Object Oriented Programming with Java (Subject Code: BCS-403)

Unit 2

Lecture 13

Lecture 13

- **The Idea behind Exception**
- **Exceptions & Errors**
- **Types of Exception, Checked and Un-Checked Exceptions**

Exception Handling

Exception Handling in Java is one of the effective means to handle runtime errors so that the regular flow of the application can be preserved.

Java Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

Control Flow in Exceptions

The core advantage of exception handling is to maintain the normal flow of the application.

An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions.

Let's consider a scenario

```
statement 1;
statement 2;
statement 3;
statement 4;
statement 5;//exception occurs
statement 6;
statement 7;
statement 8;
statement 9;
statement 10;
```

Suppose there are 10 statements in a Java program and an exception occurs at statement 5; the rest of the code will not be executed, i.e., statements 6 to 10 will not be executed. However, when we perform exception handling, the rest of the statements will be executed. That is why we use exception handling in Java.

The Idea behind Exception

- Exception is an unwanted or unexpected event, which occurs during the execution of a program, i.e. at run time, that disrupts the normal flow of the program's instructions.
- Exceptions can be caught and handled by the program.
- When an exception occurs within a method, it creates an object.

- This object is called the exception object.
- It contains information about the exception, such as the name and description of the exception and the state of the program when the exception occurred.

Major reasons why an exception Occurs

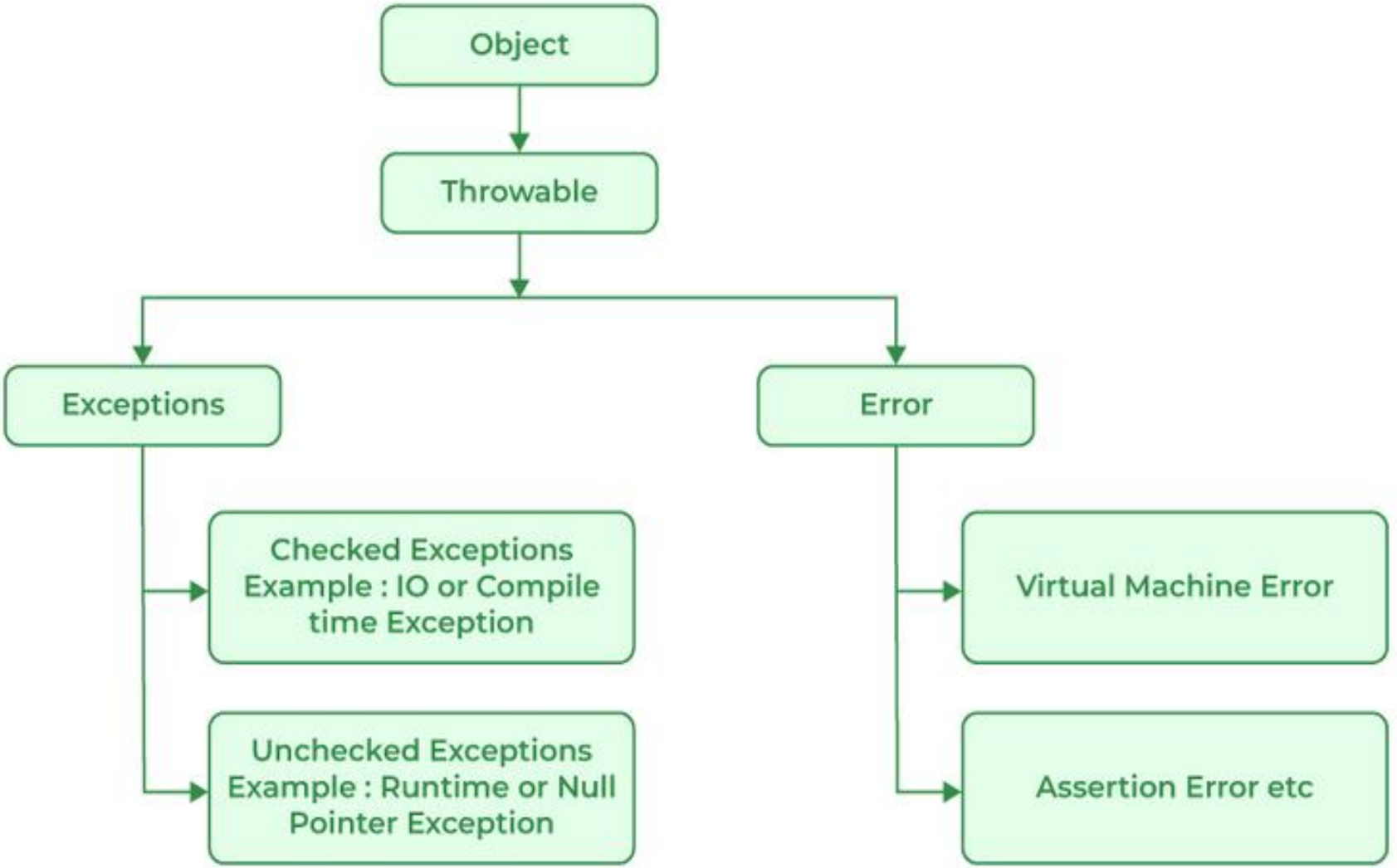
- Invalid user input
- Device failure
- Loss of network connection
- Physical limitations (out-of-disk memory)
- Code errors
- Opening an unavailable file

Errors

Errors represent irrecoverable conditions such as Java virtual machine (JVM) running out of memory, memory leaks, stack overflow errors, library incompatibility, infinite recursion, etc.

Errors are usually beyond the control of the programmer, and we should not try to handle errors.

Exception Hierarchy

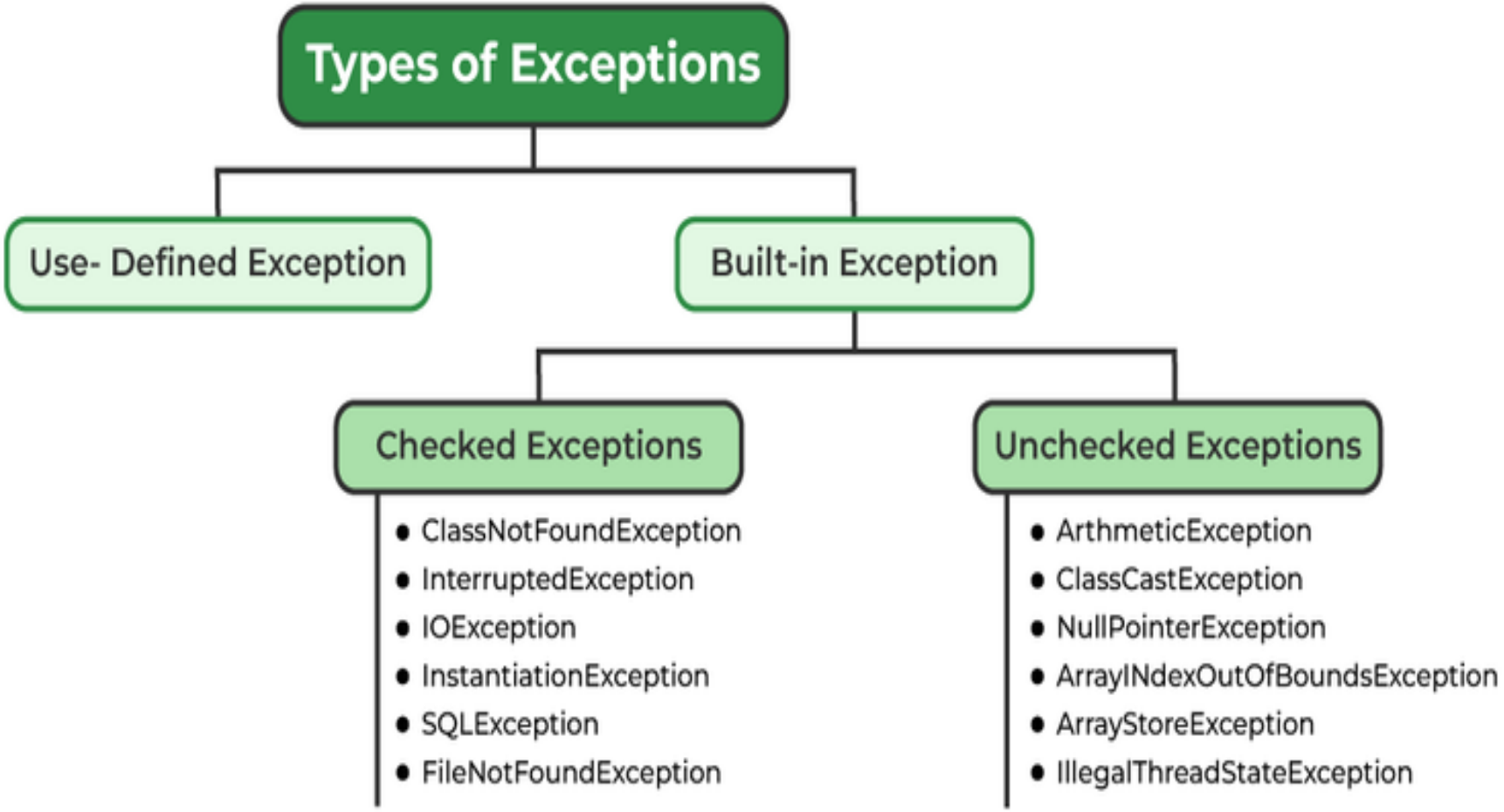


Difference between Error and Exception

Error: An Error indicates a serious problem that a reasonable application should not try to catch.

Exception: Exception indicates conditions that a reasonable application might try to catch.

Types of Exceptions



Exceptions can be categorized in two ways:

1. **Built-in Exceptions**
   - Checked Exception
   - Unchecked Exception
2. **User-Defined Exceptions**

1. Built-in Exceptions

Built-in exceptions are the exceptions that are available in Java libraries. These exceptions are suitable to explain certain error situations.

Checked Exceptions: Checked exceptions are called compile-time exceptions because these exceptions are checked at compile-time by the compiler.

Unchecked Exceptions:

The unchecked exceptions are just opposite to the checked exceptions.

The compiler will not check these exceptions at compile time.

In simple words, if a program throws an unchecked exception, and even if we didn't handle or declare it, the program would not give a compilation error.

2. User-Defined Exceptions:

Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, users can also create exceptions, which are called ‘user-defined Exceptions’.

The advantages of Exception Handling in Java are as follows:

- Provision to Complete Program Execution
- Easy Identification of Program Code and Error-Handling Code
- Propagation of Errors
- Meaningful Error Reporting
- Identifying Error Types