# **Day 20 - 25 July 2025**

**Document Name:**Day 20 - hmuvvala@ - Hari Gopal Muvvala

### SOLID Principles – Summary Notes

#### 1. S — Single Responsibility Principle (SRP)

A class should have **only one reason to change**.

* Each class should focus on **one task or functionality**.
* Helps in better **modularity and maintainability**.

✅ Split classes based on **separation of concerns**.

#### 2. O — Open/Closed Principle (OCP)

Software entities should be **open for extension, but closed for modification**.

* New features should be added by **extending**, not changing existing code.
* Promotes use of **abstraction, inheritance, and polymorphism**.

✅ Prefer **interfaces and abstract classes** for future extension.

#### 3. L — Liskov Substitution Principle (LSP)

Subtypes must be **substitutable for their base types**.

* Derived classes should not **break the behavior** of the base class.
* Ensures **polymorphism works safely**.

✅ Avoid designing parent classes with methods that don’t apply to all children.

#### 4. I — Interface Segregation Principle (ISP)

No class should be **forced to implement** methods it does not use.

* Create **smaller, role-specific interfaces**.
* Promotes **cleaner contracts** between interfaces and implementers.

✅ Split large interfaces into **cohesive, meaningful parts**.

#### 5. D — Dependency Inversion Principle (DIP)

High-level modules should not depend on low-level modules.

Both should depend on **abstractions**.

* Avoid tight coupling between classes.
* Use **interfaces or abstract classes** to inject dependencies.

✅ Use **constructor injection or interfaces** to manage dependencies.

### Notes for Task 3 – UML Class Diagram (SRP)

* This UML diagram represents the **Single Responsibility Principle (SRP)** applied to a Book class.
* The diagram includes three classes:
  + Task3\_Book: Holds data (title, author, price) and provides getter methods.
  + Task3\_TitleFormatter: Has a method to format the book title — only concerned with **presentation logic**.
  + Task3\_DiscountCalculator: Has a method to calculate discounted price — only handles **pricing logic**.
* Arrows in the diagram indicate **dependency (uses-a)** relationships:
  + Both formatter and calculator classes depend on Task3\_Book.
* Each class has **one clearly defined responsibility**, matching the SRP goal.
* This separation also helps in **unit testing** and **future refactoring**, since changes in formatting or discount logic won’t affect the Book class.

