


Course 4 - Sprint 1: Implement Exception Handling

Learning Consolidation

1. What is the difference between error and exception in java?
 - a. Errors are mainly caused by the environment in which an application is running. For example, OutOfMemoryError happens when JVM runs out of memory. Where as exceptions are mainly caused by the application itself. For example, NullPointerException occurs when an application tries to access null object. NumberFormatException occurs when Parse of String to Integer is done.
2. When you get unreachable catch block error?
 - a. When you are keeping multiple catch blocks, the order of catch blocks must be from most specific to most general ones. i.e sub classes of Exception must come first and super classes later. If you keep super classes first and sub classes later, compiler will show unreachable catch block error.

```
public class ExceptionHandlingDemo
{
    public static void main(String[] args)
    {
        try
        {
            int i = Integer.parseInt("John");
            //This statement throwNumberFormatException
        }

        catch(Exception ex)
        {
            
            System.out.println("This is super class
            of all Exception and it handles all
            exception types");
        }

        catch(NumberFormatException ex)
        {
            //Compile time error
            //This block becomes unreachable as
            //exception is already caught by above
            catch block
        }
    }
}
```

3. Checked and Unchecked Exception in Java

- a. **Checked Exceptions** are exceptional scenarios that we can anticipate in a program and try to recover from it, for example, `FileNotFoundException/IOException/SQLException`. We should catch this exception and provide a useful message to the user and log it properly in Loggers for debugging purposes. Exception is the parent class of all Checked Exceptions.
- b. **Runtime Exceptions** are caused by bad programming, for example, trying to retrieve an element from the Array. We should check the length of the array first before trying to retrieve the element otherwise it might throw `ArrayIndexOutOfBoundsException` at runtime. `RuntimeException` is the parent class of all runtime exceptions.

4. What are the important methods of Java Exception Class?

- a. Exception and all its subclasses don't provide any specific methods and all of the methods are defined in the base class `Throwable`.
- b. **`String getMessage()`**: This method returns the message `String` of `Throwable` and the message can be provided while creating the exception through its constructor.
- c. **`String getLocalizedMessage()`**: This method is provided so that subclasses can override it to provide the locale-specific messages to the calling program. `Throwable` class implementation of this method simply use `getMessage()` method to return the exception message.

- d. **synchronized Throwable getCause()**: This method returns the cause of the exception or null if the cause is unknown.
 - e. **String toString()**: This method returns the information about Throwable in String format, the returned String contains the name of Throwable class and localized message.
 - f. **void printStackTrace()**: This method prints the stack trace information to the standard error stream, this method is overloaded and we can pass `PrintStream` or `PrintWriter` as an argument to write the stack trace information to the file or stream.
5. What happens when an exception is thrown by the main method of the class?
- a. When an exception is thrown by a `main()` method, Java Runtime terminates the program and prints the exception message and stack trace in the system console.
6. What will be the output of the below Java program?

```
class ExceptionDemo {  
    public static void main(String args[]) {  
        try {  
            System.out.print("Hello" + " " + 10 / 0  
);  
        } catch (ArithmeticException arithmeticexce  
ption) {  
            System.out.print("World");  
        }  
    }  
}
```

7. Following are the advantages of using exceptions:

- a. Separating Error-handling code from "regular" code.
- b. Propagating errors up the call stack.
- c. Grouping and differentiating error types.

Further Read - <https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html>