

CodeForge - B01 - Abstract Class Cơ Bản

Độ khó: ★ Easy

Đề bài

Tạo abstract class đầu tiên:

- **Abstract** class `Shape` với:
 - Abstract method `abstract double getArea();` (no body)
- Class `Circle` extends `Shape` với:
 - `double radius`
 - Constructor nhận radius
 - **Implement** `getArea()` return $\pi * r^2$

Trong `main()`:

1. **KHÔNG THỂ** tạo `new Shape()` (abstract class)
2. Tạo `Circle` object
3. Gọi `getArea()`

◇ Input

- Một số thực radius

◇ Output

- Area (2 chữ số)

◇ Constraints

- $0 < radius \leq 100$

Ví dụ

Test case 1

Input:

5.0

Output:

78.54

Test case 2

Input:

3.0

Output:

28.27

Tags: `abstract`, `class`, `method`, `basic`, `cannot-instantiate`

CodeForge - B02 - Abstract Method Phải Được Implement

Độ khó: ★ Easy

Đề bài

Tạo hierarchy:

- Abstract class `Animal` với:
 - Abstract method `abstract void sound();`
- Class `Dog` extends `Animal` với:
 - **PHẢI** implement `sound()` in "Woof"
- Class `Cat` extends `Animal` với:
 - **PHẢI** implement `sound()` in "Meow"

Lưu ý: Nếu không implement → compile error

◇ Input

- Một dòng: "D" (Dog) hoặc "C" (Cat)

◇ Output

- Sound tương ứng

◇ Constraints

- Input chỉ là D hoặc C

Ví dụ

Test case 1

Input:

D

Output:

Woof

Test case 2

Input:

C

Output:

Meow

Tags: `abstract`, `method`, `must-implement`, `polymorphism`

CodeForge - B03 - Multiple Abstract Methods

Độ khó: ★ Easy

Đề bài

Tạo abstract class với nhiều abstract methods:

- Abstract class **Vehicle** với:
 - `abstract void start();`
 - `abstract void stop();`
 - `abstract double getFuelEfficiency();`
- Class **Car** extends Vehicle với:
 - Implement tất cả 3 methods

◇ Input

- Không có input

◇ Output

- 3 dòng từ 3 method calls

◇ Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Car starting
Car stopping
15.50
```

Tags: `abstract`, `multiple-methods`, `implementation`

CodeForge - B04 - Abstract Class Với Fields

Độ khó: ★ Easy

Đề bài

Abstract class có thể có fields:

- Abstract class `Employee` với:
 - `protected String name`
 - `protected double baseSalary`
 - `abstract double calculateSalary();`
- Class `Manager` extends `Employee` với:
 - `private double bonus`
 - Constructor nhận name, baseSalary, bonus
 - Implement `calculateSalary()` return `baseSalary + bonus`

◇ Input

- Dòng 1: Name
- Dòng 2: Base salary
- Dòng 3: Bonus

◇ Output

- Dòng 1: Name
- Dòng 2: Total salary

◇ Constraints

- $0 < \text{salary}, \text{bonus} \leq 1000000$

Ví dụ

Test case 1

Input:

```
Alice
50000.00
10000.00
```

Output:

```
Alice  
60000.00
```

Test case 2

Input:

```
Bob  
75000.00  
15000.00
```

Output:

```
Bob  
90000.00
```

Tags: `abstract`, `fields`, `protected`, `inheritance`

CodeForge - B05 - Polymorphism Với Abstract Class

Độ khó: ★ ★ Medium

Đề bài

Abstract class làm parent reference:

- Abstract class `Shape` với:
 - `abstract double getArea();`
 - `abstract double getPerimeter();`
- Classes `Rectangle`, `Circle`, `Triangle` extends `Shape`

Trong main():

1. Tạo `Shape[] shapes` với mixed concrete types
2. Call methods polymorphically

◇ Input

- Dòng 1: `N`
- `N` dòng: Shape type và dimensions

◇ Output

- `N` nhóm 2 dòng: area, perimeter

◇ Constraints

- $1 \leq N \leq 10$
- $0 < dimensions \leq 100$

Ví dụ

Test case 1

Input:

```
3
R 5.0 3.0
C 4.0
T 3.0 4.0 5.0
```

Output:

```
15.00  
16.00  
50.27  
25.13  
6.00  
12.00
```

Tags: `abstract`, `polymorphism`, `array`, `shapes`

CodeForge - B06 - Abstract Class Cannot Be Instantiated

Độ khó: ★ Easy

Đề bài

Demo rằng abstract class không thể instantiate:

- Abstract class `Database` với:
 - `abstract void connect();`
 - `abstract void disconnect();`
- Class `MySQLDatabase` extends `Database`

Trong main():

1. TRY: `new Database()` → compile error
2. OK: `Database db = new MySQLDatabase();` → works
3. Call `connect()`

◇ Input

- Không có input

◇ Output

- "MySQL connected"

◇ Constraints

- N/A

Ví dụ

Test case 1

Output:

```
MySQL connected
```

Tags: `abstract`, `cannot-instantiate`, `error`, `demo`

CodeForge - B07 - Concrete Methods Trong Abstract Class

Độ khó: ★ ★ Medium

Đề bài

Abstract class có thể có concrete methods:

- Abstract class `BankAccount` với:
 - `protected double balance`
 - **Concrete** method `void deposit(double amount)` (có body)
 - **Abstract** method `abstract void withdraw(double amount);`
- Classes `SavingsAccount`, `CheckingAccount` extends `BankAccount`
 - Implement `withdraw()` với logic khác nhau

◇ Input

- Dòng 1: Account type ("S" hoặc "C")
- Dòng 2: Deposit amount
- Dòng 3: Withdraw amount

◇ Output

- Balance cuối cùng

◇ Constraints

- $0 < amounts \leq 100000$

Ví dụ

Test case 1

Input:

```
S
10000.00
2000.00
```

Output:

```
8000.00
```

Test case 2

Input:

```
C  
5000.00  
6000.00
```

Output:

```
-1000.00
```

Giải thích: Checking cho phép overdraft

Tags: `abstract`, `concrete-method`, `partial-implementation`

CodeForge - B08 - Abstract Class Với Constructor

Độ khó: ★ ★ Medium

Đề bài

Abstract class có thể có constructor:

- Abstract class **Person** với:
 - `protected String name`
 - `protected int age`
 - **Constructor** nhận name, age (initialize fields)
 - `abstract void displayRole();`
- Classes **Student**, **Teacher** extends Person
 - Constructor gọi `super(name, age)`
 - Implement `displayRole()`

◇ Input

- Dòng 1: Type ("S" hoặc "T")
- Dòng 2: Name
- Dòng 3: Age

◇ Output

- Name, age, role

◇ Constraints

- $0 \leq \text{age} \leq 100$

Ví dụ

Test case 1

Input:

```
S
Alice
20
```

Output:

```
Alice
20
```

```
Student
```

Test case 2

Input:

```
T
Dr. Smith
45
```

Output:

```
Dr. Smith
45
Teacher
```

Tags: `abstract`, `constructor`, `super`, `initialization`

CodeForge - B09 - Abstract Class Với Static Members

Độ khó: ★ ★ Medium

Đề bài

Abstract class có thể có static members:

- Abstract class `Counter` với:
 - `protected static int count = 0`
 - Constructor tăng count
 - `static int getCount()` return count
 - `abstract void process();`
- Classes `TypeA`, `TypeB` extends `Counter`

Trong main():

1. Tạo nhiều objects từ subclasses
2. count tăng cho tất cả
3. Display total count

◇ Input

- Dòng 1: N objects
- N dòng: Type ("A" hoặc "B")

◇ Output

- Total count

◇ Constraints

- $1 \leq N \leq 100$

Ví dụ

Test case 1

Input:

```
5
A
B
A
B
A
```

Output:

5

Tags: `abstract`, `static`, `shared-state`, `counter`

CodeForge - B10 - Mix Abstract & Concrete Methods

Độ khó: ★ ★ Medium

Đề bài

Abstract class với mix của abstract và concrete methods:

- Abstract class **Game** với:
 - **Concrete** `void start()` in "Game starting"
 - **Abstract** `abstract void play();`
 - **Concrete** `void end()` in "Game ending"
- Classes **Chess**, **Soccer** extends Game
 - Implement `play()` only

Trong `main()`:

1. Tạo Game object (concrete subclass)
2. Call `start()` → inherited
3. Call `play()` → implemented
4. Call `end()` → inherited

◇ Input

- Một dòng: Game type ("C" hoặc "S")

◇ Output

- 3 dòng: start, play, end

◇ Constraints

- Input chỉ là C hoặc S

Ví dụ

Test case 1

Input:

```
C
```

Output:

```
Game starting
Playing chess
```

```
Game ending
```

Test case 2

Input:

```
S
```

Output:

```
Game starting  
Playing soccer  
Game ending
```

Tags: `abstract`, `concrete`, `mix`, `inheritance`

CodeForge - B11 - Abstract Class Hierarchy

Độ khó: ★ ★ ★ Hard

Đề bài

Abstract class có thể extend abstract class khác:

- Abstract class **A** với `abstract void methodA();`
- Abstract class **B** extends A với:
 - Implement methodA()
 - Add `abstract void methodB();`
- Class **C** extends B với:
 - Implement methodB()

Trong main():

1. Tạo object C
2. Call cả 2 methods

◇ Input

- Không có input

◇ Output

- 2 dòng từ 2 methods

◇ Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Method A implemented
Method B implemented
```

Tags: `abstract`, `hierarchy`, `chain`, `inheritance`

CodeForge - B12 - Shared Implementation Trong Abstract Class

Độ khó: ★ ★ Medium

Đề bài

Abstract class cung cấp shared implementation:

- Abstract class **Logger** với:
 - **Concrete** `void log(String message)`:
 - In timestamp + message
 - Gọi `writeToDestination(message)` (abstract)
 - **Abstract** `abstract void writeToDestination(String msg);`
- Classes **FileLogger**, **ConsoleLogger**, **DatabaseLogger** extends **Logger**
 - Implement `writeToDestination()` khác nhau

◇ Input

- Dòng 1: Logger type ("F", "C", hoặc "D")
- Dòng 2: Message

◇ Output

- Timestamp + message + destination

◇ Constraints

- Độ dài message ≤ 200

Ví dụ

Test case 1

Input:

```
F
Error occurred
```

Output:

```
[2024-12-22 14:30:00] Error occurred
Written to file
```

Test case 2

Input:

```
C  
Warning message
```

Output:

```
[2024-12-22 14:30:00] Warning message  
Written to console
```

Tags: `abstract`, `shared-implementation`, `logger`, `pattern`

CodeForge - B13 - Template Method Pattern Cơ Bản

Độ khó: ★ ★ Medium

Đề bài

Template Method pattern với abstract class:

- Abstract class `DataProcessor` với:
 - **Concrete** `final void process()`:
 1. `readData()` (abstract)
 2. `processData()` (abstract)
 3. `writeData()` (abstract)
 - Template method định nghĩa algorithm structure
- Classes `CSVProcessor`, `JSONProcessor` extends `DataProcessor`
 - Implement 3 abstract methods

◇ Input

- Một dòng: Processor type ("CSV" hoặc "JSON")

◇ Output

- 3 dòng: read, process, write

◇ Constraints

- N/A

Ví dụ

Test case 1

Input:

```
CSV
```

Output:

```
Reading CSV data
Processing CSV data
Writing CSV data
```

Test case 2

Input:

JSON

Output:

Reading JSON data
Processing JSON data
Writing JSON data

Tags: `abstract`, `template-method`, `pattern`, `algorithm`

CodeForge - B14 - Template Method Với Hook Methods

Độ khó: ★ ★ ★ Hard

Đề bài

Template method với optional hooks:

- Abstract class **Report** với:
 - **Concrete** `final void generate()`:
 1. `printHeader()` (concrete)
 2. `printBody()` (abstract)
 3. `printFooter()` (concrete)
 4. if (`needsSignature()`) `printSignature()` (hook)
 - `boolean needsSignature()` return false (hook, có thể override)
- Classes **Invoice**, **Receipt** extends **Report**
 - **Invoice** override `needsSignature()` return true

◇ Input

- Một dòng: Report type ("I" hoặc "R")

◇ Output

- Report content với/không signature

◇ Constraints

- N/A

Ví dụ

Test case 1

Input:

```
I
```

Output:

```
=== HEADER ===
Invoice body content
=== FOOTER ===
--- SIGNATURE ---
```

Test case 2

Input:

R

Output:

```
=== HEADER ===
Receipt body content
=== FOOTER ===
```

Tags: `abstract`, `template-method`, `hook`, `optional`

CodeForge - B15 - Template Method Với Validation

Độ khó: ★ ★ ★ Hard

Đề bài

Template method với validation steps:

- Abstract class `Transaction` với:
 - **Concrete** `final boolean execute()`:
 1. if (!validate()) return false
 2. if (!checkBalance()) return false
 3. processTransaction() (abstract)
 4. recordTransaction() (abstract)
 5. return true
 - `abstract boolean validate();`
 - `abstract boolean checkBalance();`
- Classes `Payment`, `Transfer`, `Withdrawal` extends `Transaction`

◇ Input

- Dòng 1: Transaction type
- Dòng 2+: Transaction data

◇ Output

- Success/failure message

◇ Constraints

- N/A

Ví dụ

Test case 1

Input:

```
PAYMENT
1000.00
```

Output:

```
Validation passed
Balance check passed
```

```
Payment processed: $1000.00  
Transaction recorded  
Success
```

Tags: `abstract`, `template-method`, `validation`, `business-logic`

CodeForge - B16 - Abstract Class Định Nghĩa Contract

Độ khó: ★ ★ Medium

Đề bài

Abstract class định nghĩa contract cho subclasses:

- Abstract class `Sorter` với:
 - `abstract void sort(int[] arr);`
 - `abstract String getAlgorithmName();`
 - **Concrete** `void displayInfo()` gọi `getAlgorithmName()`
- Classes `BubbleSort`, `QuickSort`, `MergeSort` extends `Sorter`
 - Implement cả 2 abstract methods

◇ Input

- Dòng 1: Algorithm ("B", "Q", hoặc "M")
- Dòng 2: N
- Dòng 3: N numbers

◇ Output

- Algorithm name
- Sorted array

◇ Constraints

- $1 \leq N \leq 100$

Ví dụ

Test case 1

Input:

```
B
5
5 2 8 1 9
```

Output:

```
Bubble Sort
1 2 5 8 9
```

Tags: abstract, contract, sorter, algorithm

CodeForge - B17 - When To Use Abstract Class

Độ khó: ★ ★ Medium

Đề bài

Demo khi nào nên dùng abstract class:

- **Use case:** Shared code + abstract behavior
- Abstract class **Vehicle** với:
 - **Shared** fields: brand, year
 - **Shared** method: displayInfo()
 - **Abstract** method: calculateTax() (different for each type)
- Classes **Car**, **Motorcycle**, **Truck** extends Vehicle
 - Reuse shared code
 - Implement specific behavior

◇ Input

- Dòng 1: Vehicle type
- Dòng 2-3: Brand, year

◇ Output

- Info + tax

◇ Constraints

- $1900 \leq \text{year} \leq 2100$

Ví dụ

Test case 1

Input:

```
CAR
Toyota
2020
```

Output:

```
Toyota 2020
Tax: $1500.00
```

Tags: abstract, when-to-use, design, decision

CodeForge - B18 - Abstract Class Vs Concrete Class

Độ khó: ★ ★ ★ Hard

Đề bài

So sánh abstract vs concrete class:

- **Concrete** class `BasicCalculator` với:
 - Full implementation
 - Can instantiate
- **Abstract** class `AdvancedCalculator` với:
 - Partial implementation
 - Abstract methods cho advanced operations
 - Cannot instantiate

Demo khi nào dùng cái nào.

◇ Input

- Dòng 1: Calculator type ("B" hoặc "A")
- Dòng 2-3: Two numbers

◇ Output

- Calculation result

◇ Constraints

- $-1000 \leq \text{numbers} \leq 1000$

Ví dụ

Test case 1

Input:

```
B
10
5
```

Output:

```
Basic: 10 + 5 = 15
```

Test case 2

Input:

```
A
10
5
```

Output:

```
Advanced: 10 ^ 5 = 100000
```

Tags: `abstract`, `concrete`, `comparison`, `design`

CodeForge - B19 - Partial Implementation Strategy

Độ khó: ★ ★ ★ Hard

Đề bài

Abstract class với partial implementation:

- Abstract class `WebCrawler` với:
 - **Concrete** `void crawl(String url)`:
 - `fetchPage(url)` (concrete - HTTP logic)
 - `parsePage()` (abstract - different for each type)
 - `storeData()` (abstract - different storage)
 - Shared HTTP fetching
 - Abstract parsing & storage
- Classes `NewsCrawler`, `ProductCrawler` extends `WebCrawler`

◇ Input

- Dòng 1: Crawler type
- Dòng 2: URL

◇ Output

- Crawling log

◇ Constraints

- Độ dài URL ≤ 200

Ví dụ

Test case 1

Input:

```
NEWS
https://example.com
```

Output:

```
Fetching: https://example.com
Parsing news article
Storing in news database
```

Tags: abstract, partial-implementation, template, crawler

CodeForge - B20A - Drawing Application Framework

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo drawing framework với abstract class:

- Abstract class `Shape` với:
 - `protected int x, y` (position)
 - `protected String color`
 - Constructor initialize position & color
 - **Abstract** `abstract double getArea();`
 - **Abstract** `abstract double getPerimeter();`
 - **Concrete** `void draw()` in "Drawing [shape] at ([x],[y]) in [color]"
 - **Concrete** `void move(int newX, int newY)` update position
- Classes `Circle`, `Rectangle`, `Triangle` extends `Shape`
 - Specific fields (radius, width/height, sides)
 - Implement `getArea()` và `getPerimeter()`

Trong `main()`:

1. Tạo N shapes
2. Draw all shapes
3. Calculate total area
4. Move shapes và redraw

◇ Input

- Dòng 1: N
- N dòng: Shape data (type, position, color, dimensions)
- Dòng N+2: M (số move operations)
- M dòng: Shape index, newX, newY

◇ Output

- Initial drawings
- Total area
- After move drawings

◇ Constraints

- $1 \leq N \leq 20$
- $-100 \leq x, y \leq 100$

Ví dụ

Test case 1

Input:

```
3
CIRCLE 0 0 Red 5.0
RECT 10 10 Blue 4.0 6.0
TRIANGLE 5 5 Green 3.0 4.0 5.0
2
0 10 10
2 0 0
```

Output:

```
Drawing Circle at (0,0) in Red
Drawing Rectangle at (10,10) in Blue
Drawing Triangle at (5,5) in Green
Total Area: 115.54
```

After moves:

```
Drawing Circle at (10,10) in Red
Drawing Rectangle at (10,10) in Blue
Drawing Triangle at (0,0) in Green
```

Tags: [abstract](#), [framework](#), [drawing](#), [shapes](#), [advanced](#)

CodeForge - B21A - Database Connection Framework

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo database framework:

- Abstract class `DatabaseConnection` với:
 - `protected String host, username, password`
 - Constructor
 - **Template method** `final void connect()`:
 1. `validateCredentials()` (concrete - common logic)
 2. `establishConnection()` (abstract)
 3. `configureConnection()` (abstract)
 4. `testConnection()` (concrete - ping)
 - **Concrete** `void disconnect()` shared logic
 - **Abstract** methods cho specific DB
- Classes `MySQLConnection`, `PostgreSQLConnection`, `MongoDBConnection` extends `DatabaseConnection`
 - Implement abstract methods với specific protocols

Trong main():

1. Create connection pool (array of different DB types)
2. Connect all
3. Execute query polymorphically
4. Disconnect all

◇ Input

- Dòng 1: N (connections)
- N dòng: DB type và credentials

◇ Output

- Connection log cho mỗi database

◇ Constraints

- $1 \leq N \leq 10$

Ví dụ

Test case 1

Input:

```
3
MYSQL localhost root pass123
POSTGRES 192.168.1.1 admin admin123
MONGO cloud.mongo.com user mongo123
```

Output:

```
[MySQL] Validating credentials...
[MySQL] Establishing connection to localhost
[MySQL] Configuring MySQL-specific settings
[MySQL] Connection test: OK

[PostgreSQL] Validating credentials...
[PostgreSQL] Establishing connection to 192.168.1.1
[PostgreSQL] Configuring PostgreSQL-specific settings
[PostgreSQL] Connection test: OK

[MongoDB] Validating credentials...
[MongoDB] Establishing connection to cloud.mongo.com
[MongoDB] Configuring MongoDB-specific settings
[MongoDB] Connection test: OK
```

Tags: [abstract](#), [database](#), [framework](#), [template-method](#), [advanced](#)

CodeForge - B22A - HTTP Request Handler Framework

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo HTTP handler framework:

- Abstract class `RequestHandler` với:
 - **Template method** `final void handle(String request)`:
 1. `parseRequest(request)` (concrete)
 2. `authenticate()` (abstract - different auth methods)
 3. `authorize()` (abstract - different permissions)
 4. `processRequest()` (abstract - business logic)
 5. `formatResponse()` (concrete)
 - Shared parsing & formatting
 - Abstract authentication & processing
- Classes `AdminHandler`, `UserHandler`, `GuestHandler` extends `RequestHandler`
 - Different authentication levels
 - Different allowed operations

Trong `main()`:

1. Receive N requests
2. Route to appropriate handler
3. Process với different auth/authorization

◇ Input

- Dòng 1: N
- N dòng: Request type (ADMIN/USER/GUEST) và request data

◇ Output

- Processing log cho mỗi request

◇ Constraints

- $1 \leq N \leq 50$

Ví dụ

Test case 1

Input:

```
3
ADMIN DELETE_USER user123
USER GET_PROFILE
GUEST VIEW_PUBLIC
```

Output:

```
Parsing request: DELETE_USER user123
Admin authentication: SUCCESS
Admin authorization: FULL_ACCESS
Processing: User deleted
Response: 200 OK
```

```
Parsing request: GET_PROFILE
User authentication: SUCCESS
User authorization: READ_WRITE
Processing: Profile retrieved
Response: 200 OK
```

```
Parsing request: VIEW_PUBLIC
Guest authentication: SUCCESS
Guest authorization: READ_ONLY
Processing: Public content shown
Response: 200 OK
```

Tags: [abstract](#), [http](#), [handler](#), [authentication](#), [advanced](#)

CodeForge - B23A - Game Character AI Framework

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo game AI framework:

- Abstract class `CharacterAI` với:
 - `protected int aggressiveness` (0-100)
 - **Template method** `final void takeTurn()`:
 1. `assess()` (concrete - common analysis)
 2. `chooseAction()` (abstract - AI decision)
 3. `executeAction()` (abstract - specific to character)
 4. `updateState()` (concrete - shared state management)
 - **Concrete** helper methods
 - **Abstract** decision-making
- Classes `WarriorAI`, `MageAI`, `ArcherAI` extends `CharacterAI`
 - Different decision strategies
 - Different action sets

Trong `main()`:

1. Create AI team
2. Simulate N turns
3. Each AI makes decisions independently

◇ Input

- Dòng 1: Team size
- Team data (type, aggressiveness)
- Dòng 3: Number of turns

◇ Output

- Turn-by-turn AI decisions

◇ Constraints

- $1 \leq \text{team size} \leq 5$
- $1 \leq \text{turns} \leq 10$

Ví dụ

Test case 1

Input:

```
3
WARRIOR 80
MAGE 30
ARCHER 60
3
```

Output:

```
Turn 1:
Warrior: Assessing situation -> Attack selected -> Charge executed
Mage: Assessing situation -> Cast spell selected -> Fireball executed
Archer: Assessing situation -> Ranged attack selected -> Arrow shot executed

Turn 2:
...
```

Tags: `abstract`, `game`, `ai`, `decision-making`, `advanced`

CodeForge - B24A - Document Generator Framework

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo document generator:

- Abstract class `DocumentGenerator` với:
 - **Template method** `final String generate()`:
 1. `StringBuilder sb = new StringBuilder()`
 2. `sb.append(createHeader())` (abstract)
 3. `sb.append(createBody())` (abstract)
 4. `sb.append(createFooter())` (abstract)
 5. `return formatDocument(sb.toString())` (concrete)
 - **Concrete** `String formatDocument()` common formatting
- Classes `PDFGenerator`, `HTMLGenerator`, `MarkdownGenerator` extends `DocumentGenerator`
 - Different markup syntax
 - Different structure

Trong `main()`:

1. Input document content
2. Generate in N formats
3. Display all outputs

◇ Input

- Dòng 1: Title
- Dòng 2: Body content
- Dòng 3: N (formats)
- N dòng: Format types

◇ Output

- N documents trong different formats

◇ Constraints

- $1 \leq N \leq 3$

Ví dụ

Test case 1

Input:

```
Java Tutorial
This is a Java programming guide
2
PDF
HTML
```

Output:

```
=== PDF ===
%PDF-1.4
[Header: Java Tutorial]
[Body: This is a Java programming guide]
[Footer: Generated 2024-12-22]

=== HTML ===
<!DOCTYPE html>
<h1>Java Tutorial</h1>
<p>This is a Java programming guide</p>
<footer>Generated 2024-12-22</footer>
```

Tags: [abstract](#), [document](#), [generator](#), [template](#), [advanced](#)

CodeForge - B25A - Complete Abstract System - E-Learning Platform

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo complete e-learning platform:

- Abstract class **Course** với:
 - `protected String title, instructor`
 - `protected int enrolledStudents`
 - `protected ArrayList<String> lessons`
 - **Template method** `final void conductCourse()`:
 1. `displayInfo()` (concrete)
 2. `checkPrerequisites()` (abstract - different for each type)
 3. `deliverContent()` (abstract - video/live/text)
 4. `assessStudents()` (abstract - quiz/project/exam)
 5. `issueCertificate()` (concrete - shared)
 - **Abstract** pricing strategy
- Classes **VideoCourse**, **LiveCourse**, **TextCourse** extends **Course**
 - Different content delivery
 - Different assessment methods
 - Different prerequisites
- Abstract class **Assessment** với template cho testing
- Classes **Quiz**, **Project**, **Exam** extends **Assessment**

Trong `main()`:

1. Create course catalog (polymorphic collection)
2. Enroll students
3. Conduct courses
4. Generate reports

◇ Input

- Dòng 1: N (courses)
- N nhóm: Course data
- Dòng X: M (enrollments)
- M dòng: Student enrollments

◇ Output

- Course execution logs
- Certificate issuance
- Final statistics

◇ Constraints

- $1 \leq N \leq 20$
- $1 \leq M \leq 100$

Ví dụ

Test case 1

Input:

```
3
VIDEO "Java Basics" "John Doe" 10
LIVE "Advanced OOP" "Jane Smith" 5
TEXT "Design Patterns" "Bob Johnson" 8
5
0 Alice
0 Bob
1 Charlie
2 David
2 Eve
```

Output:

```
=== Java Basics (Video Course) ===
Instructor: John Doe
Enrolled: 2 students
Checking prerequisites: None required
Delivering content: 10 video lessons
Assessment: Online quiz
Certificates issued to: Alice, Bob

=== Advanced OOP (Live Course) ===
Instructor: Jane Smith
Enrolled: 1 student
Checking prerequisites: Java Basics required
Delivering content: 5 live sessions
Assessment: Final project
Certificates issued to: Charlie

=== Design Patterns (Text Course) ===
Instructor: Bob Johnson
Enrolled: 2 students
Checking prerequisites: Advanced OOP required
Delivering content: 8 reading modules
Assessment: Written exam
```

Certificates issued to: David, Eve

Platform Statistics:

Total Courses: 3

Total Enrollments: 5

Certificates Issued: 5

Tags: abstract, elearning, complete-system, template-method, capstone, advanced