SOLID Principles

A WORKSHOP USING FINTECH PAYMENT ENGINE

SRP - Single Responsibility Principle

- Each class should have only one reason to change.
- ✅ Applied to: PaymentEngine components like FraudChecker, Logger / ReportStorage.
- ❌ Avoid: Mixing fraud logic, notification, and logging in one class.

OCP - Open/Closed Principle

- Software entities should be open for extension, closed for modification.
- ✅ Applied to: PaymentGateway Map - add new gateways without modifying core.
- ❌ Avoid: switch/case on gateway type inside engine.

LSP - Liskov Substitution Principle

- Subtypes must be substitutable for their base types.
- ✅ Applied to: StripeGateway, CryptoGateway both implement PaymentGateway.
- ❌ Avoid: Creating subclasses that throw 'UnsupportedOperationException'.

ISP - Interface Segregation Principle

- Clients should not be forced to depend on interfaces they do not use.
- ✅ Applied to: Separate EmailNotifier and SMSNotifier.
- ❌ Avoid: Notifier interface with sendEmail, sendSMS, sendPush combined.

DIP - Dependency Inversion Principle

- High-level modules should not depend on low-level modules.
- ✅ Applied to: Injecting FraudChecker, ReportStorage via interfaces.
- ❌ Avoid: Using 'new' to instantiate dependencies in PaymentEngine.

Workshop Takeaways

- ✔ SRP: Isolate responsibilities in distinct classes
- ✔ OCP: Allow system behavior to change via extensions
- ✔ LSP: Respect substitutability in polymorphic design
- ✔ ISP: Design fine-grained interfaces by client needs
- ✔ DIP: Invert dependency flow using abstractions

- 💡 Modular, extensible, testable code is maintainable code.