JASMINE LISONDRA

Chapter 7: Error Handling

Use Exceptions Rather Than Return Codes

Using exceptions is better than return codes for handling errors. Exceptions make the code cleaner and allow for more detailed error messages, giving you more flexibility in how errors are handled.

Write Your Try-Catch-Finally Statement First

When you write a function, start with a try-catch-finally statement. This makes sure your function works properly and handles errors correctly. The finally part ensures that some code runs no matter what, even if there's an error.

Use Unchecked Exceptions

Unchecked exceptions make the code simpler by reducing the need for a lot of error handling in every method. This helps keep the code less cluttered.

Provide Context with Exceptions

When you throw an exception, include enough information to show where the error happened. This helps in finding and fixing the problem faster.

Define Exception Classes in Terms of a Caller's Needs

Create exception classes that make sense to the person using your code. This way, they can understand where and why an error occurred.

Define the Normal Flow

This means writing the main path of your program so it runs smoothly without errors. Your code should follow a clear and typical sequence of operations.

Don't Return Null

Avoid returning null in your code. Returning null can cause runtime errors if you try to use a null object. Instead, return empty collections or objects, or throw exceptions to handle errors better.

Don't Pass Null

Passing null is worse than returning it. Handle nulls properly because passing them can cause errors. It's hard to manage nulls, but you need to prevent them from causing problems in your code.

Chapter 8: Boundaries

Using Third-Party Code

Third-party code includes libraries, frameworks, or modules that you add to your codebase. These can speed up development and add functionality to your system. However, relying on third-party code can also introduce risks related to availability, compatibility, and stability.

Exploring and Learning Boundaries

When you use new third-party code, it's tempting to spend a lot of time reading the documentation and experimenting. Instead, write tests to explore how the third-party code works and what its limits are. This saves time and helps you understand it better.

Learning Tests Are Better Than Free

Learning tests are useful because you need to understand the API you're using. When you use third-party packages, running tests can show if they are compatible with what you need.

Using Code That Does Not Yet Exist

Sometimes you need to work with ideas or code that aren't fully developed yet. Keep exploring and experimenting with new concepts so you don't get stuck and can keep making progress.