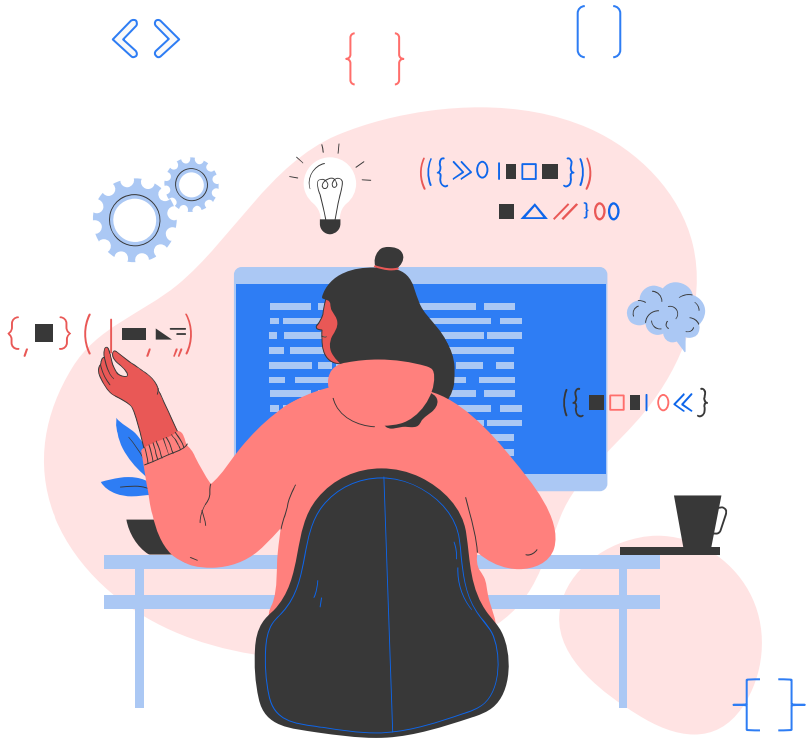
Exception Handling



Advanced World Systems, Inc.

JAVA TRAINING. FOR INTERNAL USE ONLY.

Objectives

01 Understand about Exception Handling.

02 Understand try, catch, finally, throw and throws syntax.

03 Design a Java program implementing Exception Handling.

Exception Handling

JAVA TRAINING. FOR INTERNAL USE ONLY.

Types of Error in Java

- **Run Time Error** occurs during execution of the program. The Java Virtual Machine will detect the errors while executing the program.
- **Compile Time Error** prevents the code from running because of an incorrect syntax such as a missing semicolon at the end of a statement or a missing bracket, class not found. etc. Also known as Syntax Errors.

Let’s identify! Identify what type of error is shown on the image below...

Run time error

```
int a = 6;

int b = a/0;
```

```
Console ×
<terminated> Main (3) [Java Application] C:\Program Files\Java\jre1.8.0_202\bin\javaw.exe (23 Nov 2023, 1:31:01 pm)
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at main.Main.main(Main.java:34)
```

Compile time error

```
System.out.println("The value of b is " + b)
```

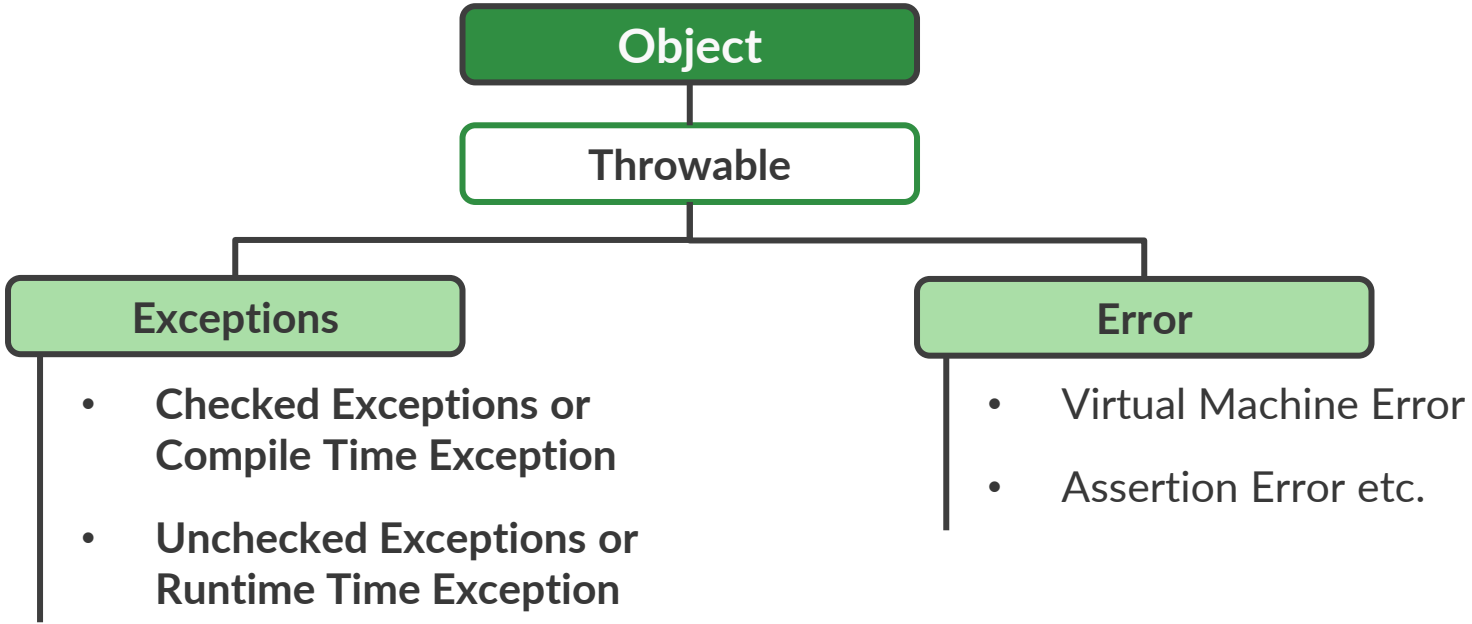
```
Console ×
<terminated> Main (3) [Java Application] C:\Program Files\Java\jre1.8.0_202\bin\javaw.exe (23 Nov 2023, 1:34:46 pm)
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        Syntax error, insert ";" to complete Statement

        at main.Main.main(Main.java:36)
```

Exception Handling

- In Java, an **exception** is an unwanted or unexpected event that disrupts the normal flow of the program.
- When an exception occurs within a method, it creates an object called **exception object**. It contains information about the exception, such as the name and description of the exception and the state of the program when exception occurred.
- Major reasons why exception occurs is because of invalid user input, device failure, loss of network connection, physical limitations (out-of-disk memory), code errors, and opening an unavailable file.
- **Exception Handling** is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.

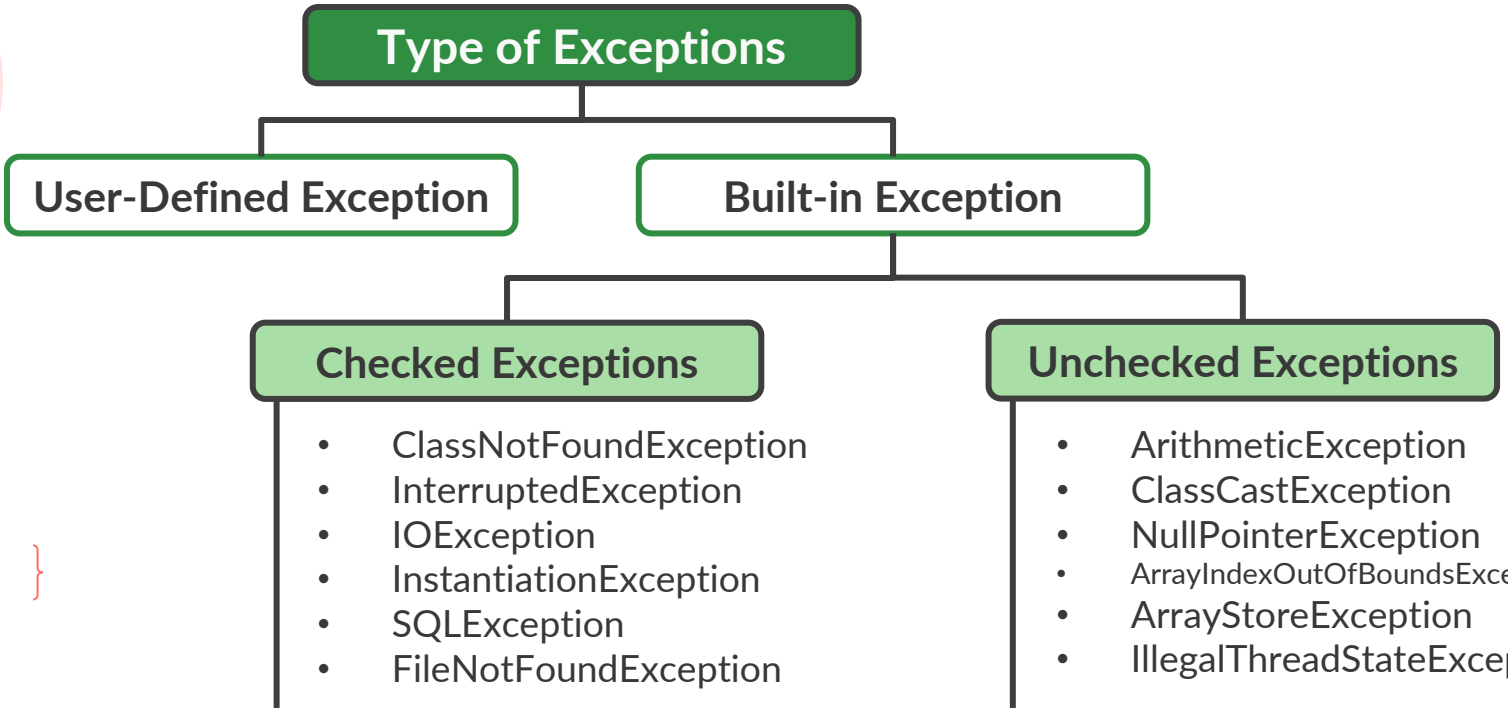
Exception Hierarchy



- **Checked Exceptions or Compile Time Exception**
- **Unchecked Exceptions or Runtime Time Exception**

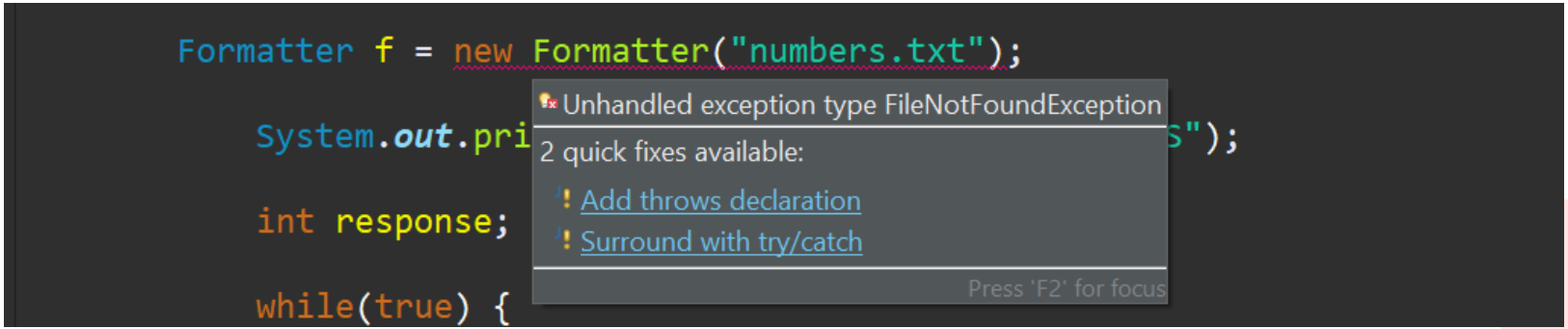
- Virtual Machine Error
- Assertion Error etc.

Exception Hierarchy



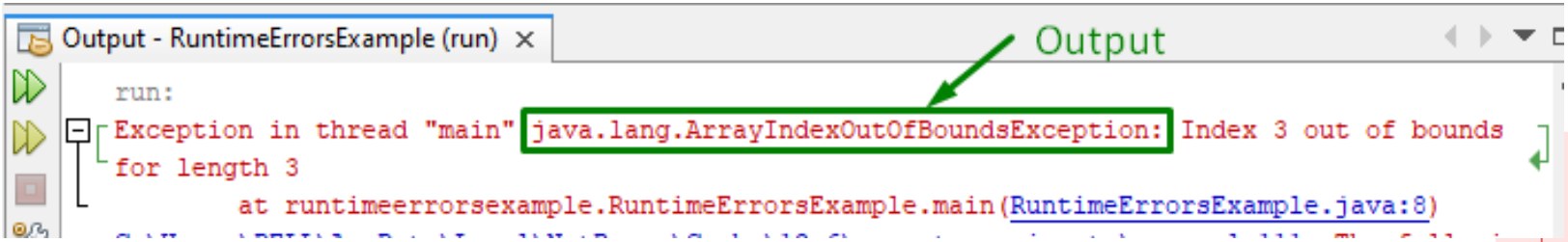
Checked Exception

- The classes that directly inherit the Throwable class except `RuntimeException` and `Error` are known as checked exceptions.
- For example, `IOException`, `SQLException`, etc. Checked exceptions are checked at compile-time.



Unchecked Exception

- The classes that inherit the `RuntimeException` are known as unchecked exceptions.
- For example, `ArithmeticException`, `NullPointerException`, `ArrayIndexOutOfBoundsException`, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.



Unchecked Exception

Exceptions	Description	Example
ArithmeticException	Thrown by the JVM when code attempts to divide by zero	int answer = 11 / 0;
ArrayIndexOutOfBoundsException	Thrown by the JVM when code uses an illegal index to access an array	int[] countsOfMoose = new int[3]; System.out.println(countsOfMoose[-1]);
ClassCastException	Thrown by the JVM when an attempt is made to cast an exception to a subclass of which it is not an instance	String type = "moose"; Object obj = type; Integer number = (Integer) obj;

IllegalArgumentException	Thrown by the programmer to indicate that a method has been passed an illegal or inappropriate argument	public void setNumberEggs
		(int numberEggs) {
		if (numberEggs >= 0) {
		this.numberEggs = numberEggs; }

NullPointerException	Thrown by the JVM when there is a null reference where an object is required	public void printLength () {
		System.out.println(name.length());
		}

NumberFormatException	Thrown by the programmer when an attempt is made to convert a string to a numeric type but the string doesn't have an appropriate format	Integer.parseInt("abc");

Checked Exception

Exceptions	Description
FileNotFoundException	Thrown programmatically when code tries to reference a file that does not exist.
IOException	Thrown programmatically when there's a problem reading or writing a file.

Error

Exceptions	Description
ExceptionInInitializerError	Thrown by the JVM when a static initializer throws an exception and doesn't handle it
StackOverflowError	Thrown by the JVM when a method calls itself too many times(this is called infi nite recursion because the method typically calls itself without end)
NoClassDefFoundError	Thrown by the JVM when a class that the code uses is available at compile time but not runtime.

Throw
Try
Catch

JAVA TRAINING. FOR INTERNAL USE ONLY.

Throw & Throws

throw new RuntimeException();
inside the method

method might have an exception so put **throws Exception** on method signature


```
void explore() {
    try {
        fall();
        System.out.println("never get here");
    } catch (Exception e) {
        getUp();
    }
    seeAnimals();
}
void fall() throws Exception{
    throw new RuntimeException()
}
```

Throw & Throws

Throw	Throws
Is used throw an exception explicitly in the code, inside the function or the block of code.	Is used in the method signature to declare an exception which might be thrown by the function while the execution of the code.
Using throw keyword, we can only propagate unchecked exception i.e., the checked exception cannot be propagated using throw only.	Using throws keyword, we can declare both checked and unchecked exceptions. However, the throws keyword can be used to propagate checked exceptions only.
We are allowed to throw only one exception at a time i.e. we cannot throw multiple exceptions.	We can declare multiple exceptions using throws keyword that can be thrown by the method. For example, main() throws IOException, SQLException.

Types of Exceptions

Type	How to Recognize	Okay for program to catch?	Is program required to handle or declare?
Runtime	Subclass of RuntimeException	Yes	No
Checked	Subclass of Exception but not RuntimeException	Yes	Yes
Error	Subclass of Error	No	No

Try, Catch, and Finally

```
void explore() {
    try {
        fall();
        System.out.println("never get here");
    } catch (Exception e) {
        getUp();
    }
    finally {
        seeMoreAnimals();
    }
    goHome();
}
void fall() throws Exception{
    throw new RuntimeException();
}
```

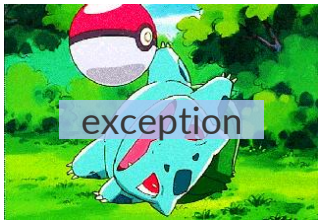
try block allows you to define a block of code to be tested for errors while it is being executed.

catch block allows you to define a block of code to be executed, if an error occurs in the try block.

finally block always executes, whether or not an exception occurs in the try block

Try to execute the code with possible exception.

When exception is detected it will be caught.



Let’s try! Try the try-catch syntax on the given code snippet...

```
1 public class TryCatchExample {
2
3  public static void main(String[] args) {
4
5    int data=50/0;
6
7    System.out.println("rest of the code");
8  }
9 }
```

Let's try! Try the try-catch syntax on the given code snippet...

```
1 public class TryCatchExample {
2
3  public static void main(String[] args) {
4
5    int arr[]= {1,3,5,7};
6    System.out.println(arr[10]);
8  }
9 }
```

Let’s answer! What is the output of the code snippet below?

```
1 try
2  fall();
3 catch (Exception e)
4  System.out.println("get up");
```

Does not compile as curly braces are missing on try-catch

Let’s answer! What is the output of the code snippet below?

```
1 try {
2  fall();
3 }
```

Does not compile as try doesn’t have anything after. You may put catch or finally or both of them.

Let’s answer! What is the output of the code snippet below?

```
1 try {
2  fall();
3 } finally {
4    System.out.println(“all better”);
5 } catch (Exception e) {
6    System.out.println("get up");
7 }
```

Does not compile as catch and finally blocks are in the wrong order.

Let's evaluate! What is the output of the code snippet below
(1) if throw w/o exception, (2) if animal is out for walk,
(3) if exhibit is closed?

```
class AnimalsOutForAWalk extends RuntimeException { }
class ExhibitClosed extends RuntimeException { }
class ExhibitClosedForLunch extends ExhibitClosed { }
```

```
1 public void visitPorcupine() {
2  try {
3   seeAnimal();
4  } catch (AnimalsOutForAWalk e) {
5     System.out.print("try back later");
6  } catch (ExhibitClosed e) {
7     System.out.print("not today");
8  }
9 }
```

Order of the catch blocks could be reversed because the exceptions don't inherit from each other.

(1) prints nothing

(2) first catch block runs

(3) second catch block runs

Let's evaluate! What is the output of the code snippet below (1) If exhibit closed for lunch is thrown and (2) if exhibit is closed?

```
class AnimalsOutForAWalk extends RuntimeException { }
class ExhibitClosed extends RuntimeException { }
class ExhibitClosedForLunch extends ExhibitClosed { }
```

```
1 public void visitMonkeys() {
2  try {
3   seeAnimal();
4  } catch (ExhibitClosedForLunch e) {
5     System.out.print("try back later");
6  } catch (ExhibitClosed e) {
7     System.out.print("not today");
8  }
9 }
```

(1) first catch block runs

(2) second catch block runs

Let's evaluate! What is the output of the code snippet below (1) if line 4 throws a NullPointerException and (2) if line 4 throws a IOException?

```
1 public static void main(String[] args) {
2  FileReader reader = null;
3  try {
4   reader = read();
5  } catch (IOException e) {
6     try {
7       if (reader != null) reader.close();
8     } catch (IOException inner) { }
9  }
10}
11 private static FileReader read()throws IOException {
12 }
```

(1) skips line 5 – 9 and throws the exception.

(2) Enters line 5 - 7

Let’s evaluate! What is the output of the code snippet below?

```
1 try {
2  throw new RuntimeException();
3 } catch (RuntimeException e) {
4    throw new RuntimeException();
5 } finally {
6    throw new Exception();
7 }
```

Line 6 is the latest Exception thrown but not caught, as for line 4 it is forgotten due to the finally block being executed.

Let's evaluate! What is the output of the code snippet below?

```
1 public String exceptions() {
2   String result = "";
3   String v = null;
4   try {
5     try {
6       result += "before";
7       v.length();
8       result += "after";
9     } catch (NullPointerException e) {
10      result += "catch";
11      throw new RuntimeException();
12    } finally {
13      result += "finally";
14      throw new Exception();
15    }
16  } catch (Exception e) { result += "done"; }
17  return result;
18 }
```

Let’s evaluate! What is the output of the code snippet below?

```
class CanNotHopException extends Exception { }
```

```
1 class Hopper {
2  public void hop() throws CanNotHopException { }
3 }
4 class Bunny extends Hopper {
5  public void hop() { }
6 }
```

Does compile as a subclass is allowed to declare fewer exceptions than the superclass or interface.

Let's evaluate! What is the output of the code snippet below?

```
1 class Hopper {
2  public void hop() throws Exception { }
3 }
4 class Bunny extends Hopper {
5  public void hop() throws CanNotHopException { }
6 }
```

Does compile as Bunny could declare that it throws Exception directly, or it could declare that it throws a more specific type of Exception. It could even declare that it throws nothing at all.

Let’s evaluate! What is the output of the code snippet below?

```
1 public static void main(String[] args) {
2  try {
3   hop();
4  } catch (Exception e) {
5     System.out.println(e);
6     System.out.println(e.getMessage());
7     e.printStackTrace();
8  }
9 }
10 private static void hop() {
11  throw new RuntimeException("cannot hop");
12 }
```

```
java.lang.RuntimeException: cannot hop
cannot hop
java.lang.RuntimeException: cannot hop at
trycatch.Handling.hop(Handling.java:15)
at trycatch.Handling.main(Handling.java:7)
```

What questions do you have?

References

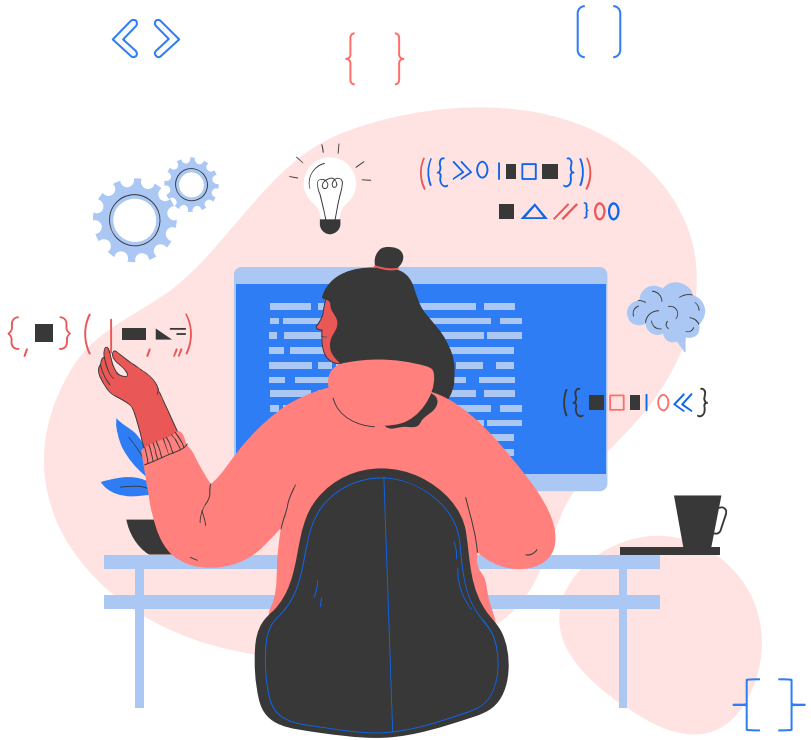
[1] GeeksforGeeks. (2022, June 8). Types of Errors in Java with Examples. GeeksforGeeks. https://www.geeksforgeeks.org/types-of-errors-in-java-with-examples/

[2] GeeksforGeeks. (2023b, August 7). Exceptions in Java. GeeksforGeeks. https://www.geeksforgeeks.org/exceptions-in-java/

[3] Java throw exception - javatpoint. (n.d.). www.javatpoint.com. https://www.javatpoint.com/throw-keyword

[4] Java Throws Keyword - javatpoint. (n.d.). www.javatpoint.com. https://www.javatpoint.com/throws-keyword-and-difference-between-throw-and-throws

Exception Handling



Advanced World Systems, Inc.

JAVA TRAINING. FOR INTERNAL USE ONLY.