

# CODEFORGE - BUỔI 13: INHERITANCE

## 25 BÀI TẬP (18 CORE + 7 ADVANCED)

### Kiến thức Buổi 13:

- Inheritance Basics (extends keyword, is-a relationship, Parent/Child class, Single inheritance)
- Constructor in Inheritance (super() call, Implicit super(), Constructor chaining, Initialization order)
- Method Overriding (@Override annotation, Rules for overriding, super.method() call, Overriding vs Overloading)
- final Keyword (final variables, final methods - cannot override, final classes - cannot extend)
- protected Access Modifier

## CodeForge - B01 - Hệ Thống Quản Lý Nhân Viên

**Độ khó:** ★ ★ ★ Hard

**Loại:** ☑ Competitive

**Kiến thức:** Inheritance, Constructor Chaining, Method Overriding

### Đề bài

Công ty ABC cần xây dựng hệ thống quản lý 2 loại nhân viên:

#### 1. Nhân viên thường (Employee):

- Có mã nhân viên, tên, lương cơ bản
- Lương thực nhận = lương cơ bản

#### 2. Quản lý (Manager):

- Có đầy đủ thông tin như Employee
- Thêm thông tin số lượng nhân viên quản lý (team size)
- Lương thực nhận = lương cơ bản + thưởng quản lý
- Thưởng quản lý = số lượng nhân viên × 2,000,000đ

**Yêu cầu:** Viết chương trình nhập thông tin N nhân viên (gồm cả Employee và Manager), sau đó:

1. In thông tin chi tiết từng nhân viên
2. Tính tổng lương công ty phải trả
3. Thống kê số lượng từng loại nhân viên và tổng lương từng loại

#### ◊ Input

```
Dòng 1: N (số lượng nhân viên, 1 ≤ N ≤ 100)
N dòng tiếp theo, mỗi dòng có 2 định dạng:
```

Định dạng 1 - Employee:  
 E <id> <name> <baseSalary>

Định dạng 2 - Manager:  
 M <id> <name> <baseSalary> <teamSize>

### Trong đó:

- E hoặc M: Loại nhân viên
- id: Mã nhân viên (chuỗi không có khoảng trắng)
- name: Tên nhân viên (chuỗi không có khoảng trắng, có thể dùng \_ thay space)
- baseSalary: Lương cơ bản (số thực)
- teamSize: Số người quản lý (số nguyên, chỉ có ở Manager)

### ❖ Output

#### Phần 1: Thông tin từng nhân viên theo format:

```
==== NHAN VIEN <thu_tu> ====
Loai: <Employee|Manager>
Ma NV: <id>
Ten: <name>
Luong co ban: <baseSalary>d
[Team size: <teamSize>]           ← Chỉ có ở Manager
[Thuong quan ly: <bonus>d]       ← Chỉ có ở Manager
Luong thuc nhan: <actualSalary>d
```

#### Phần 2: Thống kê tổng kết:

```
==== THONG KE ====
Tong so nhan vien: <total>
- Employee: <empCount> nguoi - Tong luong: <empTotalSalary>d
- Manager: <mgrCount> nguoi - Tong luong: <mgrTotalSalary>d
TONG LUONG CONG TY: <companySalary>d
```

### Lưu ý:

- Số thực làm tròn 0 chữ số thập phân
- Format số: không có dấu phẩy ngăn cách hàng nghìn

### Ví dụ

#### Test case 1

#### Input:

```
5
E EMP001 Alice 15000000
M MGR001 Bob 25000000 5
E EMP002 Charlie 18000000
M MGR002 Diana 30000000 8
E EMP003 Eve 16000000
```

**Output:**

```
==== NHAN VIEN 1 ====
Loai: Employee
Ma NV: EMP001
Ten: Alice
Luong co ban: 15000000d
Luong thuc nhan: 15000000d

==== NHAN VIEN 2 ====
Loai: Manager
Ma NV: MGR001
Ten: Bob
Luong co ban: 25000000d
Team size: 5
Thuong quan ly: 10000000d
Luong thuc nhan: 35000000d

==== NHAN VIEN 3 ====
Loai: Employee
Ma NV: EMP002
Ten: Charlie
Luong co ban: 18000000d
Luong thuc nhan: 18000000d

==== NHAN VIEN 4 ====
Loai: Manager
Ma NV: MGR002
Ten: Diana
Luong co ban: 30000000d
Team size: 8
Thuong quan ly: 16000000d
Luong thuc nhan: 46000000d

==== NHAN VIEN 5 ====
Loai: Employee
Ma NV: EMP003
Ten: Eve
Luong co ban: 16000000d
Luong thuc nhan: 16000000d

==== THONG KE ====
Tong so nhan vien: 5
- Employee: 3 nguoi - Tong luong: 49000000d
```

```
- Manager: 2 nguoi - Tong luong: 81000000d  
TONG LUONG CONG TY: 130000000d
```

## 💡 Gợi ý

### Tư duy bài toán:

- Manager có quan hệ gì với Employee? (IS-A relationship?)
- Làm sao để tránh lặp code giữa 2 loại nhân viên?
- Method nào cần override để tính lương khác nhau?

### Kiến thức cần dùng:

- Inheritance (kế thừa)
- Constructor chaining với super()
- Method overriding
- Polymorphism (mảng chứa nhiều loại đối tượng)

**Tags:** inheritance, extends, super, override, polymorphism, competitive

---

# CodeForge - B02 - Hệ Thống Quản Lý Phương Tiện

**Độ khó:** ★ ★ Medium

**Loại:** ✅ Competitive

**Kiến thức:** Inheritance, Constructor, Method Overriding

## 📝 Đề bài

Công ty cho thuê xe cần quản lý 2 loại phương tiện:

### 1. Ô tô (Car):

- Có biển số, hãng, giá thuê/ngày, số chỗ ngồi
- Phí thuê = giá thuê × số ngày

### 2. Xe máy (Motorcycle):

- Có biển số, hãng, giá thuê/ngày, dung tích xi-lanh (cc)
- Phí thuê = giá thuê × số ngày
- Nếu dung tích ≥ 150cc: thêm phí bảo hiểm 100,000đ/ngày

**Yêu cầu:** Nhập thông tin N phương tiện và số ngày thuê, tính tổng doanh thu.

## ◊ Input

Dòng 1: N ( $1 \leq N \leq 100$ )

N nhóm, mỗi nhóm:

Nhóm 1 - Car:

C <plate> <brand> <pricePerDay> <seats> <days>

Nhóm 2 - Motorcycle:

M <plate> <brand> <pricePerDay> <cc> <days>

## Trong đó:

- C hoặc M: Loại xe
- plate: Biển số (không có khoảng trắng)
- brand: Hãng xe (không có khoảng trắng)
- pricePerDay: Giá thuê/ngày (số thực)
- seats: Số chỗ ngồi (số nguyên, chỉ có ở Car)
- cc: Dung tích xi-lanh (số nguyên, chỉ có ở Motorcycle)
- days: Số ngày thuê (số nguyên)

## ◊ Output

```

==== XE <thu_tu> ====
Loai: <Car|Motorcycle>
Bien so: <plate>
Hang: <brand>
Gia thue: <pricePerDay>d/ngay
[So cho: <seats>]           ← Chỉ có ở Car
[Dung tich: <cc>cc]          ← Chỉ có ở Motorcycle
[Phi bao hiem: <insurance>d/ngay] ← Chỉ có ở Motorcycle ≥150cc
So ngay thue: <days>
Tong phi: <totalFee>d

==== DOANH THU ====
Tong so xe: <total>
- Car: <carCount> xe - Doanh thu: <carRevenue>d
- Motorcycle: <motoCount> xe - Doanh thu: <motoRevenue>d
TONG DOANH THU: <totalRevenue>d

```

## Ví dụ

Test case 1

**Input:**

```

4
C 30A12345 Toyota 500000 7 3
M 29X98765 Honda 150000 125 5
C 51G67890 Ford 800000 4 2
M 50H11111 Yamaha 200000 155 4

```

**Output:**

```

==== XE 1 ====
Loai: Car
Bien so: 30A12345
Hang: Toyota
Gia thue: 500000d/ngay
So cho: 7
So ngay thue: 3
Tong phi: 1500000d

==== XE 2 ====
Loai: Motorcycle
Bien so: 29X98765
Hang: Honda
Gia thue: 150000d/ngay
Dung tich: 125cc
So ngay thue: 5
Tong phi: 750000d

```

```
==== XE 3 ====
Loai: Car
Bien so: 51G67890
Hang: Ford
Gia thue: 800000d/ngay
So cho: 4
So ngay thue: 2
Tong phi: 1600000d
```

```
==== XE 4 ====
Loai: Motorcycle
Bien so: 50H11111
Hang: Yamaha
Gia thue: 200000d/ngay
Dung tich: 155cc
Phi bao hiem: 100000d/ngay
So ngay thue: 4
Tong phi: 1200000d
```

```
==== DOANH THU ====
Tong so xe: 4
- Car: 2 xe - Doanh thu: 3100000d
- Motorcycle: 2 xe - Doanh thu: 1950000d
TONG DOANH THU: 5050000d
```

## 💡 Gợi ý

### Tư duy:

- Car và Motorcycle có thuộc tính chung gì? → Class cha Vehicle?
- Method tính phí thuê có cần override không?
- Motorcycle có logic đặc biệt (phí bảo hiểm) → Override calculateFee()

**Tags:** inheritance, extends, override, polymorphism, competitive

---

# CodeForge - B03 - Vườn Thú Ảo

**Độ khó:** ★ ★ Medium

**Loại:** ☑ Competitive

**Kiến thức:** Inheritance, Method Overriding

## 📝 Đề bài

Vườn thú cần quản lý động vật với 2 loại:

### 1. Chó (Dog):

- Có tên, tuổi, giống
- Tiếng kêu: "Gau gau!"
- Hành động: "Chạy quanh sân"

### 2. Mèo (Cat):

- Có tên, tuổi, màu lông
- Tiếng kêu: "Meo meo!"
- Hành động: "Leo cây"

**Yêu cầu:** Nhập thông tin N động vật, sau đó mô phỏng chúng kêu và hành động.

## ◊ Input

Dòng 1: N (1 ≤ N ≤ 50)

N dòng:

Dog:

D <name> <age> <breed>

Cat:

C <name> <age> <color>

## Trong đó:

- D hoặc C: Loại động vật
- name: Tên (không có khoảng trắng)
- age: Tuổi (số nguyên)
- breed: Giống chó (không có khoảng trắng, chỉ có ở Dog)
- color: Màu lông (không có khoảng trắng, chỉ có ở Cat)

## ◊ Output

Mỗi dòng vật từ gioi thieu va hoat dong:

```
<name> (<age> tuoi):  
- Loai: <Dog|Cat>  
[- Giong: <breed>] ← Chỉ có ở Dog  
[- Mau long: <color>] ← Chỉ có ở Cat  
- Tieng keu: <sound>  
- Hanh dong: <action>
```

## 💡 Ví dụ

### Test case 1

#### Input:

```
4  
D Buddy 3 Golden_Retriever  
C Mimi 2 Trang  
D Max 5 Husky  
C Luna 1 Den
```

#### Output:

```
Moi dong vat tu gioi thieu va hoat dong:
```

```
Buddy (3 tuoi):  
- Loai: Dog  
- Giong: Golden_Retriever  
- Tieng keu: Gau gau!  
- Hanh dong: Chay quanh san
```

```
Mimi (2 tuoi):  
- Loai: Cat  
- Mau long: Trang  
- Tieng keu: Meo meo!  
- Hanh dong: Leo cay
```

```
Max (5 tuoi):  
- Loai: Dog  
- Giong: Husky  
- Tieng keu: Gau gau!  
- Hanh dong: Chay quanh san
```

```
Luna (1 tuoi):  
- Loai: Cat  
- Mau long: Den  
- Tieng keu: Meo meo!  
- Hanh dong: Leo cay
```

## 💡 Gợi ý

**Tư duy:**

- Class cha Animal với name, age
- Dog và Cat extends Animal
- Override methods: makeSound(), performAction()
- Mỗi loại có tiếng kêu và hành động riêng

**Tags:** inheritance, override, polymorphism, competitive

---

# CodeForge - B04 - Hình Học Cơ Bản

**Độ khó:** ★ ★ ★ Hard

**Loại:** ☑ Competitive

**Kiến thức:** Inheritance, Abstract Concept, Method Overriding

## 📝 Đề bài

Xây dựng hệ thống quản lý hình học với 2 loại:

### 1. Hình tròn (Circle):

- Có bán kính ( $r$ )
- Diện tích =  $\pi \times r^2$  (dùng  $\pi = 3.14159$ )
- Chu vi =  $2 \times \pi \times r$

### 2. Hình chữ nhật (Rectangle):

- Có chiều dài (length) và chiều rộng (width)
- Diện tích =  $length \times width$
- Chu vi =  $2 \times (length + width)$

**Yêu cầu:** Nhập N hình, tính tổng diện tích và tìm hình có chu vi lớn nhất.

#### ◊ Input

Dòng 1: N ( $1 \leq N \leq 100$ )

N dòng:

Circle:

C <radius>

Rectangle:

R <length> <width>

#### Trong đó:

- C hoặc R: Loại hình
- radius: Bán kính (số thực,  $> 0$ )
- length, width: Chiều dài, rộng (số thực,  $> 0$ )

#### ◊ Output

Danh sách hình:

Hình <thu\_tu>: <Circle|Rectangle>  
[Bán kính: <radius>] ← Chỉ có Circle

```
[Chieu dai: <length>]           ← Chỉ có Rectangle
[Chieu rong: <width>]            ← Chỉ có Rectangle
Dien tich: <area>
Chu vi: <perimeter>

==== THONG KE ====
Tong dien tich: <totalArea>
Hinh co chu vi lon nhat:
- Loai: <type>
- Chu vi: <maxPerimeter>
```

**Lưu ý:** Làm tròn 2 chữ số thập phân

## Ví dụ

Test case 1

**Input:**

```
4
C 5.0
R 4.0 6.0
C 3.5
R 10.0 2.0
```

**Output:**

Danh sach hinh:

Hinh 1: Circle  
Ban kinh: 5.00  
Dien tich: 78.54  
Chu vi: 31.42

Hinh 2: Rectangle  
Chieu dai: 4.00  
Chieu rong: 6.00  
Dien tich: 24.00  
Chu vi: 20.00

Hinh 3: Circle  
Ban kinh: 3.50  
Dien tich: 38.48  
Chu vi: 21.99

Hinh 4: Rectangle  
Chieu dai: 10.00  
Chieu rong: 2.00  
Dien tich: 20.00

```
Chu vi: 24.00
```

```
==== THONG KE ===
```

```
Tong dien tich: 161.02
```

```
Hinh co chu vi lon nhat:
```

- Loai: Circle
- Chu vi: 31.42

## 💡 Gợi ý

### Tư duy:

- Class cha Shape với methods tính diện tích và chu vi
- Circle và Rectangle override các methods này
- Dùng polymorphism để lưu mang hỗn hợp
- Tìm max perimeter bằng cách duyệt mang

**Tags:** inheritance, override, polymorphism, geometry, competitive

---

# CodeForge - B05 - Hệ Thống Ngân Hàng

**Độ khó:** ★ ★ ★ Hard

**Loại:** ☑ Competitive

**Kiến thức:** Inheritance, Method Overriding, Business Logic

## 📝 Đề bài

Ngân hàng có 2 loại tài khoản:

### 1. Tài khoản tiết kiệm (SavingsAccount):

- Có số tài khoản, tên chủ, số dư
- Lãi suất: 0.5%/tháng
- Tính lãi: số dư × 0.005 × số tháng
- Không được rút nếu số dư sau rút < 1,000,000đ

### 2. Tài khoản thanh toán (CheckingAccount):

- Có số tài khoản, tên chủ, số dư, hạn mức thấu chi (overdraft limit)
- Không có lãi suất
- Có thể rút âm tới hạn mức thấu chi
- Phí thấu chi: 50,000đ nếu số dư âm

**Yêu cầu:** Nhập N tài khoản, thực hiện M giao dịch, in trạng thái cuối.

## ◊ Input

Dòng 1: N (số tài khoản,  $1 \leq N \leq 50$ )

N dòng:

SavingsAccount:

S <accNum> <owner> <balance>

CheckingAccount:

C <accNum> <owner> <balance> <overdraftLimit>

Dòng N+2: M (số giao dịch,  $1 \leq M \leq 100$ )

M dòng:

DEPOSIT <accNum> <amount>

WITHDRAW <accNum> <amount>

INTEREST <accNum> <months> ← Chỉ áp dụng cho SavingsAccount

## ◊ Output

Mỗi giao dịch:

```
GD <thu_tu>: <DEPOSIT|WITHDRAW|INTEREST> - TK <accNum>
- <Ket_qua_giao_dich>
- So du: <balance>d
```

Cuối cùng:

```
== TONG KET ==
TK <accNum> (<owner>) - <SavingsAccount|CheckingAccount>:
So du: <balance>d
[Phi thau chi: <fee>d] ← Chỉ có nếu CheckingAccount âm
```

## Ví dụ

Test case 1

**Input:**

```
3
S SA001 Alice 5000000
C CA001 Bob 2000000 3000000
S SA002 Charlie 10000000
5
DEPOSIT SA001 1000000
WITHDRAW CA001 4000000
INTEREST SA001 6
WITHDRAW SA002 9500000
WITHDRAW CA001 500000
```

**Output:**

```
GD 1: DEPOSIT - TK SA001
- Nop 1000000d thanh cong
- So du: 6000000d

GD 2: WITHDRAW - TK CA001
- Rut 4000000d thanh cong (thau chi 2000000d)
- So du: -2000000d

GD 3: INTEREST - TK SA001
- Lai suat 6 thang: 180000d
- So du: 6180000d

GD 4: WITHDRAW - TK SA002
- Khong the rut! So du con lai < 10000000d
- So du: 10000000d
```

```
GD 5: WITHDRAW - TK CA001
- Rut 500000d thanh cong (thau chi 2500000d)
- So du: -2500000d

==== TONG KET ====
TK SA001 (Alice) - SavingsAccount:
So du: 6180000d

TK CA001 (Bob) - CheckingAccount:
So du: -2500000d
Phi thau chi: 50000d

TK SA002 (Charlie) - SavingsAccount:
So du: 10000000d
```

## 💡 Gợi ý

### Tư duy:

- Class cha BankAccount với số dư, các method cơ bản
- SavingsAccount: override withdraw() với điều kiện min balance
- CheckingAccount: override withdraw() cho phép thấu chi
- Dùng instanceof hoặc method riêng để check loại tài khoản

**Tags:** inheritance, override, business-logic, competitive

---

# CodeForge - B06 - Demo Constructor Chaining

**Độ khó:** ★ ★ Medium

**Loại:** ✅ Competitive

**Kiến thức:** Constructor, super(), Initialization Order

## 📝 Đề bài

Xây dựng hệ thống quản lý học sinh với constructor chaining:

**Class Person (cha):**

- Có name, age
- Constructor: Person(String name, int age)

**Class Student (con):**

- Kế thừa Person
- Thêm studentId, gpa
- Constructor: Student(String studentId, String name, int age, double gpa)
- Phải gọi super(name, age) để khởi tạo phần Person

**Yêu cầu:** Nhập N học sinh, in thứ tự khởi tạo để demo constructor chaining.

## ◊ Input

```
Dòng 1: N (1 ≤ N ≤ 50)
N nhóm, mỗi nhóm 4 dòng:
<studentId>
<name>
<age>
<gpa>
```

## ◊ Output

Mỗi học sinh:

```
==== KHOI TAO STUDENT <thu_tu> ====
Buoc 1: Gọi super(name, age)
    -> Person constructor: name=<name>, age=<age>
Buoc 2: Khoi tao phan Student
    -> studentId=<studentId>, gpa=<gpa>
Student hoan tat:
    Ma SV: <studentId>
    Ten: <name>
    Tuoi: <age>
    GPA: <gpa>
```

## Ví dụ

### Test case 1

#### Input:

```
2
SV001
Alice
20
3.5
SV002
Bob
22
3.8
```

#### Output:

```
==== KHOI TAO STUDENT 1 ====
Buoc 1: Goi super(name, age)
    -> Person constructor: name=Alice, age=20
Buoc 2: Khoi tao phan Student
    -> studentId=SV001, gpa=3.5
Student hoan tat:
    Ma SV: SV001
    Ten: Alice
    Tuoi: 20
    GPA: 3.5

==== KHOI TAO STUDENT 2 ====
Buoc 1: Goi super(name, age)
    -> Person constructor: name=Bob, age=22
Buoc 2: Khoi tao phan Student
    -> studentId=SV002, gpa=3.8
Student hoan tat:
    Ma SV: SV002
    Ten: Bob
    Tuoi: 22
    GPA: 3.8
```

## Gợi ý

### Tư duy:

- super() phải là lệnh ĐẦU TIÊN trong constructor con
- Constructor cha chạy TRƯỚC constructor con
- In log trong constructor để thấy thứ tự

**Tags:** constructor-chaining, super, initialization-order, competitive

---

# CodeForge - B07 - Thứ Tự Khởi Tạo Phức Tạp

**Độ khó:** ★ ★ ★ Hard

**Loại:** ☑ Competitive

**Kiến thức:** Initialization Order, super(), Field Initialization

## 📝 Đề bài

Demo thứ tự khởi tạo trong inheritance với nhiều bước:

### Class Base:

- Field: baseValue (khởi tạo = 100)
- Constructor in: "Base constructor"

### Class Derived extends Base:

- Field: derivedValue (khởi tạo = 200)
- Constructor:
  - Gọi super()
  - In "Derived constructor"
  - Khởi tạo derivedValue = 200

### Thứ tự thực thi khi tạo Derived:

- Khởi tạo field của Base (baseValue = 100)
- Constructor của Base
- Khởi tạo field của Derived (derivedValue = 200)
- Constructor của Derived

**Yêu cầu:** Nhập N objects, mỗi lần tạo object in ra các bước khởi tạo theo đúng thứ tự.

### ◊ Input

```
Dòng 1: N (1 ≤ N ≤ 20)
```

### ◊ Output

Mỗi object:

```
==== TAO OBJECT <thu_tu> ====
Buoc 1: Khoi tao field Base
    baseValue = 100
Buoc 2: Base constructor
    In: "Base constructor"
Buoc 3: Khoi tao field Derived
    derivedValue = 200
```

```
Buoc 4: Derived constructor
In: "Derived constructor"
Object hoan tat!
```

## Ví dụ

### Test case 1

#### Input:

```
2
```

#### Output:

```
==== TAO OBJECT 1 ====
Buoc 1: Khoi tao field Base
    baseValue = 100
Buoc 2: Base constructor
    In: "Base constructor"
Buoc 3: Khoi tao field Derived
    derivedValue = 200
Buoc 4: Derived constructor
    In: "Derived constructor"
Object hoan tat!

==== TAO OBJECT 2 ====
Buoc 1: Khoi tao field Base
    baseValue = 100
Buoc 2: Base constructor
    In: "Base constructor"
Buoc 3: Khoi tao field Derived
    derivedValue = 200
Buoc 4: Derived constructor
    In: "Derived constructor"
Object hoan tat!
```

## Gợi ý

#### Tư duy:

- Thứ tự: Field cha → Constructor cha → Field con → Constructor con
- Dùng println() trong constructor để trace execution
- Hiểu rõ thứ tự này để tránh bug khởi tạo

**Tags:** initialization-order, constructor, super, fields, competitive

# CodeForge - B08 - Override `toString()`

**Độ khó:** ★ ★ Medium

**Loại:** ✅ Competitive

**Kiến thức:** Method Overriding, `toString()`, `super.method()`

## 📝 Đề bài

Xây dựng hệ thống sách với override `toString()`:

### Class Book:

- Có title, author, price
- `toString()`: "Book[title=..., author=..., price=...]"

### Class EBook extends Book:

- Thêm fileSize (MB)
- Override `toString()`:
  - Gọi `super.toString()`
  - Thêm ", fileSize=...MB"
  - Kết quả: "EBook[title=..., author=..., price=..., fileSize=...MB]"

**Yêu cầu:** Nhập N sách (Book hoặc EBook), in `toString()` của từng cuốn.

## ◊ Input

```
Dòng 1: N (1 ≤ N ≤ 50)
```

```
N dòng:
```

```
Book:
```

```
B <title> <author> <price>
```

```
EBook:
```

```
E <title> <author> <price> <fileSize>
```

## ◊ Output

```
Sach <thu_tu>:  
<toString_output>
```

## 📊 Ví dụ

Test case 1

**Input:**

```
3
B Java-Programming Horstmann 450000
E Python-Basics Smith 250000 15.5
B Clean-Code Martin 380000
```

**Output:**

```
Sach 1:
Book[title=Java-Programming, author=Horstmann, price=450000d]

Sach 2:
EBook[title=Python-Basics, author=Smith, price=250000d, fileSize=15.5MB]

Sach 3:
Book[title=Clean-Code, author=Martin, price=380000d]
```

 **Gợi ý****Tư duy:**

- Override `toString()` trong cả `Book` và `EBook`
- `EBook.toString()` gọi `super.toString()` để tái sử dụng code
- Thêm thông tin riêng của `EBook` vào chuỗi

**Tags:** override, toString, super-method-call, competitive

---

# CodeForge - B09 - Override equals()

**Độ khó:** ★ ★ ★ Hard

**Loại:** ☑ Competitive

**Kiến thức:** Method Overriding, equals(), instanceof

## 📝 Đề bài

Xây dựng hệ thống điểm 2D với override equals():

### Class Point:

- Có x, y (tọa độ)
- equals(Object obj):
  - Kiểm tra obj có phải Point không
  - So sánh x và y

### Class ColoredPoint extends Point:

- Thêm color (String)
- Override equals(Object obj):
  - Gọi super.equals(obj) để kiểm tra tọa độ
  - Kiểm tra thêm màu sắc

**Yêu cầu:** Nhập N điểm, sau đó thực hiện M phép so sánh.

### ◊ Input

Dòng 1: N (1 ≤ N ≤ 50)

N dòng:

Point:

P <id> <x> <y>

ColoredPoint:

C <id> <x> <y> <color>

Dòng N+2: M (số lần so sánh)

M dòng: <id1> <id2>

### ◊ Output

Mỗi phép so sánh:

```
So sanh <id1> vs <id2>:  
<id1>: <type> at (<x>, <y>) [color=<color>]  
<id2>: <type> at (<x>, <y>) [color=<color>]
```

```
equals(): <true|false>
Ly do: <explanation>
```

## 💡 Ví dụ

### Test case 1

#### Input:

```
4
P p1 5 10
C c1 5 10 Red
P p2 5 10
C c2 5 10 Blue
3
p1 p2
p1 c1
c1 c2
```

#### Output:

```
So sanh p1 vs p2:
p1: Point at (5, 10)
p2: Point at (5, 10)
equals(): true
Ly do: Cung toa do (5, 10)

So sanh p1 vs c1:
p1: Point at (5, 10)
c1: ColoredPoint at (5, 10) [color=Red]
equals(): false
Ly do: Khac loai (Point vs ColoredPoint)

So sanh c1 vs c2:
c1: ColoredPoint at (5, 10) [color=Red]
c2: ColoredPoint at (5, 10) [color=Blue]
equals(): false
Ly do: Cung toa do nhung khac mau (Red vs Blue)
```

## 💡 Gợi ý

#### Tư duy:

- equals() phải kiểm tra instanceof
- super.equals() kiểm tra phần cha
- Thêm logic kiểm tra fields của con

**Tags:** override, equals, instanceof, super-method-call, competitive



# CodeForge - B10 - Demo super.method()

**Độ khó:** ★ ★ Medium

**Loại:** ✅ Competitive

**Kiến thức:** Method Overriding, super.method()

## 📝 Đề bài

Xây dựng hệ thống máy tính với override calculate():

### Class Calculator:

- Method calculate(int a, int b): return a + b (cộng)

### Class AdvancedCalculator extends Calculator:

- Override calculate(int a, int b):
  - Tính tổng (gọi super.calculate(a, b))
  - Tính tích (a × b)
  - Return cả 2 kết quả

**Yêu cầu:** Nhập N phép tính, mỗi phép cho 2 số, in kết quả từ cả Calculator và AdvancedCalculator.

## ◊ Input

Dòng 1: N (1 ≤ N ≤ 50)

N dòng: <a> <b>

## ◊ Output

Mỗi phép tính:

```
Phep tinh <thu_tu>: a=<a>, b=<b>
Calculator (cha):
    Ket qua: <sum>
AdvancedCalculator (con):
    Tong (super.calculate): <sum>
    Tich: <product>
```

## 💻 Ví dụ

Test case 1

### Input:

```
3  
5 3  
10 7  
-4 6
```

### Output:

```
Phep tinh 1: a=5, b=3  
Calculator (cha):  
Ket qua: 8  
AdvancedCalculator (con):  
Tong (super.calculate): 8  
Tich: 15
```

```
Phep tinh 2: a=10, b=7  
Calculator (cha):  
Ket qua: 17  
AdvancedCalculator (con):  
Tong (super.calculate): 17  
Tich: 70
```

```
Phep tinh 3: a=-4, b=6  
Calculator (cha):  
Ket qua: 2  
AdvancedCalculator (con):  
Tong (super.calculate): 2  
Tich: -24
```

### 💡 Gợi ý

### Tư duy:

- super.method() gọi implementation của cha
- Tái sử dụng logic cha, thêm logic mới
- Tránh duplicate code

**Tags:** override, super-method-call, reuse, competitive

# CodeForge - B11 - Overriding vs Overloading

**Độ khó:** ★ ★ ★ Hard

**Loại:** ☑ Competitive

**Kiến thức:** Overriding, Overloading, Polymorphism

## 📝 Đề bài

Demo sự khác biệt giữa Overriding và Overloading:

### Class Printer:

- print(String msg): In "Printer: "
- print(int num): In "Printer number: " (overloading)

### Class ColorPrinter extends Printer:

- Override print(String msg): In "ColorPrinter: (in color)"
- Kế thừa print(int num) từ Printer (không override)

**Yêu cầu:** Nhập N lệnh in, mỗi lệnh có thể là String hoặc int, demo overriding và overloading.

## ◊ Input

Dòng 1: N (1 ≤ N ≤ 50)

N dòng:

String:

S <message>

Integer:

I <number>

## ◊ Output

Mỗi lệnh:

Lenh <thu\_tu>: <type> - <value>

Sử dụng Printer (cha):

<output>

Sử dụng ColorPrinter (con):

<output>

Ghi chú: <Overriding|Overloading|Ke\_thua>

## Ví dụ

### Test case 1

#### Input:

```
4
S Hello
I 42
S World
I 99
```

#### Output:

```
Lenh 1: String - Hello
Su dung Printer (cha):
Printer: Hello

Su dung ColorPrinter (con):
ColorPrinter: Hello (in color)

Ghi chu: Overriding - print(String) bi override

Lenh 2: Integer - 42
Su dung Printer (cha):
Printer number: 42

Su dung ColorPrinter (con):
Printer number: 42

Ghi chu: Ke thua - print(int) khong override, dung cua cha

Lenh 3: String - World
Su dung Printer (cha):
Printer: World

Su dung ColorPrinter (con):
ColorPrinter: World (in color)

Ghi chu: Overriding - print(String) bi override

Lenh 4: Integer - 99
Su dung Printer (cha):
Printer number: 99

Su dung ColorPrinter (con):
```

Printer number: 99

Ghi chú: Khi thừa - print(int) không override, dùng của cha

## 💡 Gợi ý

### Tư duy:

- Overloading: Cùng tên method, khác signature (trong cùng class)
- Overriding: Cùng tên + signature, khác implementation (giữa cha-con)
- ColorPrinter override print(String), nhưng không override print(int)

**Tags:** overriding, overloading, polymorphism, competitive

---

# CodeForge - B12 - protected Access Modifier

**Độ khó:** ★ ★ Medium

**Loại:** ✅ Competitive

**Kiến thức:** protected, Access Modifiers, Inheritance

## 📝 Đề bài

Demo sự khác biệt giữa private, protected, và public trong inheritance:

### Class Account:

- private String accountNumber (chỉ truy cập trong Account)
- protected double balance (truy cập từ Account và các class con)
- public String owner (truy cập từ mọi nơi)

### Class PremiumAccount extends Account:

- Có thể truy cập balance (protected) → Tính lãi
- KHÔNG thể truy cập accountNumber (private)
- Có thể truy cập owner (public)

**Yêu cầu:** Nhập N tài khoản, demo access từ class con.

## ◊ Input

Dòng 1: N (1 ≤ N ≤ 50)

N dòng:

Account:

A <accNum> <owner> <balance>

PremiumAccount:

P <accNum> <owner> <balance> <interestRate>

## ◊ Output

Mỗi tài khoản:

Tài khoản <thu\_tu>: <Account|PremiumAccount>

Truy cập từ class con (PremiumAccount):

- accountNumber: KHÔNG TRUY CẬP ĐƯỢC (private)
- balance: <balance>d (protected - OK)
- owner: <owner> (public - OK)

[Tính lãi: <interest>d] ← Chỉ có PremiumAccount

## Ví dụ

### Test case 1

#### Input:

```
3
A ACC001 Alice 5000000
P ACC002 Bob 10000000 0.05
A ACC003 Charlie 3000000
```

#### Output:

```
Tai khoan 1: Account

Truy cap tu class con (PremiumAccount):
- accountNumber: KHONG TRUY CAP DUOC (private)
- balance: 5000000d (protected - OK)
- owner: Alice (public - OK)

Tai khoan 2: PremiumAccount

Truy cap tu class con (PremiumAccount):
- accountNumber: KHONG TRUY CAP DUOC (private)
- balance: 10000000d (protected - OK)
- owner: Bob (public - OK)

Tinh lai: 500000d (balance * interestRate)

Tai khoan 3: Account

Truy cap tu class con (PremiumAccount):
- accountNumber: KHONG TRUY CAP DUOC (private)
- balance: 3000000d (protected - OK)
- owner: Charlie (public - OK)
```

## Gợi ý

### Tư duy:

- private: Chỉ truy cập trong class đó
- protected: Truy cập từ class đó + class con
- public: Truy cập từ mọi nơi
- Class con KHÔNG thể truy cập private fields của cha

**Tags:** protected, access-modifiers, inheritance, competitive

# CodeForge - B13 - final Method Cannot Override

**Độ khó:** ★ ★ Medium

**Loại:** ✅ Competitive

**Kiến thức:** final method, Method Overriding

## Đề bài

Demo final method không thể override:

### **Class Vehicle:**

- final void startEngine(): In "Engine started" (FINAL - không override được)
- void drive(): In "Vehicle is moving" (có thể override)

### **Class Car extends Vehicle:**

- KHÔNG thể override startEngine() (vì final)
- CÓ THỂ override drive(): In "Car is moving on road"

**Yêu cầu:** Nhập N xe, mỗi xe thực hiện startEngine() và drive(), demo final method.

## ◊ Input

Dòng 1: N (1 ≤ N ≤ 50)

N dòng:

Vehicle:

V <name>

Car:

C <name>

## ◊ Output

Mỗi xe:

Xe <thu\_tu>: <name> (<Vehicle|Car>)

startEngine() - FINAL METHOD:  
Engine started

drive() - <Overridden|Original>:  
<output>

## Ví dụ

## Test case 1

### Input:

```
3  
V Bike  
C Toyota  
C Honda
```

### Output:

```
Xe 1: Bike (Vehicle)  
  
startEngine() - FINAL METHOD:  
Engine started  
  
drive() - Original:  
Vehicle is moving  
  
Xe 2: Toyota (Car)  
  
startEngine() - FINAL METHOD:  
Engine started  
  
drive() - Overridden:  
Car is moving on road  
  
Xe 3: Honda (Car)  
  
startEngine() - FINAL METHOD:  
Engine started  
  
drive() - Overridden:  
Car is moving on road
```

## 💡 Gợi ý

### Tư duy:

- final method: Không thể override trong class con
- Dùng final khi muốn đảm bảo behavior không thay đổi
- Method thường: Có thể override

**Tags:** final-method, override, inheritance, competitive

# CodeForge - B14 - final Class Cannot Extend

**Độ khó:** ★ ★ Medium

**Loại:** ✅ Competitive

**Kiến thức:** final class, Inheritance

## 📝 Đề bài

Demo final class không thể kế thừa:

### Class String (giả lập):

- final class MyString (FINAL - không extends được)
- Methods: length(), toUpperCase(), toLowerCase()

### Class MyStringBuilder (không final):

- CÓ THỂ extends: class ExtendedStringBuilder extends MyStringBuilder

**Yêu cầu:** Nhập N strings, thực hiện operations, demo final class.

#### ◊ Input

```
Dòng 1: N (1 ≤ N ≤ 50)
N dòng: <text> <operation>
```

```
Operations:
LENGTH
UPPER
LOWER
```

#### ◊ Output

Mỗi string:

```
String <thu_tu>: "<text>"
Operation: <operation>
Result: <result>

Lưu ý: MyString là FINAL CLASS - không thể extends
```

Cuối cùng:

```
==== FINAL CLASS ===
MyString: KHONG THE KE THUA
```

```
MyStringBuilder: CO THE KE THUA
```

## Ví dụ

### Test case 1

#### Input:

```
3
Hello LENGTH
World UPPER
Java LOWER
```

#### Output:

```
String 1: "Hello"
Operation: LENGTH
Result: 5

Luu y: MyString la FINAL CLASS - khong the extends

String 2: "World"
Operation: UPPER
Result: WORLD

Luu y: MyString la FINAL CLASS - khong the extends

String 3: "Java"
Operation: LOWER
Result: java

Luu y: MyString la FINAL CLASS - khong the extends

==== FINAL CLASS ===
MyString: KHONG THE KE THUA
MyStringBuilder: CO THE KE THUA
```

## Gợi ý

#### Tư duy:

- final class: Không thể extends
- Dùng final class khi muốn đảm bảo implementation không thay đổi
- Ví dụ thực tế: String, Integer, Math trong Java đều final

**Tags:** final-class, inheritance, competitive

# CodeForge - B15 - Multi-Level Inheritance

**Độ khó:** ★ ★ ★ Hard

**Loại:** ☑ Competitive

**Kiến thức:** Multi-level Inheritance, Constructor Chaining

## 📝 Đề bài

Demo kế thừa đa cấp (A → B → C):

**Class LivingBeing:**

- Có age
- eat(): "Living being is eating"

**Class Animal extends LivingBeing:**

- Thêm name
- Override eat(): "Animal is eating"
- move(): "Animal is moving"

**Class Dog extends Animal:**

- Thêm breed
- Override eat(): "Dog (breed: ) is eating"
- Override move(): "Dog is running"
- bark(): "Dog is barking"

**Yêu cầu:** Nhập N sinh vật, demo các methods qua 3 cấp độ.

## ◊ Input

Dòng 1: N (1 ≤ N ≤ 50)

N dòng:

LivingBeing:

L <age>

Animal:

A <age> <name>

Dog:

D <age> <name> <breed>

## ◊ Output

Mỗi sinh vật:

```
Sinh vat <thu_tu>: <LivingBeing|Animal|Dog>
Thong tin:
- Tuoi: <age>
[- Ten: <name>]
[- Giong: <breed>]

Hanh dong:
eat(): <output>
[move(): <output>]
[bark(): <output>]
```

## Ví dụ

### Test case 1

#### Input:

```
3
L 100
A 5 Lion
D 3 Buddy Golden_Retriever
```

#### Output:

```
Sinh vat 1: LivingBeing
Thong tin:
- Tuoi: 100

Hanh dong:
eat(): Living being is eating

Sinh vat 2: Animal
Thong tin:
- Tuoi: 5
- Ten: Lion

Hanh dong:
eat(): Animal Lion is eating
move(): Animal is moving

Sinh vat 3: Dog
Thong tin:
- Tuoi: 3
- Ten: Buddy
- Giong: Golden_Retriever

Hanh dong:
eat(): Dog Buddy (breed: Golden_Retriever) is eating
```

```
move(): Dog is running  
bark(): Dog is barking
```

## 💡 Gợi ý

### Tư duy:

- Dog extends Animal extends LivingBeing (3 cấp)
- Constructor Dog phải gọi super() của Animal
- Constructor Animal phải gọi super() của LivingBeing
- Override methods ở nhiều cấp

**Tags:** multi-level-inheritance, constructor-chaining, override, competitive

---

# CodeForge - B16 - Hệ Thống Sinh Viên Đại Học

**Độ khó:** ★ ★ ★ Hard

**Loại:** ☑ Competitive

**Kiến thức:** Inheritance, Method Overriding, Business Logic

## 📝 Đề bài

Trường đại học quản lý 2 loại sinh viên:

### 1. Sinh viên đại học (UndergraduateStudent):

- Có mã SV, tên, năm học (1-4), GPA
- Học phí = 15,000,000đ/năm

### 2. Sinh viên cao học (GraduateStudent):

- Có mã SV, tên, chuyên ngành, GPA, có hỗ trợ nghiên cứu không
- Học phí = 25,000,000đ/năm
- Nếu có hỗ trợ nghiên cứu: giảm 50% học phí

**Yêu cầu:** Nhập N sinh viên, tính tổng học phí, tìm sinh viên GPA cao nhất từng loại.

## ◊ Input

Dòng 1: N (1 ≤ N ≤ 100)

N dòng:

UndergraduateStudent:

U <id> <name> <year> <gpa>

GraduateStudent:

G <id> <name> <major> <gpa> <hasResearchSupport: Y/N>

## ◊ Output

Tổng sinh viên:

```
SV <thu_tu>: <id> - <name>
Loai: <Undergraduate|Graduate>
[Nam: <year>]           ← Chỉ Undergraduate
[Chuyen nganh: <major>]   ← Chỉ Graduate
[Ho tro nghien cuu: <Y/N>] ← Chỉ Graduate
GPA: <gpa>
Hoc phi: <tuition>d
```

Tổng kết:

```
==== THONG KE ====
Tong sinh vien: <total>
- Undergraduate: <ugCount> - Hoc phi: <ugTuition>d
- Graduate: <gradCount> - Hoc phi: <gradTuition>d

GPA cao nhat:
- Undergraduate: <name> (GPA: <gpa>)
- Graduate: <name> (GPA: <gpa>)

TONG HOC PHI: <totalTuition>d
```

## Ví dụ

Test case 1

**Input:**

```
5
U SV001 Alice 2 3.5
G SV002 Bob AI 3.8 Y
U SV003 Charlie 3 3.2
G SV004 Diana ML 3.9 N
U SV005 Eve 1 3.7
```

**Output:**

```
SV 1: SV001 - Alice
Loai: Undergraduate
Nam: 2
GPA: 3.5
Hoc phi: 15000000d

SV 2: SV002 - Bob
Loai: Graduate
Chuyen nganh: AI
Ho tro nghien cuu: Y
GPA: 3.8
Hoc phi: 12500000d

SV 3: SV003 - Charlie
Loai: Undergraduate
Nam: 3
GPA: 3.2
Hoc phi: 15000000d

SV 4: SV004 - Diana
```

```
Loai: Graduate
Chuyen nganh: ML
Ho tro nghien cuu: N
GPA: 3.9
Hoc phi: 25000000d
```

```
SV 5: SV005 - Eve
Loai: Undergraduate
Nam: 1
GPA: 3.7
Hoc phi: 15000000d
```

==== THONG KE ===

```
Tong sinh vien: 5
- Undergraduate: 3 - Hoc phi: 45000000d
- Graduate: 2 - Hoc phi: 37500000d
```

GPA cao nhat:
- Undergraduate: Eve (GPA: 3.7)
- Graduate: Diana (GPA: 3.9)

TONG HOC PHI: 82500000d

## 💡 Gợi ý

### Tư duy:

- Class cha Student với id, name, gpa
- UndergraduateStudent và GraduateStudent extends Student
- Override calculateTuition() cho mỗi loại
- Dùng polymorphic array để lưu cả 2 loại

**Tags:** inheritance, override, polymorphism, business-logic, competitive

---

# CodeForge - B17 - Hệ Thống Sản Phẩm E-Commerce

**Độ khó:** ★ ★ ★ Hard

**Loại:** ☑ Competitive

**Kiến thức:** Inheritance, Method Overriding, Complex Logic

## 📝 Đề bài

Cửa hàng online bán 2 loại sản phẩm:

### 1. Sách (Book):

- Có mã SP, tên, giá, số trang
- Phí ship = 15,000đ (cố định)

### 2. Điện tử (Electronics):

- Có mã SP, tên, giá, bảo hành (tháng)
- Phí ship = 2% giá trị sản phẩm (tối thiểu 50,000đ)

**Yêu cầu:** Nhập N sản phẩm, tính tổng giá trị đơn hàng (giá + ship).

## ◊ Input

Dòng 1: N ( $1 \leq N \leq 100$ )

N dòng:

Book:

B <id> <name> <price> <pages>

Electronics:

E <id> <name> <price> <warrantyMonths>

## ◊ Output

Tổng sản phẩm:

```
SP <thu_tu>: <id> - <name>
Loai: <Book|Electronics>
Gia: <price>d
[So trang: <pages>]           ← Chỉ Book
[Bao hanh: <warranty> thang]   ← Chỉ Electronics
Phi ship: <shippingFee>d
Tong: <total>d
```

Tổng kết:

```
==== DON HANG ====
Tong san pham: <count>
Tong gia tri: <totalPrice>d
Tong phi ship: <totalShipping>d
TONG THANH TOAN: <grandTotal>d
```

## Ví dụ

### Test case 1

#### Input:

```
4
B B001 Clean-Code 350000 464
E E001 iPhone15 25000000 12
B B002 Java-Programming 450000 1200
E E002 AirPods 5000000 6
```

#### Output:

```
SP 1: B001 - Clean-Code
Loai: Book
Gia: 350000d
So trang: 464
Phi ship: 15000d
Tong: 365000d

SP 2: E001 - iPhone15
Loai: Electronics
Gia: 25000000d
Bao hanh: 12 thang
Phi ship: 500000d
Tong: 25500000d

SP 3: B002 - Java-Programming
Loai: Book
Gia: 450000d
So trang: 1200
Phi ship: 15000d
Tong: 465000d

SP 4: E002 - AirPods
Loai: Electronics
Gia: 5000000d
Bao hanh: 6 thang
Phi ship: 100000d
Tong: 5100000d
```

```
==== DON HANG ====
Tong san pham: 4
Tong gia tri: 30800000d
Tong phi ship: 630000d
TONG THANH TOAN: 31430000d
```

## 💡 Gợi ý

### Tư duy:

- Class Product với id, name, price
- Book và Electronics extends Product
- Override calculateShippingFee() cho mỗi loại
- Electronics: Math.max(price \* 0.02, 50000)

**Tags:** inheritance, override, business-logic, competitive

---

# CodeForge - B18 - Trình Tự Khởi Tạo Đầy Đủ

**Độ khó:** ★ ★ ★ Hard

**Loại:** ☑ Competitive

**Kiến thức:** Initialization Order, Constructor, Fields

## 📝 Đề bài

Demo đầy đủ trình tự khởi tạo trong inheritance:

### Class A:

- static field: staticA = 1
- instance field: instanceA = 10
- static block: in "Static block A"
- Constructor: in "Constructor A"

### Class B extends A:

- static field: staticB = 2
- instance field: instanceB = 20
- static block: in "Static block B"
- Constructor: in "Constructor B"

### Thứ tự khi tạo object B:

1. Static block A
2. Static block B
3. Instance field A (instanceA = 10)
4. Constructor A
5. Instance field B (instanceB = 20)
6. Constructor B

**Yêu cầu:** In đầy đủ trình tự khởi tạo khi tạo N objects.

#### ◊ Input

```
Dòng 1: N (1 ≤ N ≤ 10)
```

#### ◊ Output

#### Lần đầu tiên (static chỉ chạy 1 lần):

```
==== TAO OBJECT 1 ====
[STATIC] Static block A - staticA = 1
[STATIC] Static block B - staticB = 2
```

```
[INSTANCE] Instance field A - instanceA = 10
[CONSTRUCTOR] Constructor A
[INSTANCE] Instance field B - instanceB = 20
[CONSTRUCTOR] Constructor B
```

### Lần thứ 2+ (static không chạy nữa):

```
==== TAO OBJECT <n> ====
[INSTANCE] Instance field A - instanceA = 10
[CONSTRUCTOR] Constructor A
[INSTANCE] Instance field B - instanceB = 20
[CONSTRUCTOR] Constructor B
```

## Ví dụ

### Test case 1

#### Input:

```
3
```

#### Output:

```
==== TAO OBJECT 1 ====
[STATIC] Static block A - staticA = 1
[STATIC] Static block B - staticB = 2
[INSTANCE] Instance field A - instanceA = 10
[CONSTRUCTOR] Constructor A
[INSTANCE] Instance field B - instanceB = 20
[CONSTRUCTOR] Constructor B

==== TAO OBJECT 2 ====
[INSTANCE] Instance field A - instanceA = 10
[CONSTRUCTOR] Constructor A
[INSTANCE] Instance field B - instanceB = 20
[CONSTRUCTOR] Constructor B

==== TAO OBJECT 3 ====
[INSTANCE] Instance field A - instanceA = 10
[CONSTRUCTOR] Constructor A
[INSTANCE] Instance field B - instanceB = 20
[CONSTRUCTOR] Constructor B

==== SUMMARY ====
Static blocks: Chỉ chạy 1 lần (khi class được load)
Instance fields + Constructors: Chạy mỗi khi tạo object
```

Thu tu: Static A -> Static B -> Instance A -> Constructor A -> Instance B -> Constructor B

## 💡 Gợi ý

### Tư duy:

- Static: Chỉ chạy 1 lần khi class load
- Instance: Chạy mỗi lần tạo object
- Thứ tự: Static cha → Static con → Instance cha → Constructor cha → Instance con → Constructor con

**Tags:** initialization-order, static, constructor, fields, competitive

---

# CodeForge - B19A - Hệ Thống Quản Lý Trường Đại Học

**Độ khó:** ★ ★ ★ ★ Very Hard

**Loại:** ☑ Competitive (Advanced)

**Kiến thức:** Complex Inheritance, Multiple Classes, Business Logic

## 📝 Đề bài

Xây dựng hệ thống phức tạp cho trường đại học:

### Class Person:

- id, name, age

### Class Student extends Person:

- studentId, gpa
- Học phí = 10,000,000đ

### Class InternationalStudent extends Student:

- country
- Học phí = 20,000,000đ + phí visa 5,000,000đ

### Class Teacher extends Person:

- teacherId, subject, salary

### Class Professor extends Teacher:

- researchArea
- Salary = salary × 1.5

**Yêu cầu:** Nhập N người (có thể là Student, InternationalStudent, Teacher, Professor), quản lý và thống kê.

## ◊ Input

Dòng 1: N (1 ≤ N ≤ 100)

N dòng:

Student:

S <id> <name> <age> <studentId> <gpa>

InternationalStudent:

I <id> <name> <age> <studentId> <gpa> <country>

Teacher:

T <id> <name> <age> <teacherId> <subject> <salary>

**Professor:**

```
P <id> <name> <age> <teacherId> <subject> <salary> <researchArea>
```

## ❖ Output

Từng người:

```
Nguoi <thu_tu>: <name> (<age> tuoi)
Vai tro: <Student|InternationalStudent|Teacher|Professor>
[Ma SV: <studentId>, GPA: <gpa>]
[Quoc gia: <country>]
[Ma GV: <teacherId>, Mon: <subject>]
[Linh vuc nghien cuu: <researchArea>]
[Hoc phi: <tuition>d]
[Luong: <actualSalary>d]
```

Thống kê:

```
==== THONG KE ====
Sinh vien: <studentCount>
- Trong nuoc: <localCount> - Hoc phi: <localTuition>d
- Quoc te: <intlCount> - Hoc phi: <intlTuition>d

Giang vien: <teacherCount>
- Giang vien: <teacherOnlyCount> - Luong: <teacherSalary>d
- Giao su: <profCount> - Luong: <profSalary>d

TONG HOC PHI THU: <totalTuition>d
TONG LUONG TRA: <totalSalary>d
```

## 📊 Ví dụ

Test case 1

**Input:**

```
6
S P001 Alice 20 SV001 3.5
I P002 Bob 22 SV002 3.8 USA
T P003 Charlie 35 GV001 Math 20000000
P P004 Diana 45 GV002 Physics 25000000 Quantum
S P005 Eve 21 SV003 3.2
P P006 Frank 50 GV003 CS 30000000 AI
```

**Output:**

Nguoi 1: Alice (20 tuoi)

Vai tro: Student

Ma SV: SV001, GPA: 3.5

Hoc phi: 10000000d

Nguoi 2: Bob (22 tuoi)

Vai tro: InternationalStudent

Ma SV: SV002, GPA: 3.8

Quoc gia: USA

Hoc phi: 25000000d

Nguoi 3: Charlie (35 tuoi)

Vai tro: Teacher

Ma GV: GV001, Mon: Math

Luong: 20000000d

Nguoi 4: Diana (45 tuoi)

Vai tro: Professor

Ma GV: GV002, Mon: Physics

Linh vuc nghien cuu: Quantum

Luong: 37500000d

Nguoi 5: Eve (21 tuoi)

Vai tro: Student

Ma SV: SV003, GPA: 3.2

Hoc phi: 10000000d

Nguoi 6: Frank (50 tuoi)

Vai tro: Professor

Ma GV: GV003, Mon: CS

Linh vuc nghien cuu: AI

Luong: 45000000d

==== THONG KE ===

Sinh vien: 3

- Trong nuoc: 2 - Hoc phi: 20000000d

- Quoc te: 1 - Hoc phi: 25000000d

Giang vien: 3

- Giang vien: 1 - Luong: 20000000d

- Giao su: 2 - Luong: 82500000d

TONG HOC PHI THU: 45000000d

TONG LUONG TRA: 102500000d

## 💡 Gợi ý

### Tư duy:

- Multi-level inheritance: Person → Student → InternationalStudent
- Multi-level inheritance: Person → Teacher → Professor

- Polymorphic array Person[] để lưu tất cả
- instanceof để phân loại

**Tags:** multi-level-inheritance, polymorphism, complex-logic, advanced, competitive

---

# CodeForge - B20A - Hệ Thống Thanh Toán

**Độ khó:** ★ ★ ★ ★ Very Hard

**Loại:** ☑ Competitive (Advanced)

**Kiến thức:** Inheritance, Method Overriding, Complex Business Logic

## 📝 Đề bài

Xây dựng hệ thống thanh toán với nhiều phương thức:

**Class Payment (abstract concept):**

- amount, description

**Class CashPayment extends Payment:**

- Không có phí
- Tổng = amount

**Class CardPayment extends Payment:**

- cardNumber (4 số cuối)
- Phí = 2% amount
- Tổng = amount + phí

**Class OnlinePayment extends Payment:**

- provider (Momo, ZaloPay, VNPay)
- Phí phụ thuộc provider:
  - Momo: 1% amount
  - ZaloPay: 1.5% amount
  - VNPay: 2.5% amount
- Tổng = amount + phí

**Yêu cầu:** Nhập N giao dịch, tính tổng tiền và phí từng loại.

## ◊ Input

Dòng 1: N ( $1 \leq N \leq 100$ )

N dòng:

Cash:

CASH <amount> <description>

Card:

CARD <amount> <description> <last4Digits>

Online:

ONLINE <amount> <description> <provider>

## ❖ Output

Từng giao dịch:

```
GD <thu_tu>: <description>
Loai: <Cash|Card|Online>
So tien: <amount>d
[Ma the: ****<last4>]           ← Chỉ Card
[Nha cung cap: <provider>]      ← Chỉ Online
Phi: <fee>d
Tong: <total>d
```

Thống kê:

```
==== THONG KE ====
Cash: <cashCount> GD - Tong: <cashTotal>d - Phi: 0d
Card: <cardCount> GD - Tong: <cardTotal>d - Phi: <cardFee>d
Online: <onlineCount> GD - Tong: <onlineTotal>d - Phi: <onlineFee>d

TONG GIAO DICH: <grandTotal>d
TONG PHI: <totalFee>d
```

## 📊 Ví dụ

Test case 1

**Input:**

```
5
CASH 500000 Mua-do-an
CARD 2000000 Mua-laptop 1234
ONLINE 1000000 Nap-tien-dien-thoai Momo
CARD 5000000 Mua-TV 5678
ONLINE 3000000 Chuyen-khoan ZaloPay
```

**Output:**

```
GD 1: Mua-do-an
Loai: Cash
So tien: 500000d
Phi: 0d
Tong: 500000d

GD 2: Mua-laptop
```

```
Loai: Card
So tien: 2000000d
Ma the: ****1234
Phi: 40000d
Tong: 2040000d
```

```
GD 3: Nap-tien-dien-thoai
Loai: Online
So tien: 1000000d
Nha cung cap: Momo
Phi: 10000d
Tong: 1010000d
```

```
GD 4: Mua-TV
Loai: Card
So tien: 5000000d
Ma the: ****5678
Phi: 100000d
Tong: 5100000d
```

```
GD 5: Chuyen-khoan
Loai: Online
So tien: 3000000d
Nha cung cap: ZaloPay
Phi: 45000d
Tong: 3045000d
```

```
==== THONG KE ====
Cash: 1 GD - Tong: 500000d - Phi: 0d
Card: 2 GD - Tong: 7140000d - Phi: 140000d
Online: 2 GD - Tong: 4055000d - Phi: 55000d
```

```
TONG GIAO DICH: 11695000d
TONG PHI: 195000d
```

## 💡 Gợi ý

### Tư duy:

- Class Payment làm base
- Override calculateFee() cho từng loại
- OnlinePayment có logic phức tạp nhất (switch provider)

**Tags:** inheritance, override, complex-business-logic, advanced, competitive

# CodeForge - B21A - Hệ Thống Nhân Vật Game RPG

**Độ khó:** ★ ★ ★ ★ Very Hard

**Loại:** ☑ Competitive (Advanced)

**Kiến thức:** Inheritance, Polymorphism, Game Logic

## 📝 Đề bài

Xây dựng hệ thống nhân vật game RPG:

### Class Character:

- name, level, hp (health points), baseDamage

### Class Warrior extends Character:

- armor (giáp)
- Damage dealt = baseDamage × 1.5
- Damage taken = incomingDamage - armor (min 1)

### Class Mage extends Character:

- mana
- Damage dealt = baseDamage × 2 (nếu mana >= 50)
- Damage dealt = baseDamage (nếu mana < 50)
- Mỗi đòn tấn 50 mana
- Damage taken = incomingDamage × 1.2

### Class Archer extends Character:

- critChance (% chí mạng)
- Damage dealt = baseDamage × 3 (nếu crit)
- Damage dealt = baseDamage (nếu không crit)
- Crit xảy ra nếu random < critChance/100
- Damage taken = incomingDamage

**Yêu cầu:** Nhập N nhân vật, mô phỏng M đòn đánh.

## ◊ Input

Dòng 1: N (số nhân vật,  $2 \leq N \leq 10$ )

N dòng:

Warrior:

W <name> <level> <hp> <baseDamage> <armor>

Mage:

M <name> <level> <hp> <baseDamage> <mana>

Archer:  
A <name> <level> <hp> <baseDamage> <critChance>

Dòng N+2: K (số đòn đánh)  
K dòng: <attackerName> <defenderName>

## ❖ Output

Mỗi đòn đánh:

```
Don <thu_tu>: <attacker> -> <defender>
- <attacker> (<class>) tan cong: <damageDealt> dame
- <defender> (<class>) nhan: <damageTaken> dame
- HP: <oldHP> -> <newHP>
[- Mana: <oldMana> -> <newMana>] ← Chỉ Mage attacker
```

Kết thúc:

```
==== KET THUC ====
<name> (<class>):
HP: <hp> / <maxHP>
[Mana: <mana>] ← Chỉ Mage
Status: <Alive|Dead>
```

## Ví dụ

Test case 1

**Input:**

```
3
W Knight 10 1000 100 50
M Wizard 10 800 120 150
A Hunter 10 900 80 30
4
Knight Wizard
Wizard Knight
Hunter Knight
Knight Hunter
```

**Output:**

```
Don 1: Knight -> Wizard
- Knight (Warrior) tan cong: 150 dame
```

- Wizard (Mage) nhan: 180 dame
- HP: 800 -> 620

Đòn 2: Wizard -> Knight

- Wizard (Mage) tan cong: 240 dame
- Knight (Warrior) nhan: 190 dame
- HP: 1000 -> 810
- Mana: 150 -> 100

Đòn 3: Hunter -> Knight

- Hunter (Archer) tan cong: 80 dame
- Knight (Warrior) nhan: 30 dame
- HP: 810 -> 780

Đòn 4: Knight -> Hunter

- Knight (Warrior) tan cong: 150 dame
- Hunter (Archer) nhan: 150 dame
- HP: 900 -> 750

==== KẾT THÚC ===

Knight (Warrior):

HP: 780 / 1000

Status: Alive

Wizard (Mage):

HP: 620 / 800

Mana: 100

Status: Alive

Hunter (Archer):

HP: 750 / 900

Status: Alive

## 💡 Gợi ý

### Tư duy:

- Override calculateDamageDealt() và calculateDamageTaken()
- Mỗi class có công thức riêng
- Archer cần random để crit

**Tags:** inheritance, override, game-logic, polymorphism, advanced, competitive

# CodeForge - B22A - Mạng Lưới Giao Thông

**Độ khó:** ★ ★ ★ ★ Very Hard

**Loại:** ☑ Competitive (Advanced)

**Kiến thức:** Inheritance, Complex Calculations, Polymorphism

## 📝 Đề bài

Xây dựng hệ thống mạng lưới giao thông:

### Class Vehicle:

- id, maxSpeed (km/h), fuelEfficiency (km/liter)

### Class Car extends Vehicle:

- passengers
- Tốc độ thực tế =  $\text{maxSpeed} \times (1 - 0.05 \times \text{passengers})$  (mỗi người giảm 5%)
- Nhiên liệu tiêu thụ =  $\text{distance} / \text{fuelEfficiency}$

### Class Truck extends Vehicle:

- cargoWeight (tấn)
- Tốc độ thực tế =  $\text{maxSpeed} \times (1 - 0.1 \times \text{cargoWeight})$  (mỗi tấn giảm 10%)
- Nhiên liệu tiêu thụ =  $\text{distance} / \text{fuelEfficiency} \times (1 + 0.2 \times \text{cargoWeight})$

### Class Motorcycle extends Vehicle:

- Tốc độ thực tế =  $\text{maxSpeed}$
- Nhiên liệu tiêu thụ =  $\text{distance} / (\text{fuelEfficiency} \times 1.2)$  (hiệu quả hơn 20%)

**Yêu cầu:** Nhập N xe, mỗi xe di chuyển một quãng đường, tính thời gian và nhiên liệu.

## ◊ Input

Dòng 1: N ( $1 \leq N \leq 50$ )

N nhóm:

Car:

C <id> <maxSpeed> <fuelEff> <passengers> <distance>

Truck:

T <id> <maxSpeed> <fuelEff> <cargoWeight> <distance>

Motorcycle:

M <id> <maxSpeed> <fuelEff> <distance>

## ◊ Output

Từng chuyến đi:

```

Chuyen di <thu_tu>: <id> (<type>)
Quang duong: <distance>km
Toc do toi da: <maxSpeed>km/h
[Hanh khach: <passengers>]           ← Car
[Hang hoa: <cargoWeight>ton]          ← Truck
Toc do thuc te: <actualSpeed>km/h
Thoi gian: <time>h
Nhien lieu: <fuel>L

```

Thống kê:

```

==== THONG KE ====
Tong quang duong: <totalDistance>km
Tong thoi gian: <totalTime>h
Tong nhien lieu: <totalFuel>L

Trung binh:
- Toc do: <avgSpeed>km/h
- Hieu qua nhien lieu: <avgEfficiency>km/L

```

## Ví dụ

Test case 1

**Input:**

```

3
C CAR001 100 10 4 200
T TRU001 80 8 5 160
M MOT001 120 25 100

```

**Output:**

```

Chuyen di 1: CAR001 (Car)
Quang duong: 200km
Toc do toi da: 100km/h
Hanh khach: 4
Toc do thuc te: 80km/h
Thoi gian: 2.5h
Nhien lieu: 20.0L

```

```

Chuyen di 2: TRU001 (Truck)
Quang duong: 160km
Toc do toi da: 80km/h

```

Hang hoa: 5tan  
Toc do thuc te: 40km/h  
Thoi gian: 4.0h  
Nhien lieu: 40.0L

Chuyen di 3: MOT001 (Motorcycle)  
Quang duong: 100km  
Toc do toi da: 120km/h  
Toc do thuc te: 120km/h  
Thoi gian: 0.83h  
Nhien lieu: 3.33L

==== THONG KE ====  
Tong quang duong: 460km  
Tong thoi gian: 7.33h  
Tong nhien lieu: 63.33L

Trung binh:  
- Toc do: 62.74km/h  
- Hieu qua nhien lieu: 7.26km/L

## 💡 Gợi ý

### Tư duy:

- Override calculateActualSpeed() và calculateFuelConsumption()
- Mỗi loại xe có công thức riêng
- Thời gian = quãng đường / tốc độ thực tế

**Tags:** inheritance, override, complex-calculations, physics, advanced, competitive

---

# CodeForge - B23D - Thiết Kế Hệ Thống Hình Học

**Độ khó:** ★ ★ ★ Hard

**Loại:** ☀ Design (Advanced)

**Kiến thức:** Inheritance Design, Abstract Methods

## Đề bài

Thiết kế hệ thống hình học với kế thừa:

**Yêu cầu:** Thiết kế class hierarchy cho các hình 2D:

### 1. Class Shape (abstract concept):

- Không có fields cụ thể
- Phải có methods:
  - double getArea() - tính diện tích
  - double getPerimeter() - tính chu vi
  - String getType() - tên loại hình

### 2. Các class con:

- Circle: radius
- Rectangle: length, width
- Triangle: a, b, c (3 cạnh)
- Square: side (hình vuông extends Rectangle?)

**Thiết kế phải:**

- Tránh duplicate code
- Dùng inheritance hợp lý
- Override methods đúng
- Validation trong constructor

### ◊ Nhiệm vụ

1. Vẽ class diagram (text)
2. Implement các classes
3. Demo tạo 1 hình mỗi loại
4. Tính tổng diện tích

**Không có test cases cố định - Focus vào thiết kế!**

## Demo Expected

```
==== CLASS HIERARCHY ====
Shape (abstract)
└─ Circle
```

```
└── Rectangle
    └── Square
    └── Triangle
```

==== IMPLEMENTATION ====

(Code của bạn)

==== DEMO ====

```
Circle c = new Circle(5);
System.out.println(c.getType()); // "Circle"
System.out.println(c.getArea()); // 78.54
System.out.println(c.getPerimeter()); // 31.42
```

(Tương tự cho Rectangle, Square, Triangle)

## 💡 Gợi ý

### Câu hỏi thiết kế:

- Square có nên extends Rectangle không?
- Shape có nên là abstract class hay interface?
- Validation logic ở đâu (constructor, setters)?
- Có cần protected fields không?

**Tags:** design, inheritance, abstract, class-hierarchy, advanced

---

# CodeForge - B24D - Thiết Kế Hệ Thống Thư Viện

**Độ khó:** ★ ★ ★ ★ Very Hard

**Loại:** ☀ Design (Advanced)

**Kiến thức:** Complex Inheritance, Composition, Relationships

## Đề bài

Thiết kế hệ thống quản lý thư viện với inheritance:

**Yêu cầu:**

### 1. Class LibraryItem (base):

- Tất cả items có: id, title, status (available/borrowed)
- Methods: borrow(), return(), getInfo()

### 2. Các loại items:

- Book: author, pages, ISBN
- Magazine: issueNumber, publishDate
- DVD: duration, director

### 3. Class Member:

- id, name, borrowedItems[]
- Methods: borrowItem(), returnItem(), getHistory()

### 4. Class Librarian extends Member:

- employeeld
- Methods: addItem(), removeItem(), searchItem()

### 5. Class Library:

- items[], members[]
- Methods: register(), borrow(), return(), search()

**Thiết kế phải:**

- Sử dụng inheritance hợp lý
- Composition giữa các classes
- Tránh duplicate code
- Business logic validation

## ◊ Nhiệm vụ

1. Vẽ class diagram
2. Xác định relationships:
  - Inheritance (IS-A)

- Composition (HAS-A)

3. Implement các classes

4. Demo workflow: thêm sách → member mượn → trả

### **Không có test cases - Focus vào design patterns!**

## Demo Expected

```
==== CLASS DIAGRAM ====
```

```
LibraryItem
```

```
|--- Book  
|--- Magazine  
|--- DVD
```

```
Member
```

```
|--- Librarian
```

```
Library (HAS-A items[], members[])
```

```
==== RELATIONSHIPS ====
```

- Book IS-A LibraryItem (Inheritance)
- Library HAS-A items[] (Composition)
- Member HAS-A borrowedItems[] (Composition)
- Librarian IS-A Member (Inheritance)

```
==== DEMO ===
```

```
(Workflow code)
```

## Gợi ý

### Câu hỏi thiết kế:

- LibraryItem có nên abstract không?
- Member và Librarian quan hệ như thế nào?
- Validation ở đâu (borrow item đã mượn)?
- Có cần interface không?

**Tags:** design, inheritance, composition, relationships, library-system, advanced

---

# CodeForge - B25D - Thiết Kế Hệ Thống E-Commerce (CAPSTONE)

**Độ khó:** ★ ★ ★ ★ ★ Expert

**Loại:** 🎓 Design (CAPSTONE - Advanced)

**Kiến thức:** Full Inheritance, Polymorphism, Design Patterns

## 📝 Đề bài

**CAPSTONE PROJECT** - Thiết kế hệ thống E-commerce hoàn chỉnh:

### Yêu cầu hệ thống:

#### 1. Product Hierarchy:

- Product (base): id, name, price, stock
- Physical Products: Book, Electronics, Clothing
  - Book: author, pages, category
  - Electronics: brand, warranty
  - Clothing: size, color, material
- Digital Products: Ebook, Software
  - Ebook: fileSize, format
  - Software: version, license

#### 2. User Hierarchy:

- User (base): id, name, email
- Customer: address, cart, orders[]
- Seller: shopName, products[], revenue
- Admin: permissions, logs[]

#### 3. Order System:

- Order: orderId, customer, items[], status
- ShippingOrder (for physical): address, carrier, trackingNumber
- DigitalOrder (for digital): downloadLink, expiryDate

#### 4. Payment Integration:

- Payment (base): amount, status
- CashPayment, CardPayment, OnlinePayment

### Thiết kế phải có:

- Multi-level inheritance
- Polymorphism
- Encapsulation
- Method overriding
- Constructor chaining

- Validation logic
- Business rules

## ◊ Nhiệm vụ

1. **Class Diagram đầy đủ**
2. **Implement tất cả classes**

### 3. **Demo workflow:**

- Customer thêm sản phẩm vào cart
- Checkout với nhiều payment methods
- Seller nhận order
- Admin quản lý hệ thống

### 4. **Xử lý edge cases:**

- Out of stock
- Invalid payment
- Cancel order

## █ Design Expected

```
==== FULL CLASS HIERARCHY ====
```

```
Product
└── PhysicalProduct
    ├── Book
    ├── Electronics
    └── Clothing
└── DigitalProduct
    ├── EBook
    └── Software
```

```
User
└── Customer
└── Seller
└── Admin
```

```
Order
└── ShippingOrder
└── DigitalOrder
```

```
Payment
└── CashPayment
└── CardPayment
└── OnlinePayment
```

```
==== FEATURES ===
```

- ✓ Add to cart
- ✓ Checkout
- ✓ Multiple payment methods
- ✓ Order tracking

- ✓ Seller management
- ✓ Admin dashboard

## 💡 Gợi ý Thiết Kế

### Inheritance decisions:

- Product hierarchy: Physical vs Digital
- User roles: Customer, Seller, Admin
- Order types: Shipping vs Digital delivery
- Payment methods: Different fees

### Key methods to override:

- Product.calculateShippingFee()
- Order.process()
- Payment.processPayment()
- User.getPermissions()

### Business rules:

- Digital products: instant delivery
- Physical products: shipping required
- Seller: can only manage own products
- Admin: full system access

**Tags:** design, capstone, e-commerce, multi-level-inheritance, polymorphism, full-system, expert, advanced

---