CODE REFACTORING :

Refactoring is the process of improving code structure without changing its behavior. In Agile, teams refine code incrementally each Sprint. Without refactoring, projects may suffer from poor code quality, including unhealthy dependencies, improper class responsibilities, and code duplication. This leads to complexity and reduced maintainability. Refactoring helps simplify and organize code, ensuring clarity and efficiency.

Advantages of Refactoring:

- ☐ Provides more readable and modular code
- ☐ e maintenance.
- ☐ Architecture improvement without impacting software behavior
- ☐ Modular component refactored to maximize possible reusability

---

'Code Smell' is the phrase coined by Kent Back and Martin Fowler. Some of the most important "smells in code" are:

1- Duplicated code

2-Long Method

3-Too Many Parameters

4-Lazy Class

Guidelines for Refactoring:

- Make sure the code is working before you start.
- Run the tests frequently before, during, and after each refactoring.
- Before each refactoring, use a version control tool and save a checkpoint.
- Break each refactoring down into smaller units.
- Finally, if a refactoring tool is available in your environment, utilize it.

Clean Code Principles

Clean code is clear, well-structured, and easy to understand and maintain. Below are the key principles for writing clean code:

.Use Meaningful and Descriptive Names

.Avoid Code Duplication (Don’t Repeat Yourself - DRY)

.Keep Functions Short and Focused

.Use Simple and Clear Conditions

Ex for naming and variabels

```
def calculate_area(width, height):

    return width * height
```