Java Programming II

Lab1

514770-1
Fall 2024
9/11/2024

Kyoung Shin Park
Computer Engineering
Dankook University

DRY (Don’t Repeat Yourself) Principle

- In the book “The Pragmatic Programmer”, DRY is defined as “Every piece of **knowledge** must have a single, unambiguous, authoritative representation within a system."
  - Knowledge – a precise functionality or an algorithm
- Violations of DRY
  - WET, "We enjoy typing," or “Waste everyone’s time”.
- How to Achieve DRY
  - To avoid violating the DRY principle, divide your system into pieces. Divide your code and logic into **smaller reusable units** and use that code by calling it where you want.
- DRY Benefits
  - Less code is good: It saves time and effort, is easy to maintain, and also reduces the chances of bugs.

KISS (Keep It Simple Stupid) Principle

- "Keep It Simple Stupid", "Keep It Short and Simple"
- The KISS principle is descriptive to **keep the code simple and clear**, making it **easy to understand**.
- Violations of KISS
  - "Why they have written these unnecessary lines and conditions when we could do the same thing in just 2-3 lines?"
- How to Achieve KISS
  - To avoid violating the KISS principle, try to **write simple code**. Whenever you find lengthy code, divide that into multiple methods — **refactor**.
- KISS Benefits
  - If the code is written simply, then there will not be any difficulty in understanding that code, and also will be easy to modify.

YAGNI (You Aren't Gonna Need It) Principle

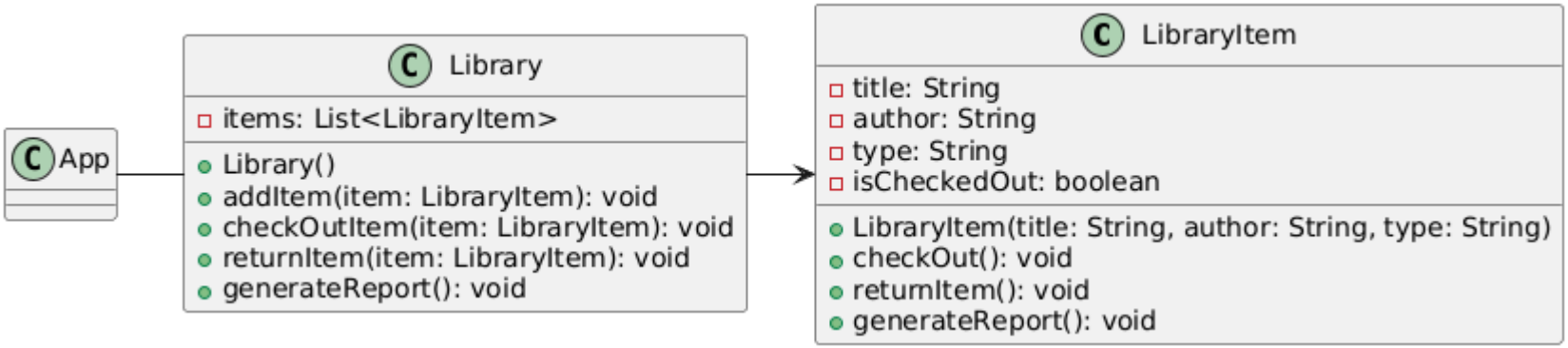
- YAGNI says "don't implement something until it is necessary." YAGNI tells us to cut off any unnecessary part while KISS advises to make the rest as simple as possible.
- Violations of YAGNI
  - "over engineering" - a feature for every possible case, functions with a lot of input parameters, multiple if-else branches, rich and detailed interfaces, all those could be a smell of over engineering.
- How to Achieve YAGNI
  - Always **implement things when you actually need them**, never when you just foresee that you need them.
- YAGNI Benefits
  - Software developers don't have enough information to make the call on extra features, the time spent could be used elsewhere more productively. Extra features mean extra development time, testing time, documentation time, code review time.

SOLID Principle

- **Single Responsibility Principle**
  - "A class should have one, and only one, reason to change."
- **Open/Closed Principle**
  - "Software entities (e.g. classes, modules, functions, etc) should be open for extension, but closed for modification."
- **Liskov Substitution Principle**
  - "Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program."
- **Interface Segregation Principle**
  - "Clients should not be forced to depend upon interfaces that they do not use." Reduce fat interfaces into multiple smaller and more specific client specific interfaces.
- **Dependency Inversion Principle**
  - One should depend on abstractions (interfaces and abstract classes) instead of concrete implementations (classes).

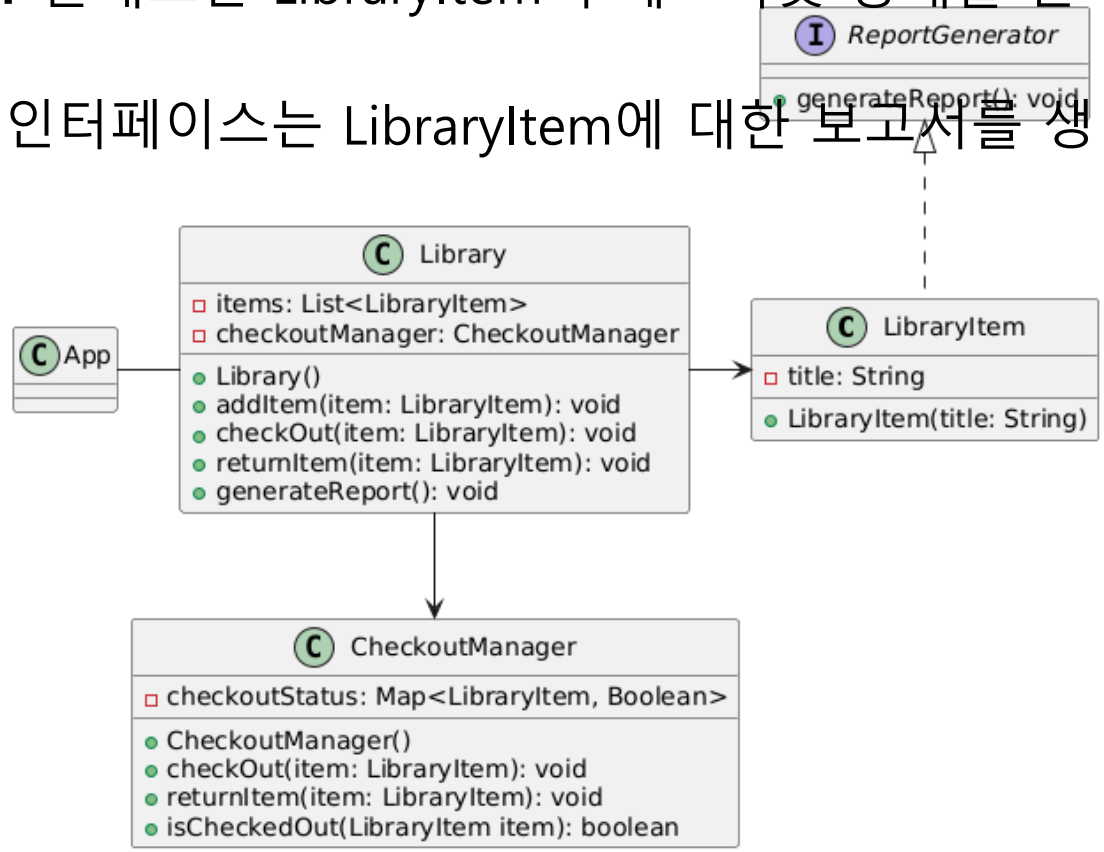
Lab1

- 도서관 항목 목록을 보고하는 LibraryItem과 Library 프로그램이 Template 으로 주어졌을 때, SOLID 원칙에 따라 이 프로그램을 다시 작성하라.



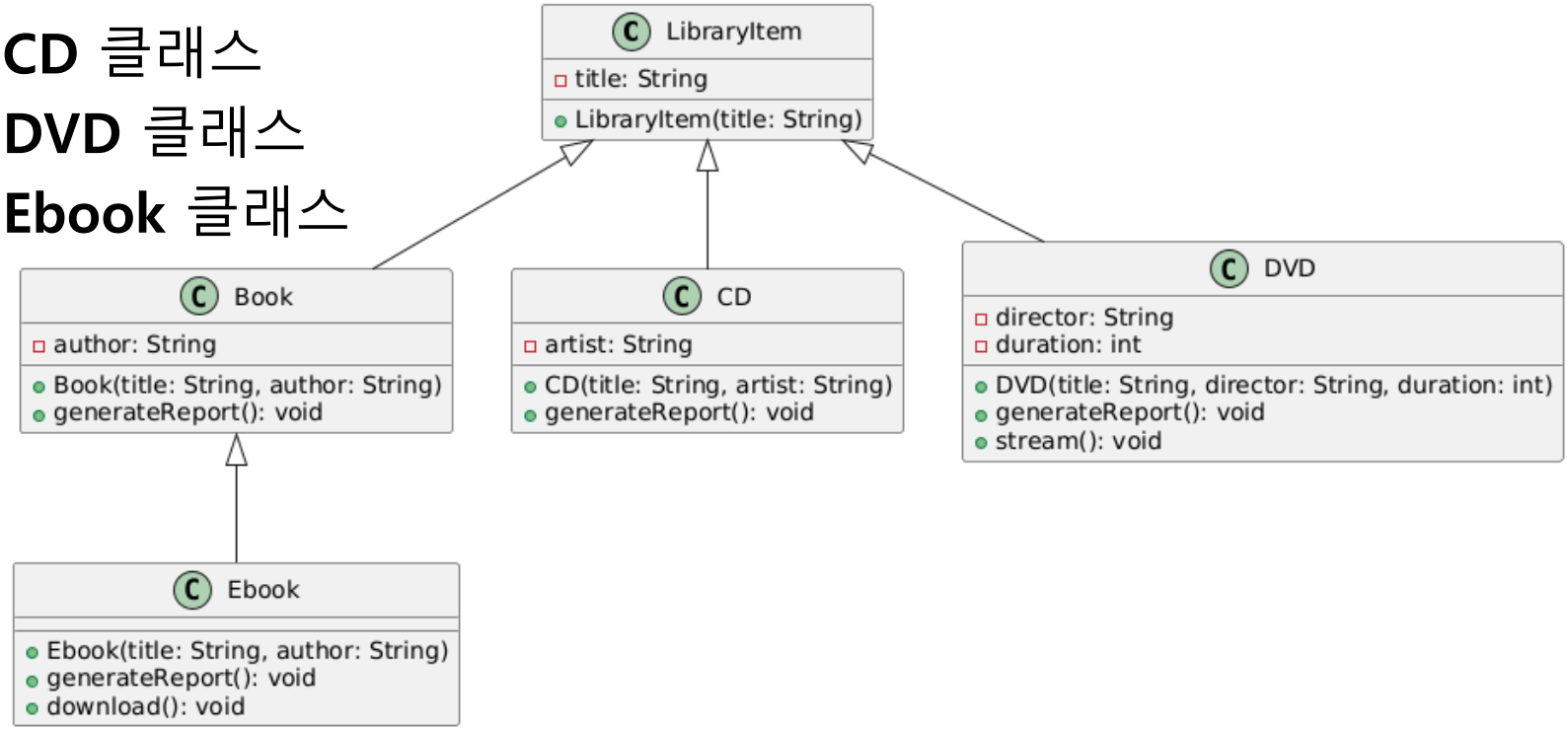
Lab1

- **LibraryItem** 클래스는 각각 **SRP**를 갖는 여러 클래스로 분할된다.
  - **LibraryItem** 클래스는 항목 기본 데이터를 처리한다.
  - **CheckoutManager** 클래스는 LibraryItem의 체크아웃 상태를 관리한다.
  - **ReportGenerator** 인터페이스는 LibraryItem에 대한 보고서를 생성한다.



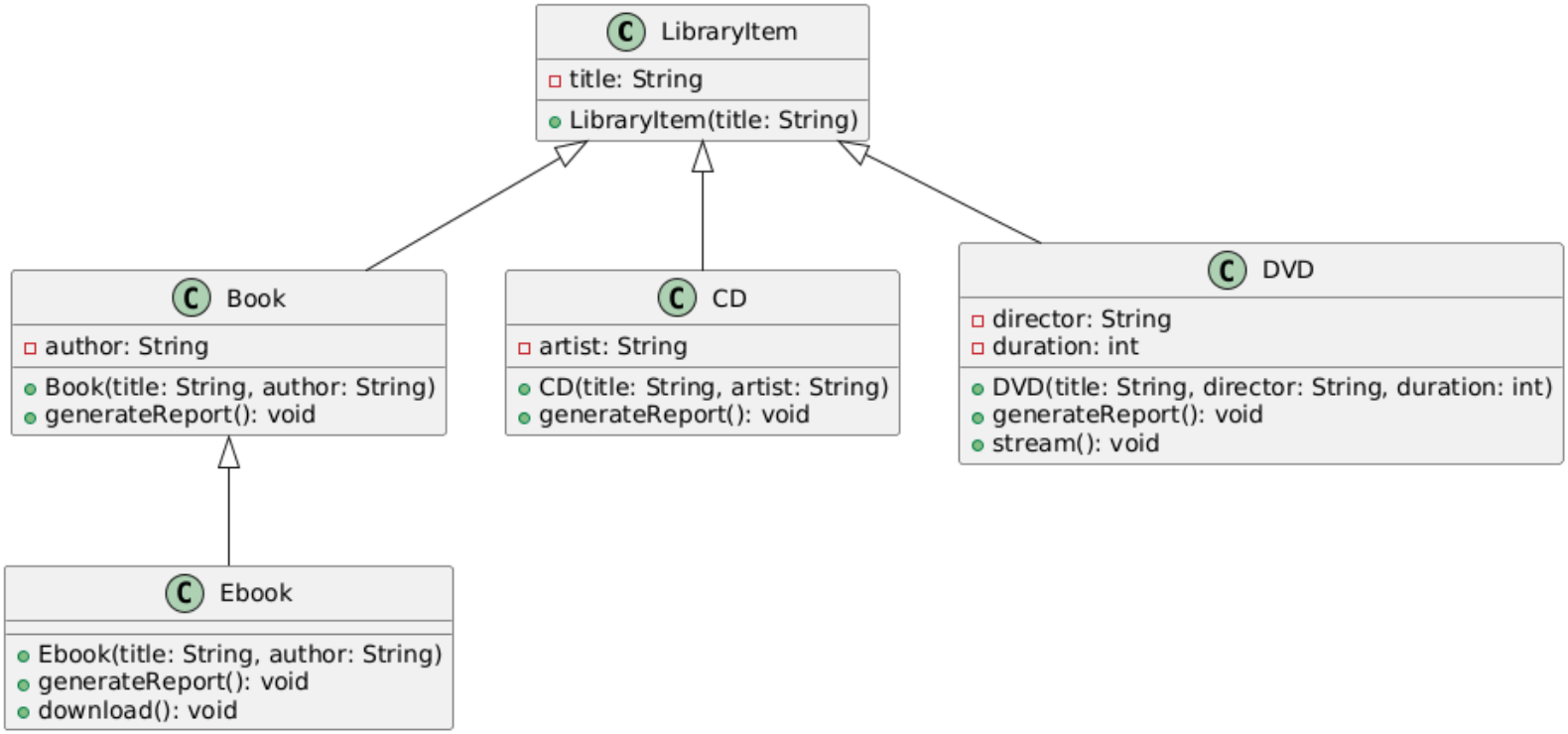
Lab1

- **LibraryItem** 클래스는 **OCP**를 기반으로 각 미디어 유형(Book, CD, DVD, Ebook)에 대한 특정 하위 클래스가 있는 기본 클래스로 리팩토링되어 기존 코드를 변경하지 않고도 새로운 유형의 미디어를 쉽게 추가할 수 있다.
  - **Book** 클래스
  - **CD** 클래스
  - **DVD** 클래스
  - **Ebook** 클래스



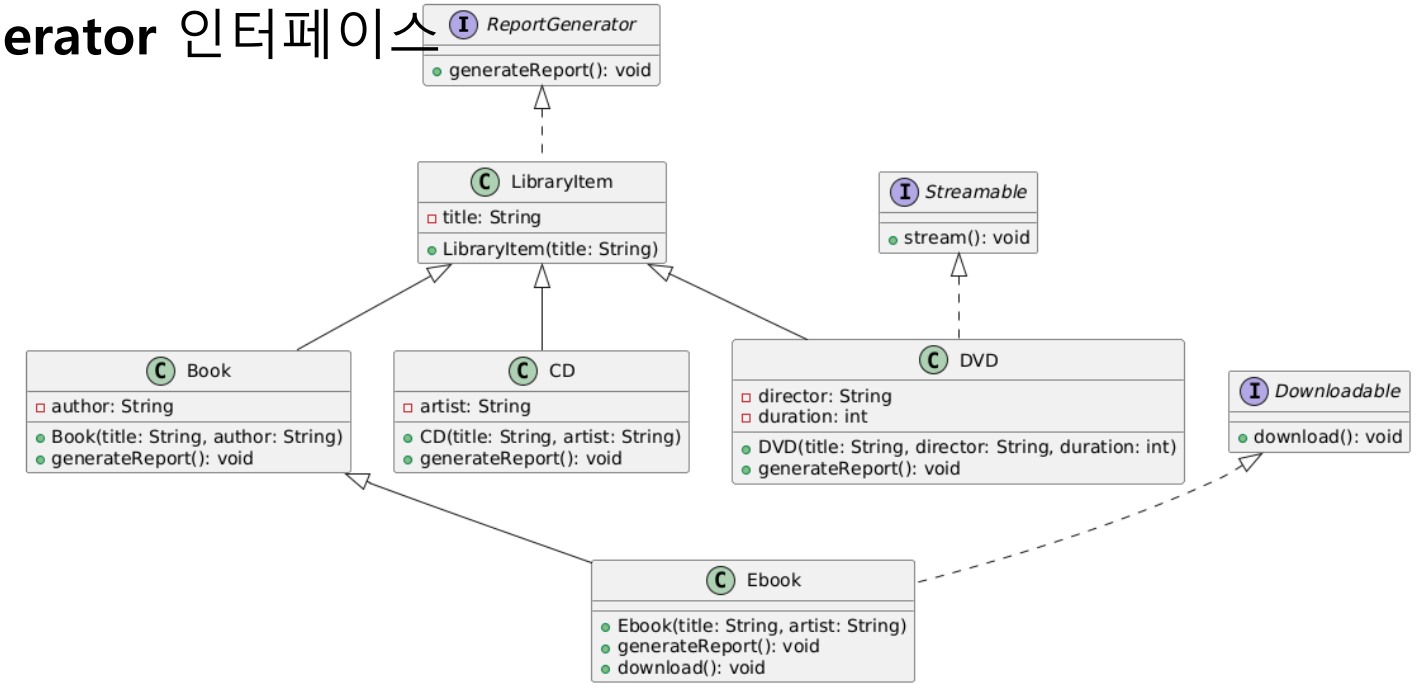
Lab1

- **LSP**를 적용하여 LibraryItem의 각 하위 클래스(예: Book, CD, DVD, Ebook)는 프로그램의 동작을 변경하지 않고도 LibraryItem 기본 클래스를 대체할 수 있도록 설계된다.



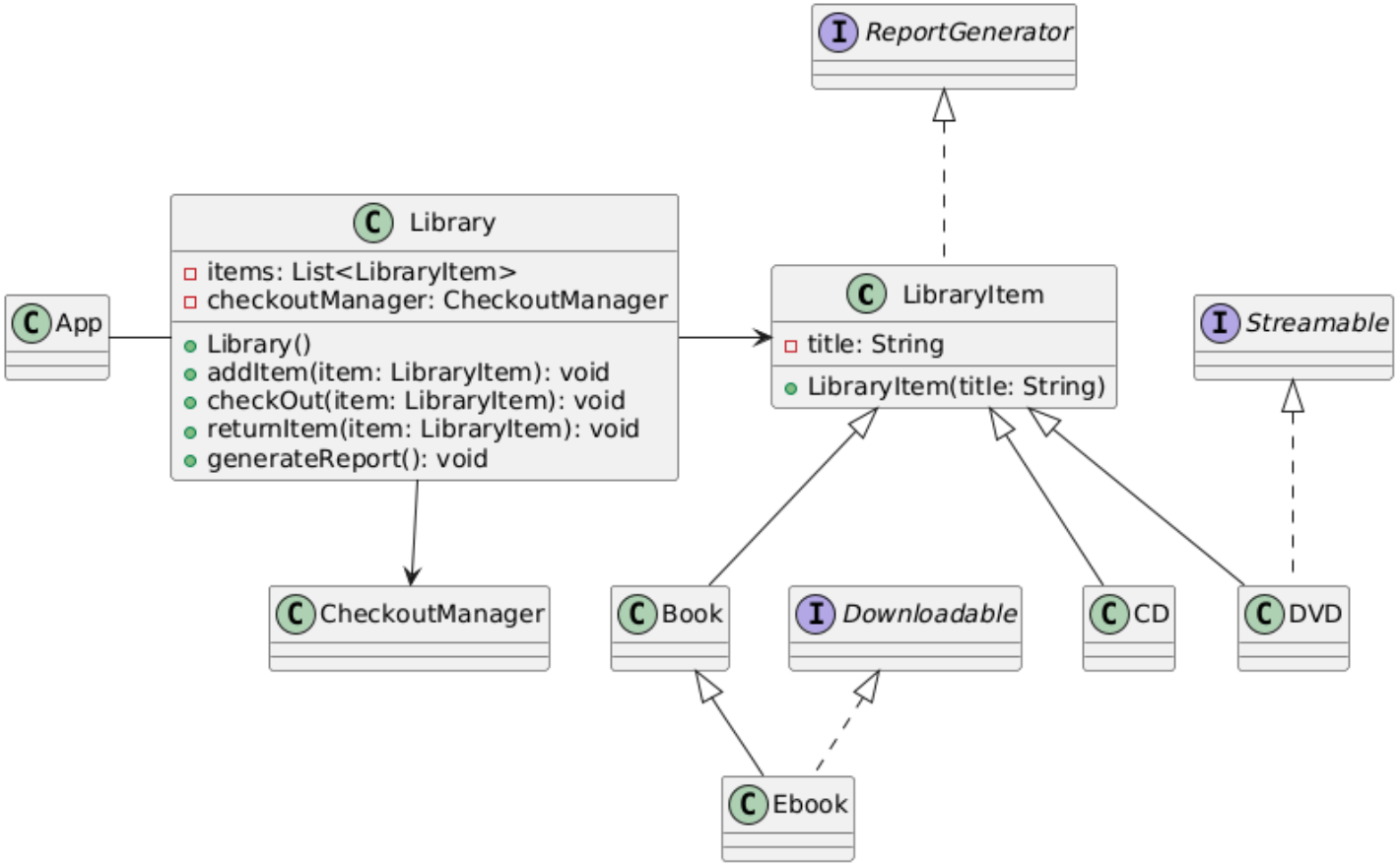
Lab1

- 다양한 유형의 작업에 대해 별도의 인터페이스를 정의하도록 리팩토링하여 (**ISP**), 클래스가 필요한 메서드만 구현하도록 하고, 불필요한 메서드를 구현하지 못하도록 한다.
  - **Downloadable** 인터페이스
  - **Streamable** 인터페이스
  - **ReportGenerator** 인터페이스



Lab1

- **DIP**를 적용하여, Library 클래스는 더 이상 구체적 클래스에 직접 의존하지 않는다. 대신 추상 인터페이스나 기본 클래스에 의존하게 된다.



Submit to e-learning

- Lab1 과제에 yourcode (e.g.: 한 가지 이상의 SOLID 원칙을 적용한 다른 예)를 추가 (yourcode 없을시 10점에서 -1점 감점)
- **Java24-2-HW1-YourID-YourName.zip** 과제(보고서에 반드시 yourcode 설명 포함)를 e러닝에 제출 **(due by 9/17)**.