**TASK- 11**

**Q1.Access Modifiers in Java**

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four Access Modifiers

* Public
* Private
* Protected
* Default

**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.

**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’t be declared as private because
* private means only visible within the enclosing class

This modifier is applied only to nested classes and not on top-level classes.

For example, if two classes are created (class a & class b) under a single package.

Private method will be created in class A and that method will be tried to access from class b to get the result.

**Protected**

The protected access modifier is specified using the keyword **protected.**

The methods or data members declared as protected are **accessible within the same package or subclasses in different packages.**

Protected means “only visible within the enclosing class and any subclasses”

**For, example two packages are created package a & package b. If a protected method is created in class a and class b is inherited from class a,** this protected method is then accessed by creating an object of class b.

**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, classes, or methods that are not declared using any access modifiers i.e. having default access modifiers are accessible **only within the same package.**

For, example if we create two packages package a and package b, the classes in the package will have default access modifiers and we will try to access a class from package a to another class of package b.

**Q2.Difference between Exception and Error**

|  |  |  |
| --- | --- | --- |
|  | **Exception** | **Error** |
| **Definition** | An issue that can disrupt the normal flow of a program but can be caught and handled. | A serious problem that cannot be recovered from, typically arising from system-level issues. |
| **Origin** | Application code, including invalid input or incorrect API usage. | System abnormalities such as hardware failures, system crashes, or out of memory. |
| **Recovery** | Can often be recovered from using try-catch blocks. | Fatal and non-recoverable. |
| **Types** | Checked Exceptions (detected at compile time) and Unchecked Exceptions (occur at runtime). | Syntax Error, Runtime Error, Logical Error. |
| **Examples** | IO Exception, Null Pointer Exception, SQL Exception, etc. | Out Of Memory Error, StackOverflowError. |
| **Handling** | Can be caught and handled in the program to maintain flow or recover from the situation. | Cannot be handled or caught by the program. |
| **Predictability** | Can be anticipated and handled through proper coding practices. | Unpredictable and often outside the control of the application. |
| **Impact** | Disrupts the normal flow but allows for redirection or handling within the program. | May cause the program (and potentially the system) to terminate. |
| **occurrence** | Primarily occur at runtime, though checked exceptions can be detected at compile time. | Can occur both at compile time and runtime |

**Q3. Difference between Checked Exception and Unchecked Exception**

|  |  |
| --- | --- |
| **Checked Exception** | **Unchecked Exception** |
| Checked exceptions occur at compile time. | Unchecked exceptions occur at runtime. |
| The compiler checks a checked exception. | The compiler does not check these types of exceptions |
| These types of exceptions can be handled at the time of compilation. | These types of exceptions cannot be a catch or handle at the time of compilation, because they get generated by the mistakes in the program. |
| They are the sub-class of the exception class. | They are runtime exceptions and hence are not a part of the Exception class. |
| The JVM needs the exception to catch and handle. | The JVM does not require the exception to catch and handle. |
| Some of checked Exception   * No Such Field Exception * File Not Found Exception * Interrupted Exception * No Such Method Exception * Class Not Found Exception | Some of Unchecked Exception   * No Such Element Exception * Undeclared Throwable Exception * Empty Stack Exception * Arithmetic Exception * Null Pointer Exception * Array Index Out of Bounds Exception * Security Exception |