Java Exceptions

An exception is an unexpected event that occurs during program execution. It affects the flow of the program instructions which can cause the program to terminate abnormally.

An exception can occur for many reasons. Some of them are:

- Invalid user input
- Device failure
- Loss of network connection
- Physical limitations (out of disk memory)
- Code errors
- Opening an unavailable file

Java Runtime Exceptions

A runtime exception happens due to a programming error. They are also known as unchecked exceptions.

These exceptions are not checked at compile-time but run-time. Some of the common runtime exceptions are:

- Null pointer access (missing the initialization of a variable) - NullPointerException
- Out-of-bounds array access - ArrayIndexOutOfBoundsException
- Dividing a number by 0 - ArithmeticException

You can think about it in this way. "If it is a runtime exception, it is your fault".

Java IOException Exceptions

An IOException is also known as a checked exception. They are checked by the compiler at the compile-time and the programmer is prompted to handle these exceptions.

Some of the examples of checked exceptions are:

- Trying to open a file that doesn't exist results in FileNotFoundException
- Trying to read past the end of a file

Main.java

→ javac Main.Java

→ j

try-catch block

The try...catch block in Java is used to handle exceptions and prevents the abnormal termination of the program.

```
try{
  // code
}
catch(exception) {
  // code
}
```

try-catch-finally block

In Java, we can also use the finally block after the try...catch block.
In this case, the finally block is always executed whether there is an exception inside the try block or not.
It is a good practice to use finally block to include important cleanup code like closing a file or connection.

Note: There are some cases when a finally block does not execute:

- Use of System.exit() method
- An exception occurs in the finally block
- The death of a thread

try-catch-finally block

In Java, we can also use the finally block after the try...catch block.
In this case, the finally block is always executed whether there is an exception inside the try block or not.
It is a good practice to use finally block to include important cleanup code like closing a file or connection.

Note: There are some cases when a finally block does not execute:

- Use of System.exit() method
- An exception occurs in the finally block
- The death of a thread