**Best Practices for Error Handling:**

1. Be specific with exceptions - Catch specific exceptions rather than using a general Exception catch block. This allows you to handle different exceptions differently.
2. Graceful degradation - The practice of designing applications to continue functioning even when it encounters errors or conditions that it cannot handle fully.
3. Avoid catching throwable - The “Throwable” class is the superclass of all errors and exceptions in Java, including Error and Exception subclasses.

Not all errors should be caught: Throwable includes Error subclasses like **OutOfMemoryError**, **StackOverflowError**, etc. These errors typically indicate severe and unrecoverable issues that should not be caught and handled programmatically. Attempting to handle these errors may lead to unpredictable behavior or worsen the situation by hiding the root cause.

1. Close resources – solution is you can declare and use the resource in try block as it automatically closes resource
2. Throw early, Catch late : when you have enough context and capability to handle the exception.
3. Use custom exceptions for application-specific errors: Create custom exception classes for specific errors in your application. This helps in distinguishing different types of errors and handling them appropriately.
4. Log exceptions properly.