A close-up of logos

Description automatically generated

**BITP 3113**

**OBJECT ORIENTED PROGRAMMING**

**LAB:**

**LAB 10 - EXCEPTION**

**LECTURER NAME:**

**DR EMALIANA BINTI KASMURI**

**STUDENT NAME:**

**MUHAMMAD AFIQ MUHAIMIN BIN MOHD ZAINI**

**MATRIC NUMBER:**

**B032410648**

**SEMESTER: 2-2024/25**

# Learning Outcomes

At the end of this lab exercise, the student should be able to:-

1. Understand the concept of exceptions in Java and their role in handling runtime errors.
2. Apply try-catch blocks to handle exceptions and prevent abrupt program termination.
3. Interpret Java API documentation to identify potential exceptions thrown by methods and constructors.
4. Analyze and differentiate between various exception types in Java.
5. Develop Java programs that demonstrate proper exception handling techniques for improved robustness and reliability.

# Tools

The exercise for this lab session is using the following tools:-

1. Eclipse for Java Developer preferably Version: 2025-06 M1 (4.36.0 M1)

# Supporting Materials

The reference files and supporting materials are available on the ulearn course site.

# Understanding Exception

In Java, an **exception** is an event that disrupts the normal flow of the program's execution. It's essentially an error condition that occurs during runtime. When an exceptional situation arises, Java creates an **exception object** containing information about the error.

Java provides a mechanism called **exception handling** to gracefully manage these runtime errors, preventing the program from abruptly terminating. This is typically done using try-catch blocks. The code that might throw an exception is placed within the try block, and the code that handles the exception is placed within the catch block.

# Exercise 1

1. Download the class **labexception-exerciseone.**zip from ulearn.
2. Execute the DataManipulatorApp class.
3. Observe the output.
4. Then, execute the SecureDataManipulatorApp class.
5. Observe the output from this class too.
6. Compare the output from both executions.

Ǫuestion: Which execution ends gracefully? State your reason(s). Record your answer in ulearn.

# Exercise 3

Based on the knowledge and skill comprehended from the previous exercises, enhanced the solution for ExceptionDemoOne.java, ExceptionDemoTwo.java and ExceptionDemoThree.java for the application to end unabruptly.

Then, submit the solutions in ulearn.

# Exception in Java API Documentation

The API documentation for Java methods explicitly tells which exceptions a method or constructor might throw. This is a crucial piece of information for any developer using that method or constructor, as it dictates how they should handle potential

errors. Typically there are two clues that provides this information - method signature and throws section.



## The Method Signature

Consider the FileInputStream API documentation. The first constructor signature clearly provides the first clue. After the parameter list, often the API shall state the throws keyword followed by a comma-separated list of exception classes. [Figure 1](#_bookmark9) the anatomy of the first constructor signature from the FileInputStream API documentation.

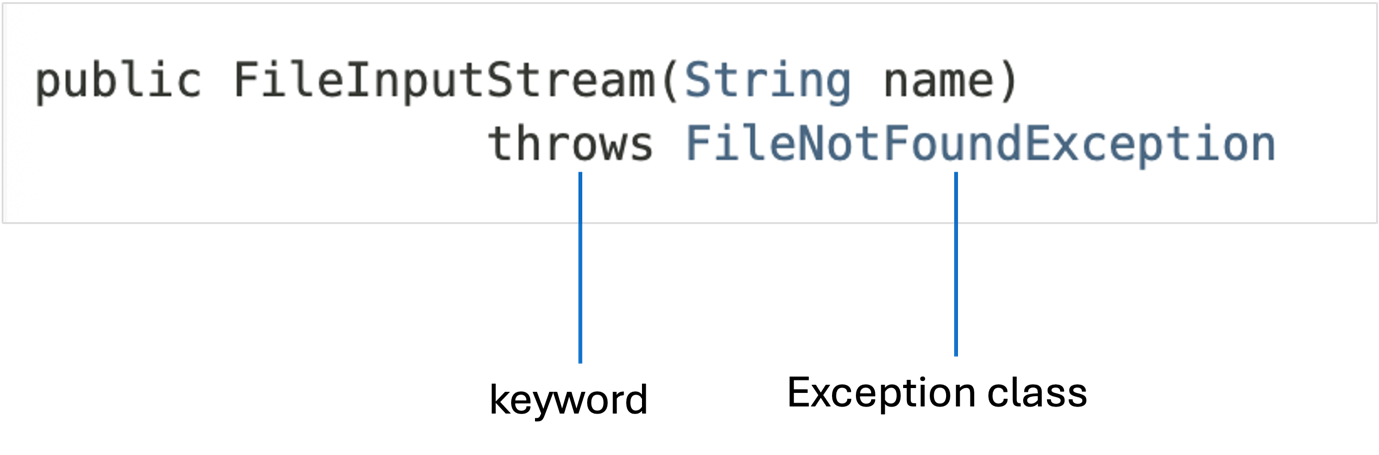


Figure 1: Anatomy of constructor that throws an exception

The throws FileNotFoundException part clearly indicates that this constructor can throw a FileNotFoundException if the file specified by the name does not exist.

## The Throws Section

Most well-written API documentation will have a dedicated **Throws** section specifically outlining the exceptions that the method can throw and under what conditions. This section provides more detail than just the method signature.

Looking at the documentation for FileInputStream:

### Throws:

FileNotFoundException - if the file does not exist, is a directory rather than a regular file, or for some other reason cannot be opened for reading.

SecurityException - if a security manager exists and its checkRead method denies read access to the file.

This **Throws** section elaborates on the specific reasons why

a FileNotFoundException or a SecurityException might be thrown. This level of detail is invaluable for understanding how to use the method correctly and handle potential issues.

A careful study these parts of the API documentation, will help the developer to anticipate potential exceptions and write codes to handle them gracefully using try- catch blocks, ensuring the robustness and reliability of the application.

# Exercise 4

1. Open the **FileReader** API documentation from this link.
2. Identify the exception thrown from the constructors.
3. Answer the following questions
   1. What is the name of the exceptions?
   2. How many exceptions can a constructor throw?

# Exercise 5

1. Open the **Socket** API documentation from this link.
2. Identify the exception thrown from:-
   1. The first constructor.
   2. The third constructor.
   3. The fourth constructor.
   4. The bind( ) method.
   5. The getOutputStream( ) method.
   6. The setSendBufferSize( ) method.
3. For each of the exception, analyze and record the analysis result in the following table.

|  |  |  |
| --- | --- | --- |
| **Constructor / Method** | **Number of exceptions to be thrown** | **Exception Name** |
|  |  |  |
|  |  |  |

# Exercise 6

Write a Java class to create a **FileReader** object using the first constructor as stated in this API documentation link to demonstrate your comprehension on exception handling.

Submit the solution in ulearn.

### End of Document