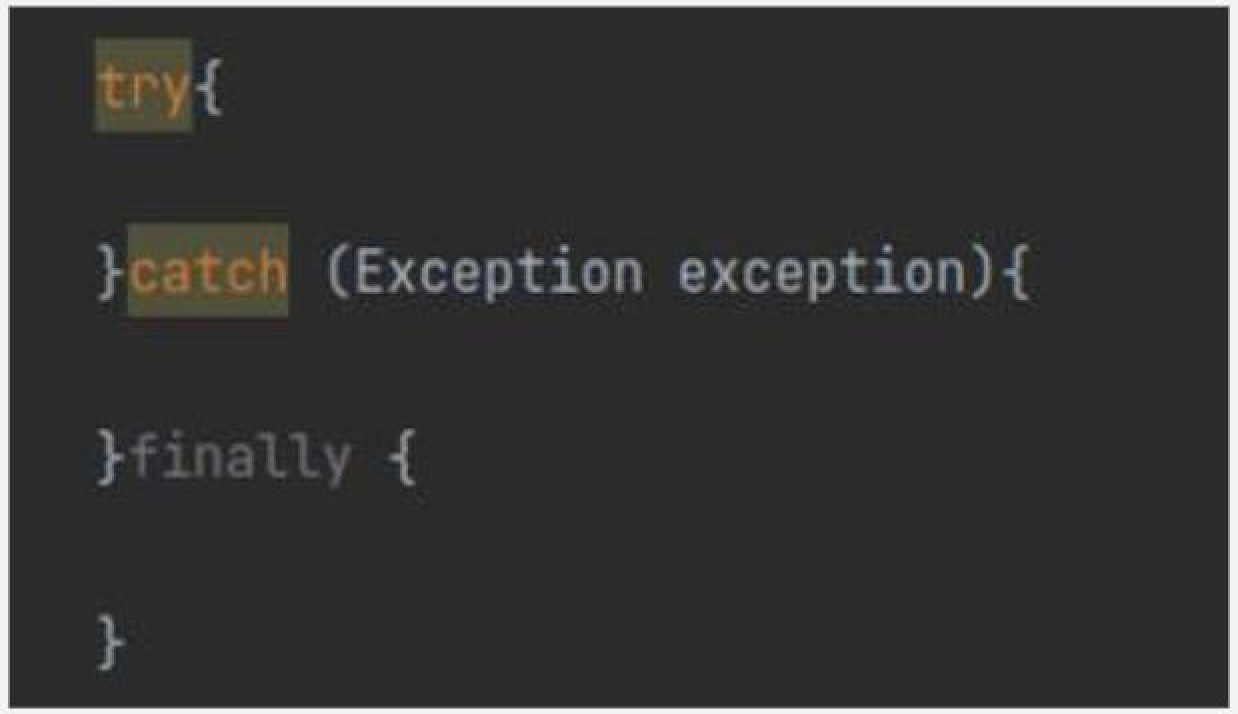
Use Try, Catch, and Finally Blocks

Menu

00:23 00:00:39 23/ 39

Slide Note


```
try{

}catch (Exception exception){

}
```


```
try{

}catch (Exception exception){

}finally {

}
```

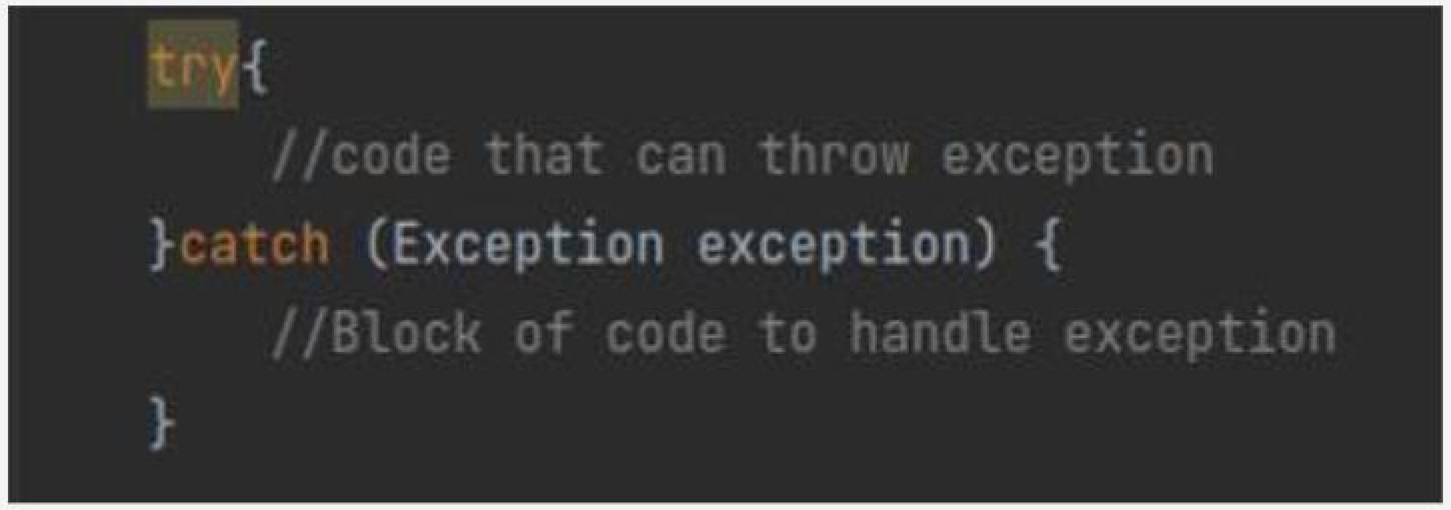
Exception Handlers in Java

- The following blocks are used to handle exceptions in Java:
  - `try – catch`
  - `try-catch-finally`

Slide Note

Exception Handlers in Java Using `try-catch`

- The try block is followed by a catch block.
- In the try block, all the code that is likely to generate an exception is written.
- The catch block defines code to handle the exception that occurred in the try block.
- The catch block takes an Exception Object as a parameter.



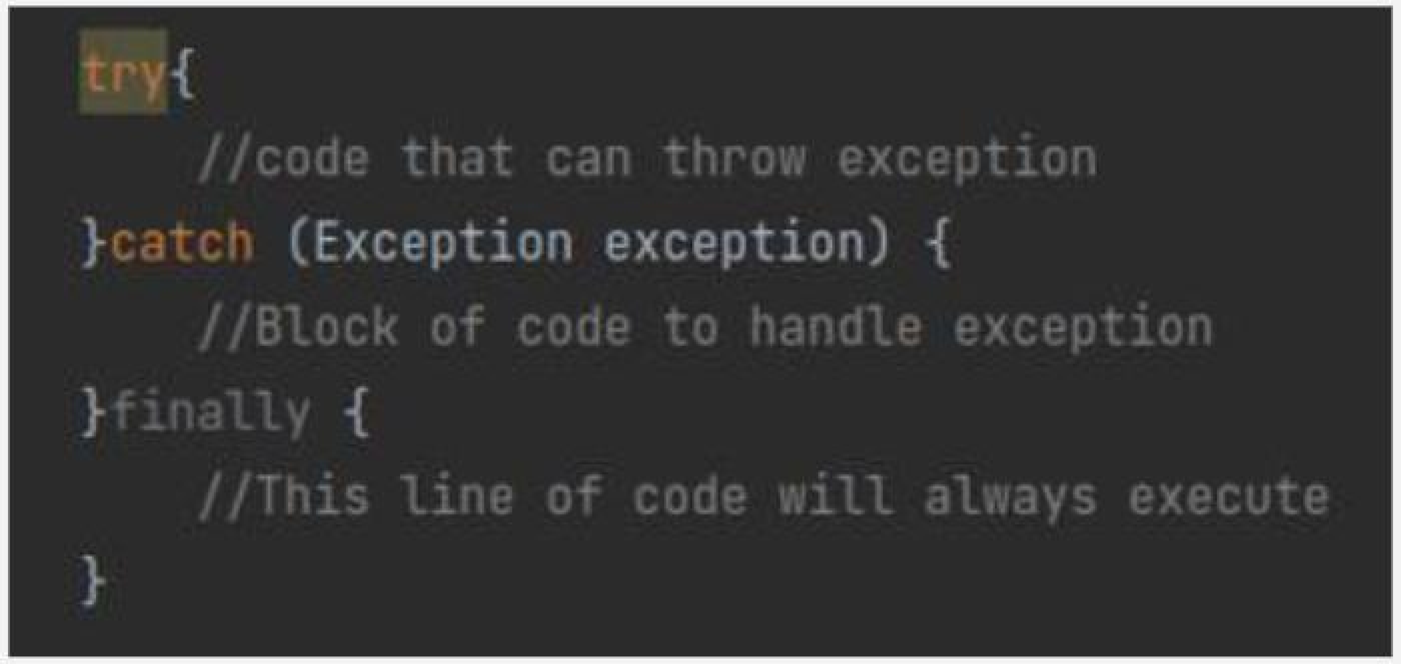
```
try{
    //code that can throw exception
}catch (Exception exception) {
    //Block of code to handle exception
}
```

Menu

Slide Note

Exception Handlers in Java Using `try-catch-finally`

- Code in the `finally` block in Java is executed whether an exception occurs or not.
- A finally block follows a try or a catch block.



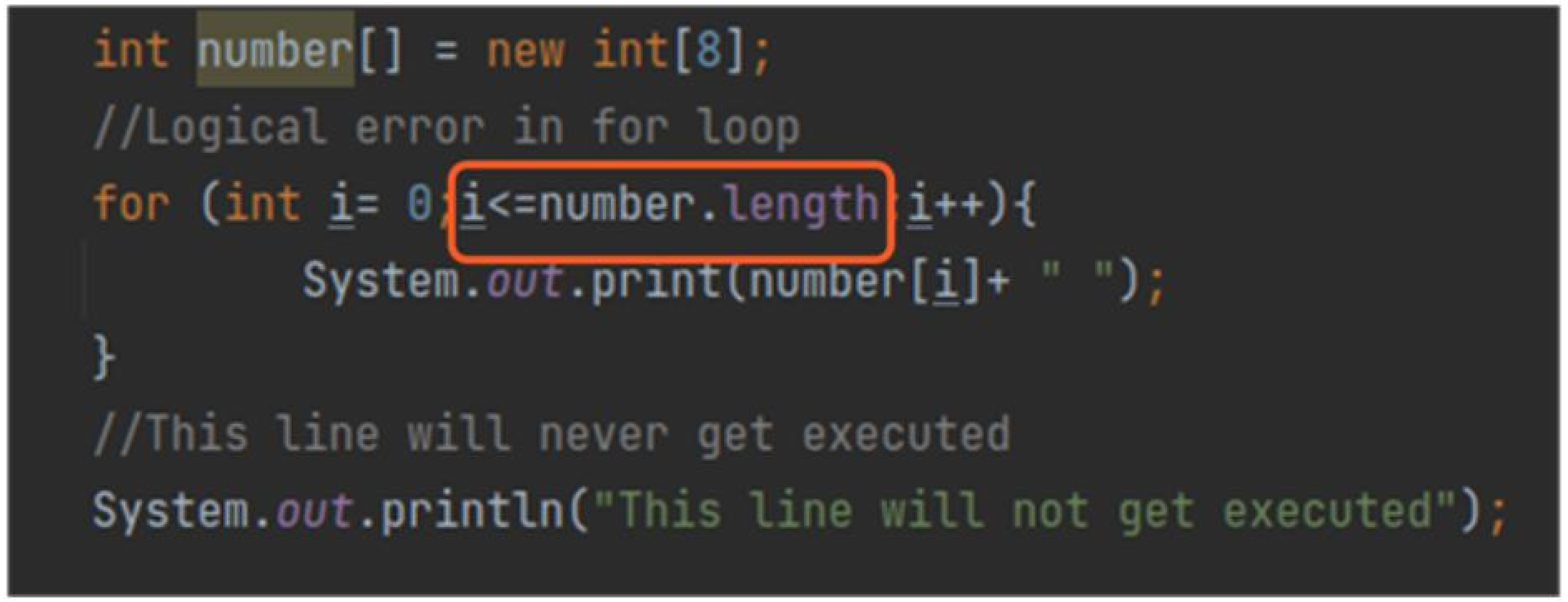
```
try{
    //code that can throw exception
}catch (Exception exception) {
    //Block of code to handle exception
}finally {
    //This line of code will always execute
}
```

Menu

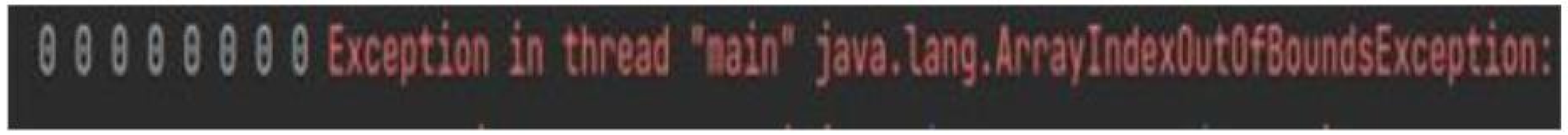
00:26 00:00:39 26/ 39

Slide Note

Code Without Exception Handling



```
int number[] = new int[8];
//Logical error in for loop
for (int i= 0;i<=number.length;i++){
        System.out.print(number[i]+ " ");
}
//This line will never get executed
System.out.println("This line will not get executed");
```



```
0 0 0 0 0 0 0 0 Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
```

- **When using the for loop to iterate over the number array,a logical error is made.**

`i <= number.length`

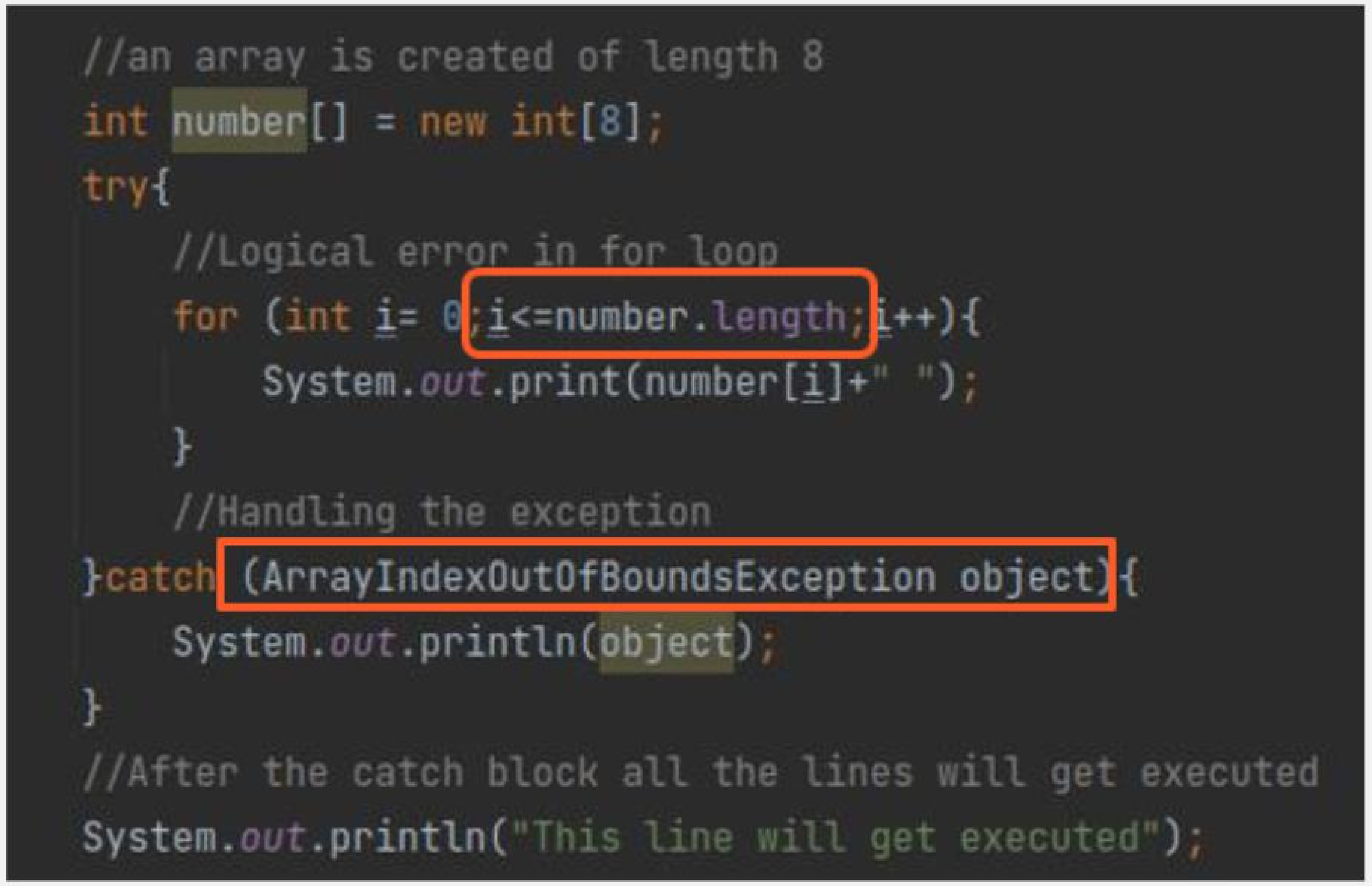
- **This will result in a runtime** `ArrayIndexOutOfBoundException`
- **Once the exception occurs, the rest of the code will not get executed, including the last line that is present outside the** `for` **loop.**

Menu

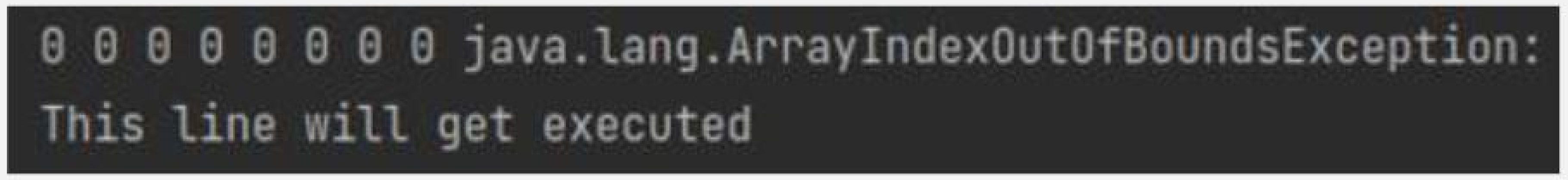
00:27 00:00:39 27/ 39

Slide Note

Code With Exception Handling



```
//an array is created of length 8
int number[] = new int[8];
try{
    //Logical error in for loop
    for (int i= 0;i<=number.length;i++){
        System.out.print(number[i]+" ");
    }
    //Handling the exception
}catch (ArrayIndexOutOfBoundsException object){
    System.out.println(object);
}
//After the catch block all the lines will get executed
System.out.println("This line will get executed");
```



```
0 0 0 0 0 0 0 0 java.lang.ArrayIndexOutOfBoundsException:
This line will get executed
```

- The code likely to generate the exception is written in the try block. Example: the for loop.
- Once the code executes, the try block will throw the `ArrayIndexOutOfBoundException.`
- The thrown exception i.e., `ArrayIndexOutOfBoundException`, is caught by the catch block.
- Since the exception is handled, the piece of code written after the catch block will execute, and thus the program will not be terminated abruptly.
- It is necessary to handle the same type of exception in the catch block generated in the try block.

Menu

ArrayIndexOutOfBoundException Demo

Write a Java program to create an array of ages.

Traverse through the array and print the age.

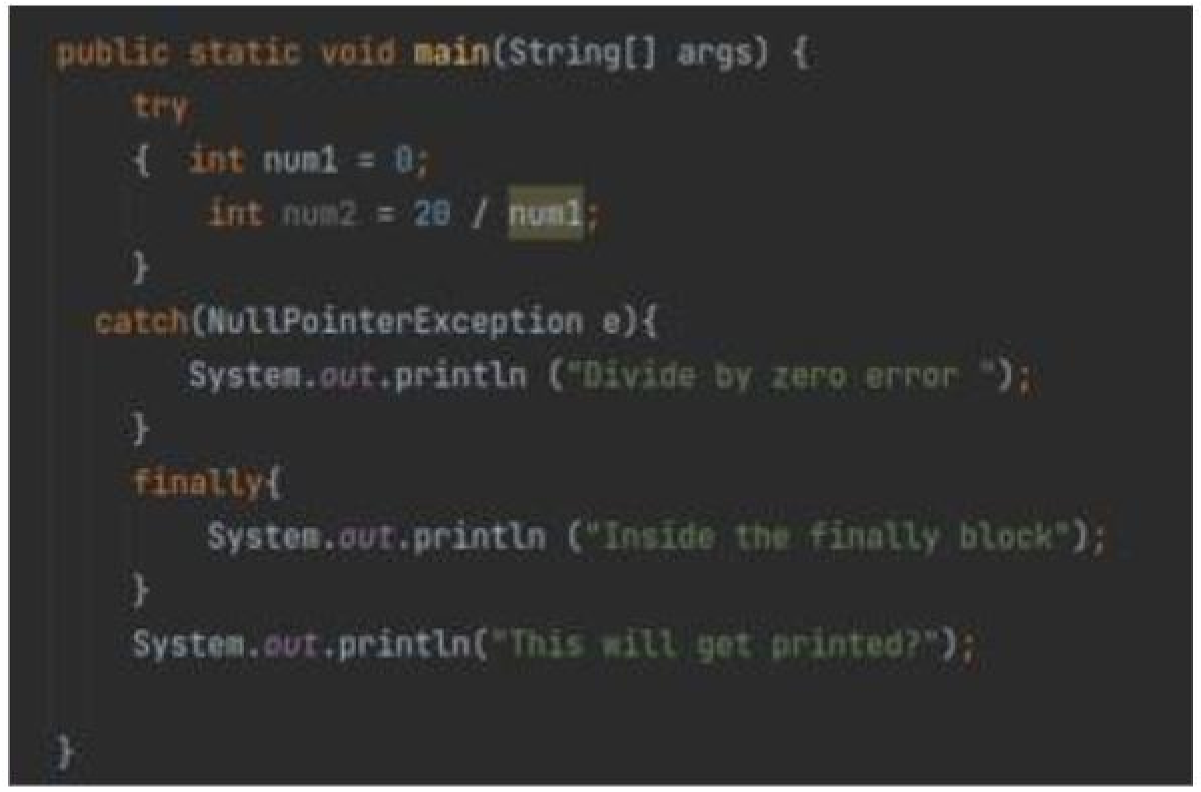
Handle the exception that can occur while traversing through an array when the length is not given correctly.

Click here for the solution.
The Intellij IDE must be used for the demonstration.

DEMO

Quick Check

Predict the output for the following code:



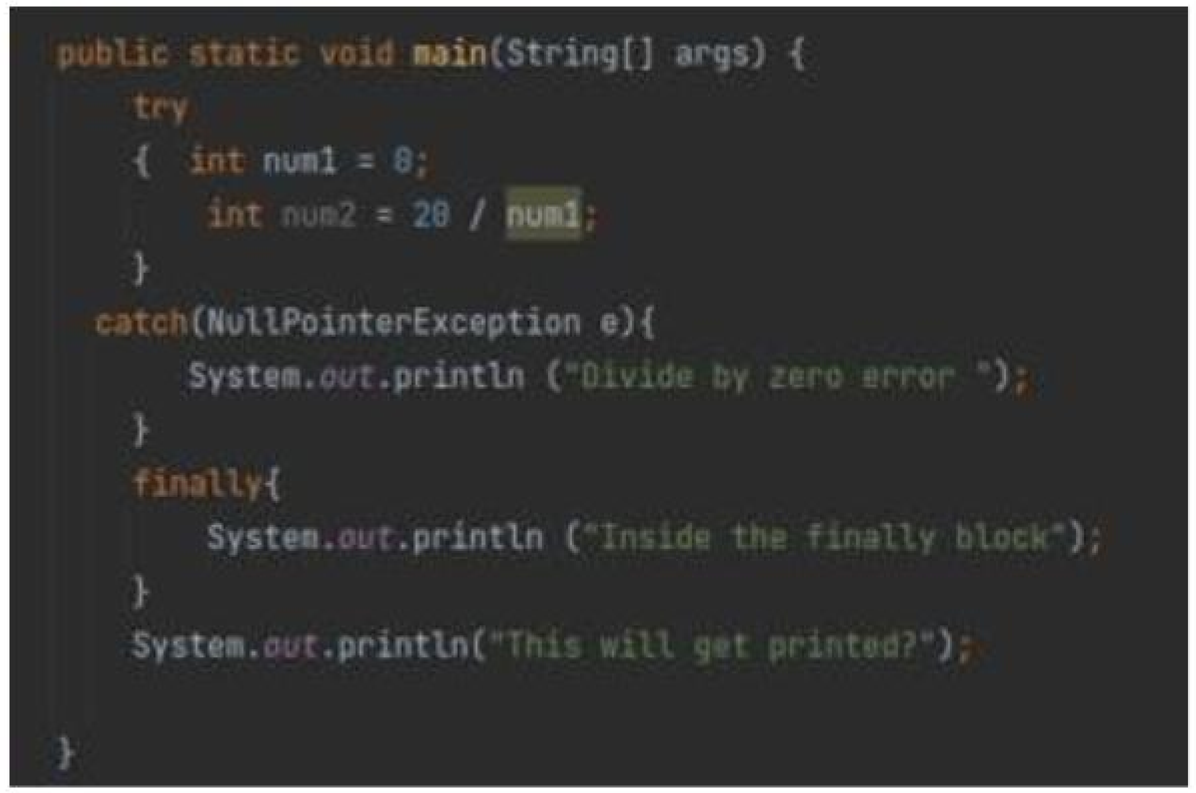
```
public static void main(String[] args) {
    try
    {  int num1 = 0;
        int num2 = 20 / num1;
    }
  catch(NullPointerException e){
       System.out.println ("Divide by zero error ");
    }
    finally{
        System.out.println ("Inside the finally block");
    }
    System.out.println("This will get printed?");

}
```

1. Compile time error
2. Divide by zero error
3. Divide by zero error
   Inside the finally block
   This will get printed?
4. Inside the finally block

Slide Note

Quick Check: Solution

Predict the output for the following code:



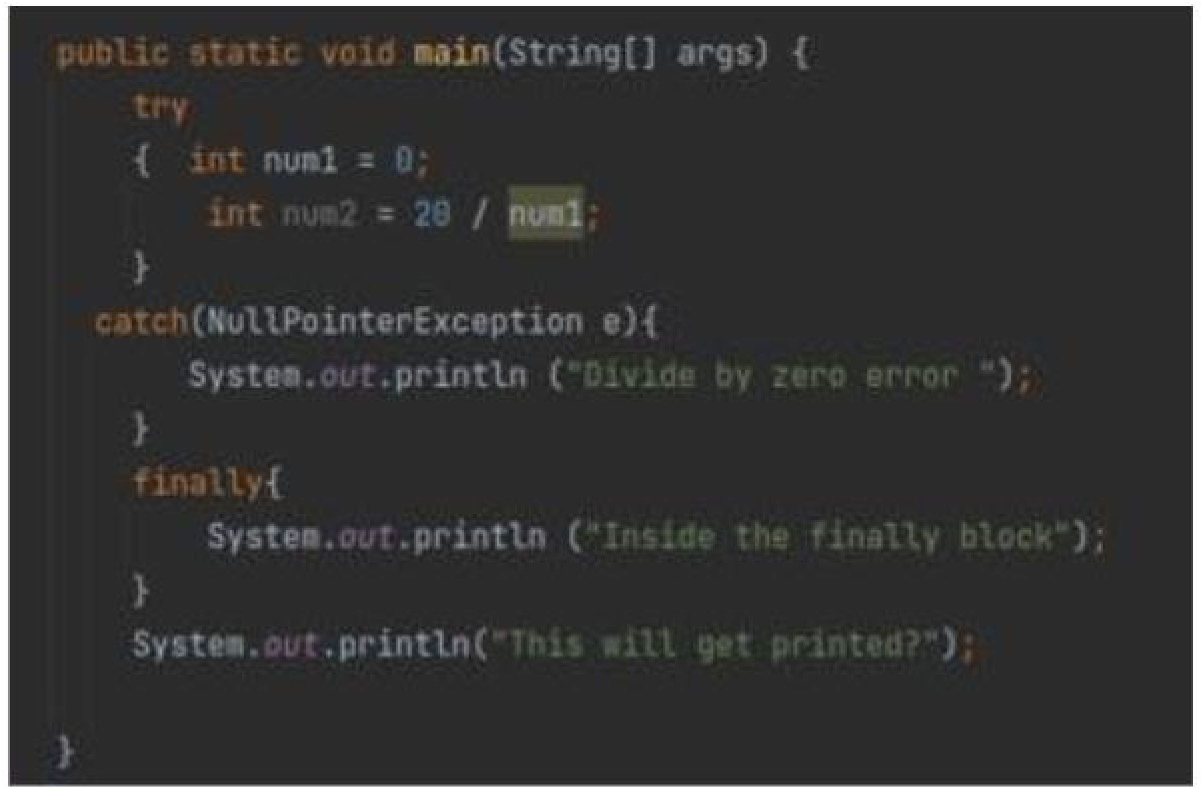
```java
public static void main(String[] args) {
    try
    {  int num1 = 0;
        int num2 = 20 / num1;
    }
  catch(NullPointerException e){
       System.out.println ("Divide by zero error ");
    }
    finally{
        System.out.println ("Inside the finally block");
    }
    System.out.println("This will get printed?");

}
```

1. Compile time error
2. Divide by zero error
3. Divide by zero error
   Inside the finally block
   This will get printed?
4. **Inside the finally block**

Menu

Quick Check

Predict the following output:



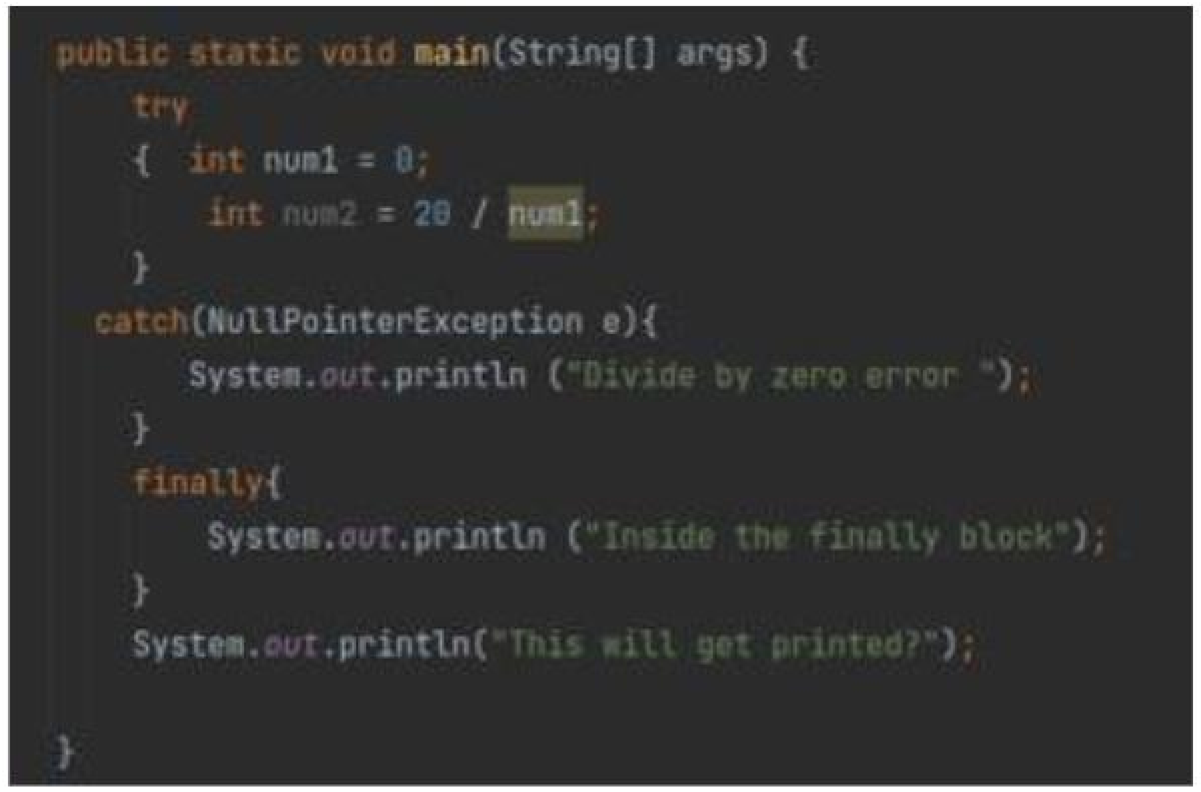
```
public static void main(String[] args) {
    try
    {  int num1 = 0;
        int num2 = 20 / num1;
    }
  catch(NullPointerException e){
       System.out.println ("Divide by zero error ");
    }
    finally{
        System.out.println ("Inside the finally block");
    }
    System.out.println("This will get printed?");

}
```

1. Compile time error
2. Inside the finally block
3. Inside the finally block
   Exception in thread "main" java.lang.ArithmeticException
4. Will this get printed?

Slide Note

Quick Check: Solution

Predict the following output:



```
public static void main(String[] args) {
    try
    {  int num1 = 0;
        int num2 = 20 / num1;
    }
  catch(NullPointerException e){
       System.out.println ("Divide by zero error ");
    }
    finally{
        System.out.println ("Inside the finally block");
    }
    System.out.println("This will get printed?");

}
```

1. Compile time error
2. Inside the finally block
3. **Inside the finally block**
   Exception in thread "main" java.lang.ArithmeticException
4. Will this get printed?

Menu

00:33 00:00:39 33/ 39

Slide Note

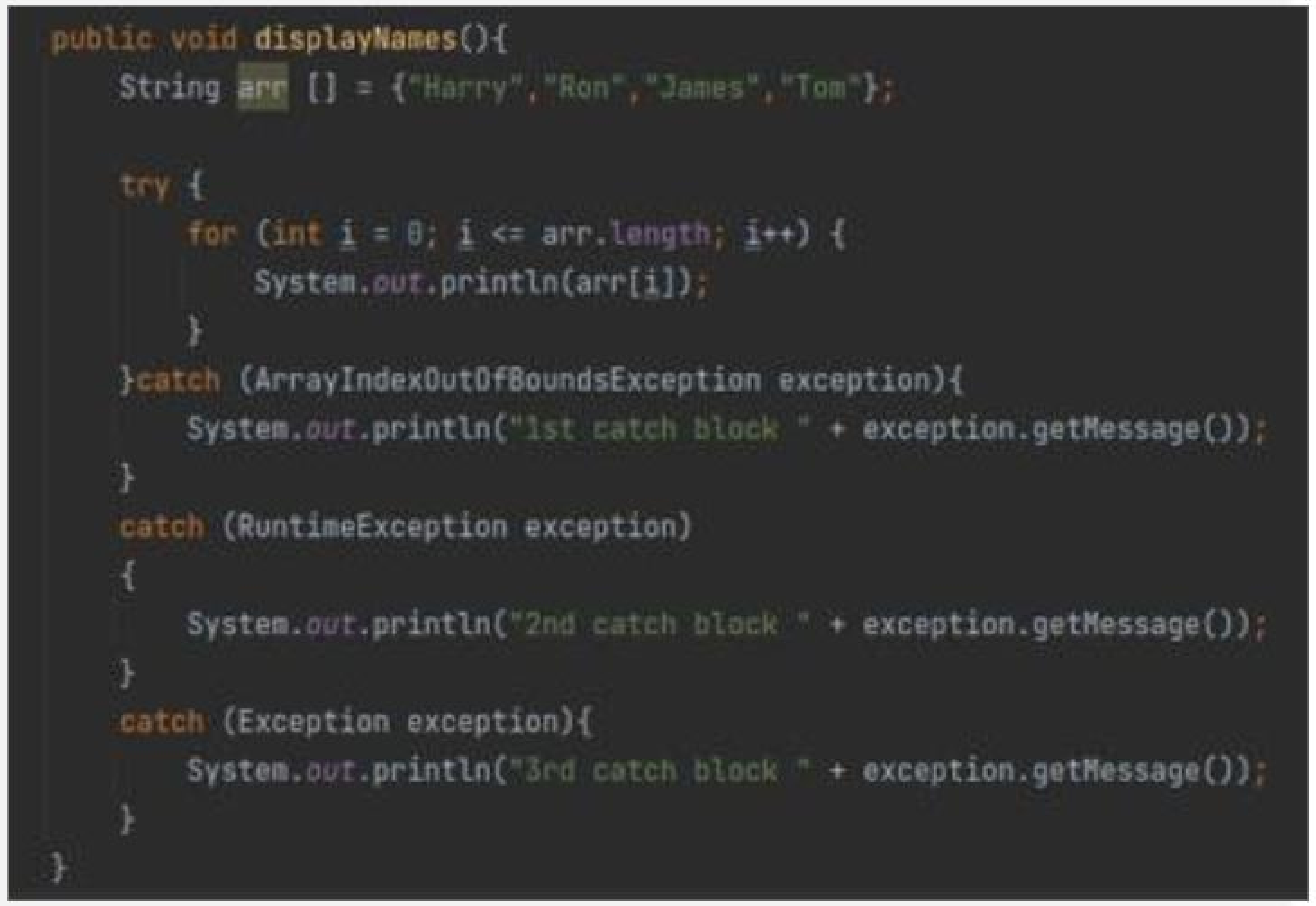
Explain Try With Multiple Catch

Menu

00:34:00:00:39 34/ 39

Slide Note

Try With Multiple Catch Block



```java
public void displayNames(){
    String arr [] = {"Harry","Ron","James","Tom"};

    try {
        for (int i = 0; i <= arr.length; i++) {
            System.out.println(arr[i]);
        }
    }catch (ArrayIndexOutOfBoundsException exception){
        System.out.println("1st catch block " + exception.getMessage());
    }
    catch (RuntimeException exception)
    {
        System.out.println("2nd catch block " + exception.getMessage());
    }
    catch (Exception exception){
        System.out.println("3rd catch block " + exception.getMessage());
    }
}
```

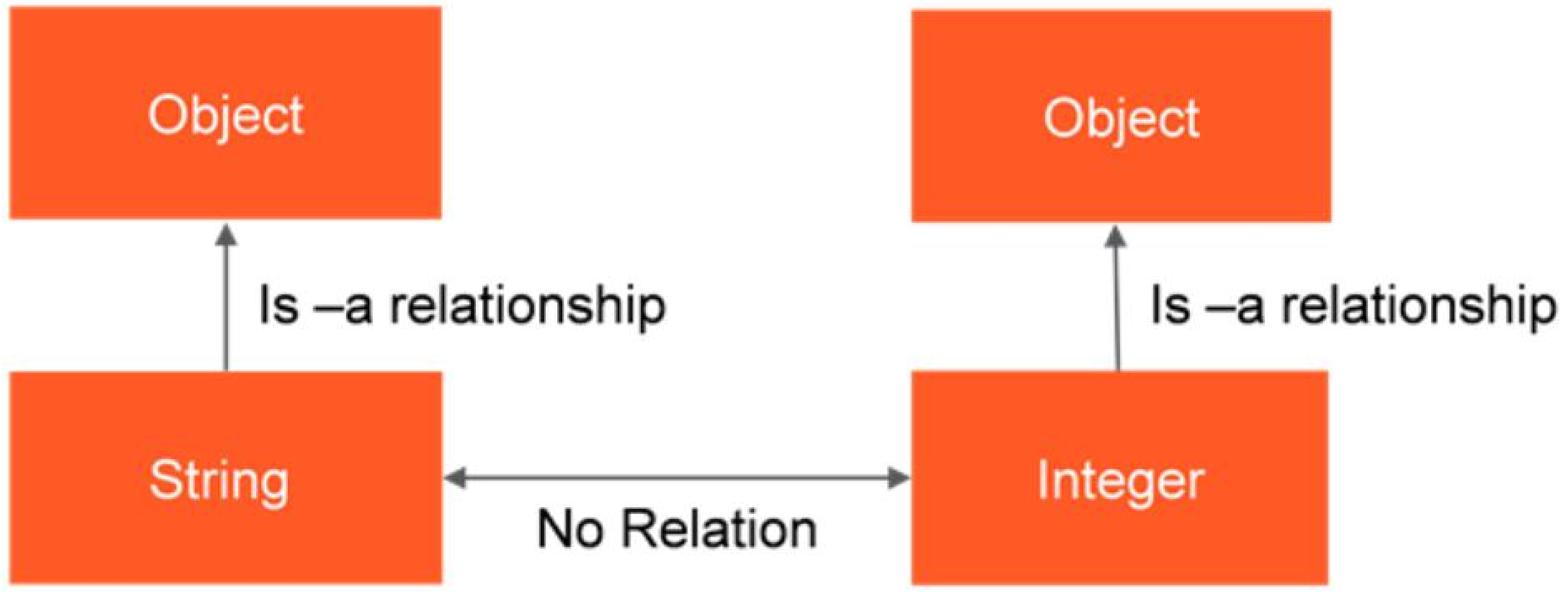
- **Multiple catch blocks can follow the try block.**
- **It is mandatory to follow the exception hierarchy while using multiple catch blocks.**
- **The parent class should be in the last catch block, and the child class should be in the preceding catch blocks.**
- **Since the exception class is the superclass of any runtime exception class, it is in the last catch block.**
- `RuntimeException` **is a child class of the exception class. Hence, it is in the catch block before the Exception.**
- `ArrayIndexOutOfBoundsException` **is a child class of** `RuntimeException.` **So, it is in the first catch block.**

Menu

00:35 00:00:39 35/ 39

ClassCastException

- ClassCastException is a runtime exception or unchecked exception that occurs when you try to convert or cast one class type object to another class type.
- This exception is thrown when you try to convert the objects of two individual classes that don't have any relationship between them.
  - For ex – Integer class to String Class
  - For ex – String class to Double class

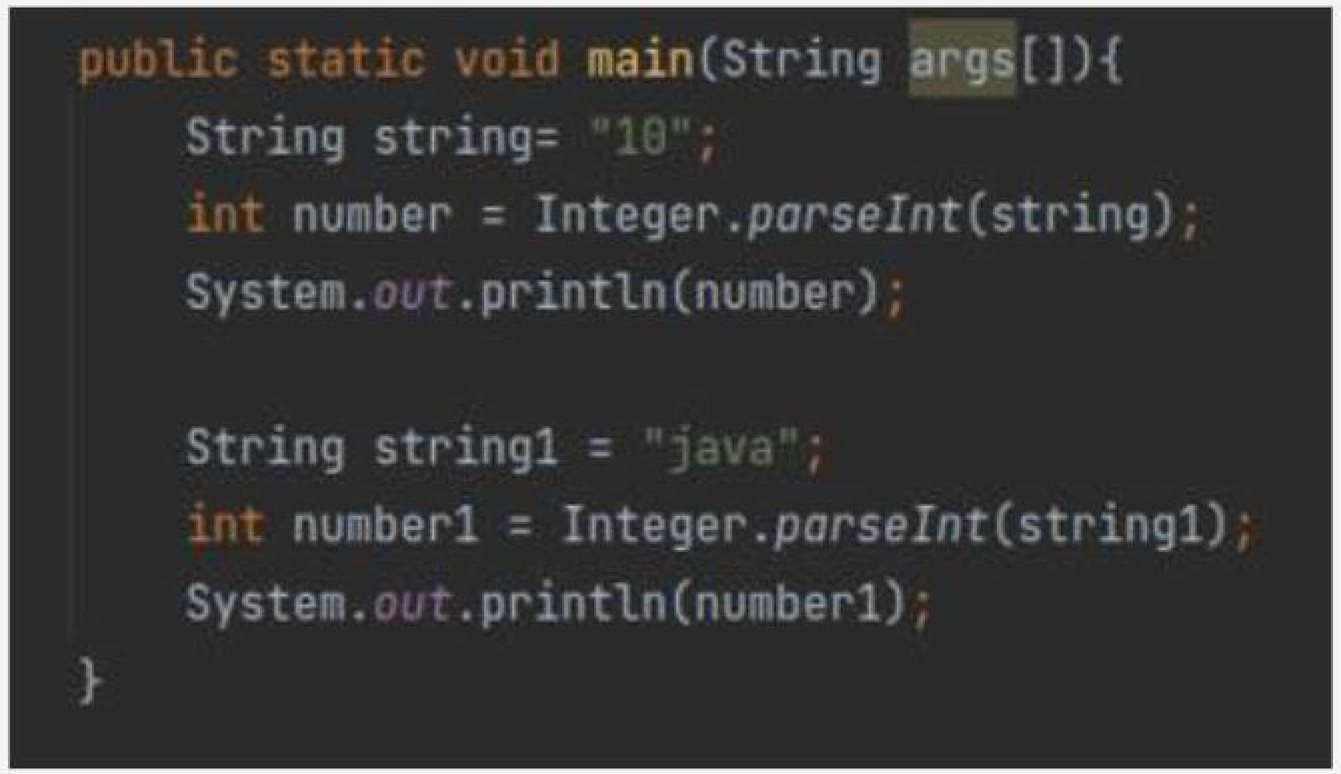


Slide Note

It is generally used if we have to perform mathematical operations on the string which contains a number. Whenever we receive data from TextField or TextArea, entered data is received as a string. If entered data is in number format, we need to convert the string to an int. To do so, we use Integer.parseInt() method.

Converting String to Primitive Type

- In certain programming scenarios, we will be required to work with raw data; even numbers can be in the form of a String like String rollNo = "101"
- In such cases, to perform some manipulation on the number data, we might have to convert text to numbers.
- Wrapper classes can be used to convert a String to a Number (int, float, double, etc.) type.
- To convert a String to an integer, the `parseInt` method is used.
- Parsing the String "10" will convert it to an `int` 10.
- Parsing the String "Java" will throw a `NumberFormatException`.



```
public static void main(String args[]){
    String string= "10";
    int number = Integer.parseInt(string);
    System.out.println(number);

    String string1 = "java";
    int number1 = Integer.parseInt(string1);
    System.out.println(number1);
}
```

Menu

00:37 / 00:00:39 37/ 39

Slide Note

NumberFormatException Demo

Write a Java program that converts a String to an Integer of the primitive type. Handle the appropriate exception that may occur while converting a String to an Integer type.

Click here for the solution.
The Intellij IDE must be used for the demonstration.

DEMO

Menu

00:38:00:00:39 38/ 39