### **1 . Access Modifiers in Java**

**Access modifiers**

Access modifiers in Java are keywords that control the visibility (accessibility) of classes, methods, variables, and constructors within your program. They are a fundamental concept in object-oriented programming, particularly for enforcing encapsulation, a core principle that restricts direct access to an object's internal components.

Java provides four access modifiers:

1. **public**
2. **protected**
3. **default** (no modifier specified)
4. **private**

Java provides four main access modifiers:

1. **Public:**

This modifier allows elements (classes, methods, variables, constructors) to be accessed from any other class in the application, regardless of the package it resides in.

1. **Private:**

This modifier restricts access to elements only within the class they are declared. They cannot be accessed from outside the class, not even from classes within the same package.

1. **Protected:**

This modifier allows access from within the same package (where the class is defined) and also from subclasses in different packages

1. **Default (Package-Private):**

If you don't explicitly use any access modifier keyword, the element is considered package-private by default. This means it can be accessed by classes within the same package but not from outside the package.

**2 .Exception and Error**

**Exception and Error**

In Java, exceptions and errors are subclasses of the Throwable class. Both are used to signal that something unexpected has occurred in a program, but they serve different purposes and are handled differently. Here's a detailed overview of exceptions and errors in Java:

### **Exceptions**

**Exceptions** represent conditions that a program might want to catch and handle. They are divided into two categories: **checked exceptions** and **unchecked exceptions**.

**3. Checked Exceptions And Unchecked Exceptions**

Checked exceptions are exceptions that are checked at compile-time. If a method can throw a checked exception, it must declare it using the throws keyword, and the calling method must handle or declare it as well.

##### **Examples of Checked Exceptions**

* IOException
* SQLException
* ClassNotFoundException

### **1. IOException**

IOException is a general class of exceptions produced by failed or interrupted I/O operations. It is part of the java.io package.

#### **Common Scenarios for IOException**

* Reading or writing to a file
* Network communication failures
* Issues with input and output streams

### **2. SQLException**

SQLException is an exception that provides information on a database access error or other errors. It is part of the java.sql package.

#### **Common Scenarios for SQLException**

* Database connection failures
* SQL syntax errors
* Constraint violations
* Issues with transactions

### **3. ClassNotFoundException**

ClassNotFoundException is thrown when an application tries to load a class through its string name but no definition for the class with the specified name could be found. It is part of the java.lang package.

#### **Common Scenarios for ClassNotFoundException**

* Dynamically loading classes using Class.forName()
* Issues with the classpath configuration

#### **Unchecked Exceptions**

Unchecked exceptions are exceptions that are not checked at compile-time but at runtime. These include RuntimeException and its subclasses. You are not required to declare or handle these exceptions.

##### **Examples of Unchecked Exceptions**

* NullPointerException
* ArrayIndexOutOfBoundsException
* ArithmeticException

### **1. NullPointerException**

NullPointerException is thrown when an application attempts to use null in a case where an object is required. This includes:

* Calling the instance method of a null object.
* Accessing or modifying the field of a null object.
* Taking the length of a null array.
* Accessing or modifying the elements of a null array.
* Throwing null as if it were a Throwable value.

### **2. ArrayIndexOutOfBoundsException**

ArrayIndexOutOfBoundsException is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array

### **3. ArithmeticException**

ArithmeticException is thrown when an exceptional arithmetic condition has occurred. For example, it can be thrown when an integer is divided by zero.

### **Errors**

**Errors** represent serious issues that are typically beyond the control of the application and usually cannot be recovered from. They are subclasses of the Error class. Errors indicate problems that are mostly related to the environment in which the application is running.

##### **Examples of Errors**

* OutOfMemoryError
* StackOverflowError
* VirtualMachineError

##### **Handling Errors**

Errors are usually not handled within the application because they represent critical failures. However, you can still catch them using a try-catch block, but it is generally not recommended.

**Exceptions**: Used for conditions that the application might want to catch and handle. Divided into checked (must be declared or handled) and unchecked (do not need to be declared or handled).

**Errors**: Indicate serious problems that are usually outside the application's control and typically should not be caught.