
ORACLE
ACADEMY

Java Fundamentals

6-2
Handling Errors

ORACLE ACADEMY

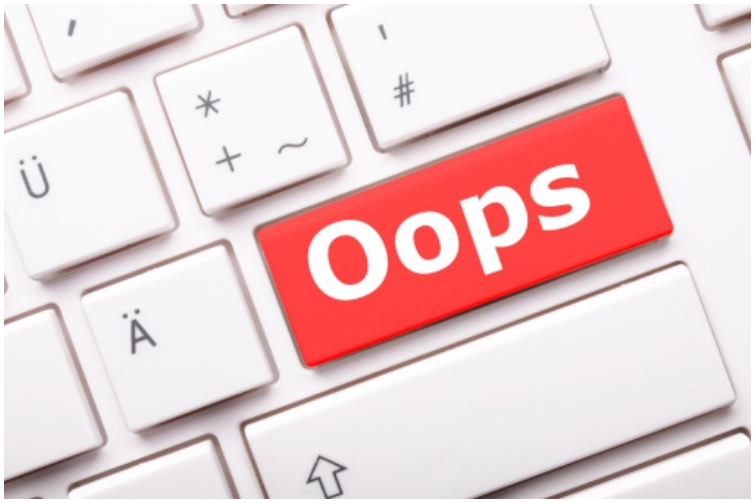
Objectives

This lesson covers the following objectives:

- Describe the different kinds of errors that can occur and how they are handled in Java
- Describe what exceptions are used for in Java
- Determine what exceptions are thrown for any foundation class
- Write code to handle an exception thrown by the method of a foundation class

Types of Errors

- An error indicates that a there is a problem with interpreting your program.
- There are three types of errors:
  - Syntax errors
  - Logic errors
  - Exceptions (run-time errors)



Syntax Errors

- An error can occur when a file is compiled. These errors are coding or syntax errors, such as:
  - Missing semicolons
  - Spelling errors
  - Assigning a value to a variable that is not the correct type
- It is common to go through several rounds of fixing syntax errors before compiling the file is successful.

Syntax Errors: Missing Semicolons and Incorrect Symbols

- Forgetting a semicolon at the end of a Java statement is a syntax error.

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Syntax error, insert ";" to complete Statement
    at ErrorIndicators.main(ErrorIndicators.java:8)
```

- Using = instead of == to compare values in an if condition is a syntax error.

```
if(x=y){
  …
      }
```

Incorrect

```
if(x==y){
  …
      }
```

Correct

Syntax Errors: Misspelling Methods or Variables

- Misspelling a variable or method name is a syntax error.
- Be sure to verify that you have spelled the variable or method the same as you declared it.
- To fix this error, check the spelling of:
  - The variable or method where it is declared.
  - Where you call the variable or method.

```
Exception in thread "main" java.lang.Error:
        Unresolved compilation problem:
       variableName cannot be resolved to a variable
      at MisspellingVariables.main(MisspellingVariables.java:7)
```

Logic Errors

- Logic errors occur as a result of programmer logic that is incorrect.
- These errors do not produce a compile or runtime error.
- For example, a loop runs too many times, or the program produces incorrect output.

Logic Errors

- Placing a semicolon after an if condition or initializing a loop:
  - Interpreters read the semicolon as the end of the loop, which means that everything after the semicolon will be treated as outside of the loop.

```
for(int i = 0; i < 5; i++);
        System.out.println(i);
```

This statement will only execute once. Why?

Logic Error Examples

- Using == to compare Strings or objects is a logic error unless you are checking to see if they refer to the same object.

```
String s1 = "Hello";
String s2 = "Goodbye";
if(s1==s2)
   System.out.println("They are equal");
```

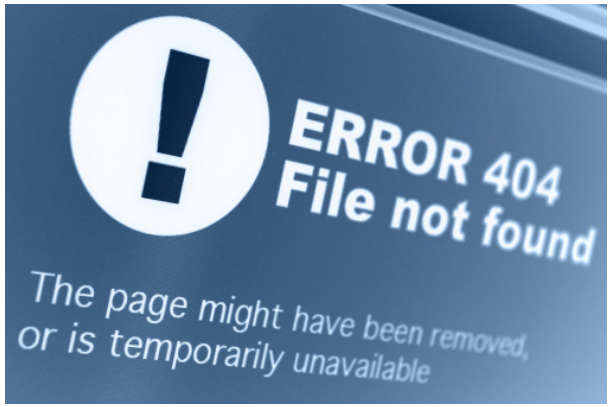
The == will check to see if these Strings are the same object. To check if the contents of the Strings are equal, use the equals() method from the String class.

- Because of the extra semicolon, "Hello" will always display, even if x is less than 10.

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

Exceptions

- Once a file compiles successfully, an error can occur when a file is tested during runtime.
- These runtime errors are called exceptions and should be handled by the programmer using code in the program, otherwise known as "throwing" the exception.



Categories of Java Exceptions

- Java exceptions fall into two categories:
  - Unchecked exceptions (errors in code).
  - Checked exceptions (errors that occur from outside the code), most of which originate in programs with Input/Output (IO).

Creating a Try/Catch Block

- A try/catch block can handle checked and unchecked exceptions. Below is an example of a try/catch block.

```
try{
        ...code that might cause an exception
}

catch (exception e){
        ...code to handle the exception
}
```

Unchecked Exceptions

- It is optional to handle unchecked exceptions in Java.
- However, if the unchecked exception is not handled, and an error occurs, the program will crash.
- Common unchecked exceptions:
  - Index out of bounds exception
  - Null pointer exception

Index Out of Bounds Exception

- Call example[3] in the following initialized array.

```
int[] example = {1, 2, 3};
```

- The following exception message would display, because example[3] does not exist in the array.

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3
   at NullPointerTest.main(NullPointerTest.java:6)
```

- The array starts at index 0 and ends at index 2. Calling index 3 means you are out of the bounds of the array.

```
example[0] = 1;
example[1] = 2;
example[2] = 3;
```

Null Pointer Exception

- The following code will throw a null pointer exception, because the String objects in the array have not been initialized.

```
String[] s1 = new String[5];
System.out.println(s1[0]);
```

- To correct the problem, assign String values to the 5 Strings in the array before trying to print out the values.

File Not Found Exception

- A File Not Found exception is an I/O (Input/Output) exception.
- I/O exceptions are checked exceptions.
- Most checked exceptions come from using I/O classes.

Handling Checked Exceptions

- Checked exceptions must be handled.
- There are two ways to handle a checked exception:
  - Use a try/catch block, which will handle the exception "gracefully."
  - Use a "throws" statement in the method declaration, which is a risk. A throws statement is when a programmer says that he/she will take a chance that an exception will not be thrown.

I/O Exception

- This code uses a try/catch block to handle an I/O exception.

```
try{
   FileReader reader = new FileReader("test.txt");
}
catch(IOException e){
   System.out.println("File not found");
}
```

File Not Found Exception

- Use a "throws" statement to handle an I/O Exception.
- The "throws" statement is used to warn that there may be an exception thrown, however, the program will still crash if an error occurs that throws an exception.

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

     FileReader reader = new FileReader("test.txt");
}
```

Throwing Exceptions

- So far you have seen exceptions handled by using a try/catch block.
- You can also handle exceptions by throwing them.
- If you throw an exception, your interpreter will stop running the program at that point, which indicates to the user that they have reached the exception.
- In code, an exception is thrown as follows:

```
throw new Exception("Array index" + i + " is out of bounds!");
```

Catching Exceptions

- To catch an exception means to handle it.
- You may throw an exception for certain cases, such as going out of bounds of an array, and catch the exception to continue the program the way you wrote it to handle the exception.
- A try/catch block enables you to do this.

Try/Catch Example

- This is an example of handling an Index Out Of Bounds exception with a try/catch block.

```
try{
    //i is the index of an array with length 10
    if(i > 9 || i < 0)
         throw new Exception("Index " + i + " is out of bounds!");
}
catch(Exception e){
    //This code will run only if the exception was thrown
    if(i > 9)
       i-=9;
    else
       i+=9;
}
//You may have additional code here that will run only if the exception was not thrown
```

Throwing and Catching Exceptions Scenario

- You are writing a program that moves a turtle to a part of the ocean that the user specifies.
- How would your program run if the user entered coordinates that were not on the map of the ocean where the turtle can move?
- This exception would be out of the programmer's control.
- Think about the throwing and catching an exception to solve this problem.

Terminology

Key terms used in this lesson included:

- Catch
- Checked exceptions
- Error
- Exception
- Logic error

Terminology

Key terms used in this lesson included:

- Runtime error
- Syntax error
- Throw
- Try/catch block
- Unchecked exceptions

Summary

In this lesson, you should have learned how to:

- Describe the different kinds of errors that can occur and how they are handled in Java
- Describe what exceptions are used for in Java
- Determine what exceptions are thrown for any foundation class
- Write code to handle an exception thrown by the method of a foundation class


ORACLE
ACADEMY