In Java, there are four access modifiers:

1. **Public**: When a class, method, or variable is declared as public, it is accessible from any other class in the Java application, regardless of the package in which it resides. This is the broadest level of access.
2. **Protected**: A protected class, method, or variable is accessible within its own package as well as by subclasses, even if they are in different packages. This modifier provides a balance between accessibility and encapsulation, allowing limited access to certain entities.
3. **Default (Package-private)**: When no access modifier is specified, it defaults to package-private access. Classes, methods, or variables with default access are only accessible within the same package. This is useful for hiding implementation details from classes outside the package.
4. **Private**: Declaring a class member as private restricts its access solely to the containing class. Neither subclasses nor other classes can access private members. This provides the highest level of encapsulation and data hiding.

These access modifiers allow developers to control the visibility and accessibility of classes, methods, and variables within their Java programs. They play a crucial role in encapsulation, security, and maintaining a clear and manageable codebase.

Q2)

**Exception:**

* An exception is like a bump in the road during the execution of your program.
* It's a problem that can be managed and handled by your code.
* Examples include reading a file that doesn't exist or trying to divide by zero.

**Error:**

* An error is more like hitting a brick wall while driving your program.
* It's usually a severe problem that your program can't recover from.
* Examples include running out of memory or encountering a system crash.

In essence, exceptions are problems that your program can deal with, while errors are usually more serious issues that your program can't fix on its own.

Q3) **Checked Exceptions:**

* These are like warnings that something might go wrong while your program runs.
* The compiler checks if you've prepared for these problems, like file not found or network issues.
* You must handle them in your code or say that your method might throw them.

**Unchecked Exceptions:**

* These are like surprises that pop up unexpectedly while your program runs.
* They're often caused by programming mistakes, like dividing by zero or trying to access something that doesn't exist.
* You don't have to handle them if you don't want to, but it's usually a good idea to prevent your program from crashing unexpectedly.