1. What are the four access modifiers available in Java and what is their significance in terms of class, method, and variable accessibility?

**1. Public**

* **Class**: Can be accessed from anywhere.
* **Method**: Can be called from any other class.
* **Variable**: Can be used from any other class.

**2. Protected**

* **Class**: Cannot be used.
* **Method**: Can be called within the same package and from subclasses.
* **Variable**: Can be used within the same package and from subclasses.

**3. Default (Package-Private)**

* **Class**: Can be accessed only within the same package.
* **Method**: Can be called only within the same package.
* **Variable**: Can be used only within the same package.

**4. Private**

* **Class**: Cannot be used (except for inner classes).
* **Method**: Can be called only within the same class.
* **Variable**: Can be used only within the same class.

**Summary**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Access Modifier** | | |  | | --- | | **Access Modifier** | | |  | | --- | | **Access Modifier** | | |  | | --- | | **Access Modifier** | |
| Public | Accessible from any class | Accessible from any class | Accessible from any class |
| Protected | Not applicable | Accessible within package and subclasses | Accessible within package and subclasses |
| Default | Accessible within package | Accessible within package | Accessible within package |
| Private | Not applicable | Accessible within the same class | Accessible within the same class |

2. What is the difference between Exception and error?

**Exception**

* **Definition**: An Exception represents conditions that a program might want to catch and handle, such as input/output errors, invalid user input, or network issues.
* **Types**: There are two main types of exceptions:
  1. **Checked Exceptions**: These are exceptions that are checked at compile-time. The programmer must handle these exceptions, either by using a try-catch block or by declaring the exception using the throws keyword. Examples include IOException, SQLException, and FileNotFoundException.
  2. **Unchecked Exceptions (Runtime Exceptions)**: These are exceptions that occur at runtime and are not checked at compile-time. They are usually a result of programming errors, such as accessing an out-of-bounds array element or null pointer dereference. Examples include NullPointerException, ArrayIndexOutOfBoundsException, and ArithmeticException.
* **Usage**: Exceptions are intended to be caught and handled by the program to allow graceful recovery or user notification.

3. What is the difference between checked Exception and unchecked Exception?

### Checked Exceptions

* **Definition**: Checked exceptions are exceptions that are checked at compile-time. The Java compiler enforces that these exceptions are either handled with a try-catch block or declared in the method signature using the throws keyword.
* **Examples**: IOException, SQLException, ClassNotFoundException.
* **Handling**: Methods that can throw checked exceptions must either handle them using a try-catch block or declare them in the method’s throws clause. This ensures that the calling method is aware of the potential exception and handles it appropriately.
* **Purpose**: Checked exceptions are typically used for recoverable conditions and represent situations that a reasonable application might want to catch and handle.

### Unchecked Exceptions

* **Definition**: Unchecked exceptions are exceptions that are not checked at compile-time. These exceptions are derived from RuntimeException and its subclasses.
* **Examples**: NullPointerException, ArrayIndexOutOfBoundsException, ArithmeticException.
* **Handling**: There is no requirement for methods to catch or declare unchecked exceptions. They can propagate up the call stack and can be caught and handled at runtime if desired.
* **Purpose**: Unchecked exceptions typically represent programming errors, such as logic errors or improper use of an API. These are conditions that the application usually cannot recover from, and they indicate a bug in the code.