**Java Exceptions and Error Handling**

**Access Modifiers:**

Access modifiers are keywords that can be used to control the visibility of fields, methods, and constructors in a class. The four access modifiers in Java are public, protected, default, and private.

**Four Types of Access Modifiers**

* **Private**: We can access the **private modifier** only within the same class and not from outside the class.
* **Default:** We can access the **default modifier** only within the same package and not from outside the package. And also, if we do not specify any access modifier it will automatically consider it as default.
* **Protected**: We can access the protected modifier within the same package and also from outside the package with the help of the child class. If we do not make the child class, we cannot access it from outside the package. So inheritance is a must for accessing it from outside the package.
* **Public**: We can access the public modifier from anywhere. We can access public modifiers from within the class as well as from outside the class and also within the package and outside the package.

Here are examples of access modifiers in Java:

**// Public access modifier**  
public class PublicClass {  
public int publicField;  
public void publicMethod() {  
// code here  
}  
}

**// Protected access modifier**  
protected class ProtectedClass {  
protected int protectedField;  
protected void protectedMethod() {  
// code here  
}  
}

**// Default (package-private) access modifier**  
class DefaultClass {  
int defaultField;  
void defaultMethod() {  
// code here  
}  
}  
**// Private access modifier**  
class PrivateClass {  
private int privateField;  
private void privateMethod() {  
// code here  
}  
}

**Significances of Access Modifiers in terms of class, method and variable accessibility:**

**Define a class:** Create a class that represents the object you want to manage.

**Define instance variables:** Within the class, define instance variables that represent the data you want to manage.

Specify an access modifier: For each instance variable, specify an access modifier that determines the visibility of the variable. The three main access modifiers in Java are private, protected, and public.

**Use private for variables that should only be accessible within the class:** If you want to prevent access to a variable from outside the class, use the private access modifier. This is the most restrictive access modifier and provides the greatest level of encapsulation.

**Use protected for variables that should be accessible within the class and its subclasses**: If you want to allow access to a variable from within the class and its subclasses, use the protected access modifier. This is less restrictive than private and provides some level of inheritance.

**Use public for variables that should be accessible from anywhere**: If you want to allow access to a variable from anywhere, use the public access modifier. This is the least restrictive access modifier and provides the least amount of encapsulation.

**Use accessor and mutator methods to manage access to the variables:** In order to access and modify the variables, use accessor (getter) and mutator (setter) methods, even if the variables have a public access modifier. This provides a level of abstraction and makes your code more maintainable and testable.

**Difference between error and exception**

**Errors**

The error signifies a situation that mostly happens due to the absence of system resources. The system crash and memory errors are an example of errors. It majorly occurs at runtime.

**Exceptions**

The exceptions are the issues that can appear at runtime and compile time. It majorly arises in the code or program authored by the developers. There are two types of exceptions: Checked exceptions and Unchecked exceptions.

|  |  |  |
| --- | --- | --- |
| S.No | Errors | Exceptions |
| 1 | The error indicates trouble that primarily occurs due to the scarcity of system resources. | The exceptions are the issues that can appear at runtime and compile time. |
| 2 | It is not possible to recover from an error. | It is possible to recover from an exception. |
| 3 | In java, all the errors are unchecked. | In java, the exceptions can be both checked and unchecked. |
| 4 | The system in which the program is running is responsible for errors. | The code of the program is accountable for exceptions. |
| 5 | They are described in the **java.lang.Error** package. | They are described in **java.lang.Exception** package |

**Differences between Checked Exception and Unchecked Exception:**

|  |  |
| --- | --- |
| Checked exceptions | Unchecked exceptions |
| 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. |
| Here, the JVM needs the exception to catch and handle. | Here, the JVM does not require the exception to catch and handle. |
| Examples of Checked exceptions: File Not Found Exception  No Such Field Exception  Interrupted Exception  No Such Method Exception  Class Not Found Exception | Examples of Unchecked Exceptions: No Such Element Exception  Undeclared Throwable Exception  Empty Stack Exception  Arithmetic Exception  Null Pointer Exception  Array Index Out of Bounds Exception  Security Exception |