**Access Specifiers In Java**

Access specifiers helps to restrict the scope of a class, constructor, variable, method, or data member. There are four types of access modifiers in java.

* Default
* Private
* Protected
* Public

**Default:**

* When no access modifier is specified for a class, method, or data member – It is said to be having the **default** access modifier by default.
* The data members, class or methods which are not declared using any access modifiers i.e. having default access modifier are accessible **only within the same package**.

**Example:**

**package** p1;

**class** Geek

{

**void** display()

{

 System.out.println("Hello World!");

    }

}

****Private**:** The private access modifier is specified using the keyword **private**.

* The methods or data members declared as private are accessible only **within the class** in which they are declared.
* Any other **class of**the **same package will not be able to access** these members.
* Top-level classes or interfaces can not be declared as private because
* private means “only visible within the enclosing class”.
* protected means “only visible within the enclosing class and any subclasses”.

**Example:**

**package** p1;

**class** A

{

**private** **void** display()

    {

        System.out.println("GeeksforGeeks");

    }

}

**class** B

{

**public** **static** **void** main(String args[])

    {

        A obj = **new** A();

        obj.display(); } }

****Public**:** The public access modifier is specified using the keyword **public**.

* The public access modifier has the **widest scope** among all other access modifiers.
* Classes, methods, or data members that are declared as public are **accessible from everywhere** in the program. There is no restriction on the scope of public data members.

**Example:**

**package** p2;

**import** p1.\*;

**class** B {

**public** **static** **void** main(String args[])

    {

        A obj = **new** A;

        obj.display();

    }

}

### Protected:

* The protected access modifier is accessible within package and outside the package but through inheritance only.
* The protected access modifier can be applied on the data member, method and constructor. It can't be applied on the class.
* It provides more accessibility than the default modifer.

**Example:**

package pack;

public class A{

protected void msg(){System.out.println("Hello");}

}

package mypack;

import pack.\*;

class B extends A{

  public static void main(String args[]){

   B obj = new B();

   obj.msg();

  }  }

# Exception Handling in Java

The **Exception Handling in Java** is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.

Exception is an abnormal condition.

In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

### Types of Java Exceptions

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three types of exceptions namely:

* Checked Exception
* Unchecked Exception
* Error

### Checked Exception

The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.

### Unchecked Exception

The classes that inherit the RuntimeException are known as unchecked exceptions. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

### 3) Error

Error is irrecoverable. Some example of errors are OutOfMemoryError, VirtualMachineError, AssertionError etc.

## java Exception Keywords

Java provides five keywords that are used to handle the exception. The following table describes each.

Try

Catch

Finally

Throw

Throws

* The try keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.
* The catch block is used to handle the exception.It must be preceded by try block which means we cant use catch block alone.It can be followed by finally block later.
* The finally block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.
* The throw keyword is used to throw an exception.
* The throws keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature.

## Java Exception Handling Example

Let's see an example of Java Exception Handling in which we are using a try-catch statement to handle the exception.

****JavaExceptionExample.java****

**public** **class** JavaExceptionExample{

**public** **static** **void** main(String args[]){

**try**{

**int** data=100/0;

   }**catch**(ArithmeticException e){System.out.println(e);}

      System.out.println("rest of the code...");

  }  }