**LOUIE JAY A. CENIZA MARCH 15, 2024**

**ERROR HANDLING (CHAPTER 7)**

**Use Exception Rather Than Return Codes.** As the title implies it’s better to use exceptions rather than return codes because it allows you to have a cleaner code than a cluttering code in handling errors. It also provides more flexibility in how errors are handled and allows for more detailed error messages to be provided to the caller.

**Write Your Try-Catch-Finally Statement First.** When creating an operation to your functions put it inside a try-catch-finally statement so that you can say that it is properly working and if has an error it will relay the error message and complete it with the final message that it is completed due to your try-catch-finally statement instead of only try-catch statement.

**Use Unchecked Exceptions.** Using unchecked exceptions can simplify the code and reduce clutter by removing the need for extensive error handling in every method.

**Provide Context with Exceptions.** Each exception that you throw should provide enough context to determine the source and location of an error.

**Define Exception Classes in Terms of a Caller’s Needs.** In this lesson you need to apply the provide context with exceptions so that it will know which class or error or where it occurs.

**Define the Normal Flow.** It refers to the main path of typical sequence of operations in your program, where it executes without encountering any errors.

**Don’t Return Null.** Avoid returning null to your codes, it may cause some runtime errors because of accessing some properties or methods and make it a null. Handle the nulls by returning empty collections or objects so that you can Identify it, and throwing exceptions.

**Don’t Pass Null.** It is much worse than returning null, you should handle null because it would make your code throws errors. Theres no good way to deal with nulls because it is by default, but you need to prevent some of it.