SOLID Det korte og det lange

1. **Single Responsibility Principle (SRP)**

- **Definition**: A class should have one, and only one, reason to change. This means that a class should have only one job or responsibility.
- **Purpose**: The SRP helps in reducing the complexity of the software by dividing it into smaller, more manageable pieces. It also makes the code more readable and maintainable.

2. **Open/Closed Principle (OCP)**

- **Definition**: Software entities (classes, modules, functions, etc.) should be open for extension but closed for modification. This means you should be able to add new functionality without changing existing code.
- **Purpose**: The OCP helps in creating a flexible system that can evolve over time without requiring major changes to the existing codebase, thus reducing the risk of introducing new bugs.

3. **Liskov Substitution Principle (LSP)**

- **Definition**: Subtypes must be substitutable for their base types. This means that objects of a derived class should be able to replace objects of the base class without affecting the correctness of the program.
- **Purpose**: The LSP ensures that a derived class can stand in for its base class without causing unexpected behavior, promoting code reusability and robustness.

4. **Interface Segregation Principle (ISP)**

- **Definition**: Clients should not be forced to depend on interfaces they do not use. This means that larger interfaces should be split into smaller, more specific ones so that clients only need to know about the methods that are of interest to them.
- **Purpose**: The ISP helps in creating a more modular and decoupled system by ensuring that classes only depend on the methods they need, making the code easier to maintain and modify.

5. **Dependency Inversion Principle (DIP)**

- **Definition**: High-level modules should not depend on low-level modules. Both should depend on abstractions. Also, abstractions should not depend on details. Details should depend on abstractions.
- **Purpose**: The DIP aims to reduce the coupling between high-level and low-level modules, making the system more flexible and easier to maintain.