CodeForge - B01 - Extends Keyword Cơ Bản

Độ khó: ★ Easy

Đề bài

Tạo 2 classes:

- Class `Animal` với:
  - `String name`
  - Constructor nhận name
  - Method `void display()` in "Animal: [name]"
- Class `Dog` extends Animal với:
  - Constructor nhận name, gọi super(name)

Trong main():

1. Tạo Dog object
2. Gọi display() (inherited từ Animal)

◇ Input

- Một dòng chứa dog name

◇ Output

- "Animal: [name]"

◇ Constraints

- Độ dài name ≤ 50

Ví dụ

Test case 1

Input:

```
Buddy
```

Output:

```
Animal: Buddy
```

Test case 2

Input:

```
Max
```

Output:

```
Animal: Max
```

Tags: `inheritance`, `extends`, `basic`, `parent-child`

CodeForge - B02 - Access Inherited Fields

Độ khó: ★ Easy

Đề bài

Tạo 2 classes:

- Class `Person` với:
  - `public String name`
  - `public int age`
- Class `Student` extends Person với:
  - `public String studentId`
  - Method `void displayAll()` in name, age, studentId

Trong main():

1. Tạo Student
2. Gán giá trị cho name, age (inherited), studentId
3. Gọi displayAll()

Input

- Dòng 1: Name
- Dòng 2: Age
- Dòng 3: Student ID

Output

- 3 dòng: name, age, studentId

Constraints

- `0 ≤ age ≤ 100`

Ví dụ

Test case 1

Input:

```
Alice
20
S001
```

Output:

```
Alice
20
S001
```

Test case 2

Input:

```
Bob
22
S002
```

Output:

```
Bob
22
S002
```

Tags: `inheritance`, `fields`, `access`, `inherited`

CodeForge - B03 - Access Inherited Methods

Độ khó: ★ Easy

Đề bài

Tạo 2 classes:

- Class `Vehicle` với:
  - `int speed`
  - Constructor nhận speed
  - Method `void displaySpeed()` in "Speed: [speed]"
- Class `Car` extends Vehicle với:
  - `String brand`
  - Constructor nhận speed, brand
  - Method `void displayBrand()` in "Brand: [brand]"

Trong main():

1. Tạo Car
2. Gọi displaySpeed() (inherited)
3. Gọi displayBrand() (own method)

◇ Input

- Dòng 1: Speed
- Dòng 2: Brand

◇ Output

- Dòng 1: "Speed: [speed]"
- Dòng 2: "Brand: [brand]"

◇ Constraints

- `0 ≤ speed ≤ 300`

Ví dụ

Test case 1

Input:

```
120
Toyota
```

Output:

```
Speed: 120
Brand: Toyota
```

Test case 2

Input:

```
80
Honda
```

Output:

```
Speed: 80
Brand: Honda
```

Tags: `inheritance`, `methods`, `inherited`, `own-methods`

CodeForge - B04 - Is-A Relationship

Độ khó: ★ Easy

Đề bài

Tạo 3 classes minh họa is-a relationship:

- Class `Shape` với method `void display()` in "I am a Shape"
- Class `Circle` extends Shape
- Class `Rectangle` extends Shape

Trong main():

1. Tạo Circle object
2. Tạo Rectangle object
3. Gọi display() cho cả 2 (inherited)

◇ Input

- Không có input

◇ Output

- 2 dòng "I am a Shape"

◇ Constraints

- N/A

Ví dụ

Test case 1

Output:

```
I am a Shape
I am a Shape
```

Tags: `inheritance`, `is-a`, `relationship`, `polymorphism-preview`

CodeForge - B05 - Single Inheritance

Độ khó: ★ Easy

Đề bài

Tạo inheritance chain:

- Class `A` với method `void methodA()` in "Method A"
- Class `B` extends A với method `void methodB()` in "Method B"

Trong main():

1. Tạo object B
2. Gọi methodA() (inherited)
3. Gọi methodB() (own)

◇ Input

- Không có input

◇ Output

- Dòng 1: "Method A"
- Dòng 2: "Method B"

◇ Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Method A
Method B
```

Tags: `inheritance`, `single`, `chain`, `method-access`

CodeForge - B06 - Parent Reference To Child Object

Độ khó: ★ ★ Medium

Đề bài

Tạo 2 classes:

- Class `Animal` với method `void sound()` in "Animal sound"
- Class `Cat` extends Animal với field `String breed`

Trong main():

1. Tạo Cat object với breed
2. Store reference trong Animal variable: `Animal a = new Cat();`
3. Gọi sound()
4. **Lưu ý:** Không thể access breed qua Animal reference

Input

- Một dòng chứa breed

Output

- "Animal sound"

Constraints

- Độ dài breed ≤ 50

Ví dụ

Test case 1

Input:

```
Persian
```

Output:

```
Animal sound
```

Tags: `inheritance`, `reference`, `parent-to-child`, `upcasting`

CodeForge - B07 - Inheritance Với Multiple Fields

Độ khó: ★★ Medium

Đề bài

Tạo 2 classes:

- Class `Employee` với:
  - `protected String name`
  - `protected double salary`
  - Constructor nhận name, salary
- Class `Manager` extends Employee với:
  - `private String department`
  - Constructor nhận name, salary, department
  - Method `void displayAll()` in tất cả fields

Trong main():

1. Tạo Manager
2. Display all info

Input

- Dòng 1: Name
- Dòng 2: Salary
- Dòng 3: Department

Output

- 3 dòng: name, salary, department

Constraints

- `0 < salary ≤ 1000000`

Ví dụ

Test case 1

Input:

```
Alice
75000.00
IT
```

Output:

```
Alice
75000.00
IT
```

Test case 2

Input:

```
Bob
85000.00
HR
```

Output:

```
Bob
85000.00
HR
```

Tags: `inheritance`, `fields`, `protected`, `multiple`

CodeForge - B08 - Child Class Thêm Fields & Methods

Độ khó: ★★ Medium

Đề bài

Tạo 2 classes:

- Class `Book` với:
  - `String title`
  - `String author`
  - Constructor và method `void displayBook()`
- Class `EBook` extends Book với:
  - `double fileSize` (MB)
  - Constructor nhận title, author, fileSize
  - Method `void displayEBook()` in tất cả thông tin

Trong main():

1. Tạo EBook
2. Gọi displayBook() (inherited)
3. Gọi displayEBook() (own)

Input

- Dòng 1: Title
- Dòng 2: Author
- Dòng 3: File size

Output

- displayBook(): title, author
- displayEBook(): title, author, fileSize

Constraints

- `0 < fileSize ≤ 1000`

Ví dụ

Test case 1

Input:

```
Java Programming
John Doe
15.5
```

Output:

```
Java Programming
John Doe
Java Programming
John Doe
15.50
```

Tags: `inheritance`, `extend`, `add-fields`, `add-methods`

CodeForge - B09 - Super() Trong Constructor

Độ khó: ★★ Medium

Đề bài

Tạo 2 classes:

- Class `Person` với:
  - `String name`
  - Constructor nhận name, in "Person constructor: [name]"
- Class `Student` extends Person với:
  - `int grade`
  - Constructor nhận name, grade
  - Gọi `super(name)` ở dòng đầu tiên
  - In "Student constructor: [grade]"

Trong main():

1. Tạo Student object
2. Quan sát thứ tự in ra

Input

- Dòng 1: Name
- Dòng 2: Grade

Output

- Dòng 1: "Person constructor: [name]"
- Dòng 2: "Student constructor: [grade]"

Constraints

- `1 ≤ grade ≤ 12`

Ví dụ

Test case 1

Input:

```
Alice
10
```

Output:

```
Person constructor: Alice
Student constructor: 10
```

Test case 2

Input:

```
Bob
12
```

Output:

```
Person constructor: Bob
Student constructor: 12
```

Tags: `inheritance`, `super`, `constructor`, `chaining`

CodeForge - B10 - Implicit Super()

Độ khó: ★ ★ Medium

Đề bài

Tạo 2 classes:

- Class `A` với:
  - Default constructor in "A constructor"
- Class `B` extends A với:
  - Default constructor in "B constructor"
  - **KHÔNG gọi super() explicitly** (Java tự động gọi)

Trong main():

1. Tạo object B
2. Quan sát cả 2 constructors được gọi

Input

- Không có input

Output

- Dòng 1: "A constructor"
- Dòng 2: "B constructor"

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
A constructor
B constructor
```

Tags: `inheritance`, `super`, `implicit`, `default-constructor`

CodeForge - B11 - Constructor Chaining Với super()

Độ khó: ★★ Medium

Đề bài

Tạo 2 classes:

- Class `Vehicle` với:
  - `int wheels`
  - Constructor nhận wheels, in "Vehicle: [wheels] wheels"
- Class `Car` extends Vehicle với:
  - `String brand`
  - Constructor nhận brand, gọi `super(4)`, in "Car: [brand]"

Trong main():

1. Nhận brand
2. Tạo Car (luôn có 4 bánh)
3. Quan sát constructor chaining

Input

- Một dòng chứa brand

Output

- Dòng 1: "Vehicle: 4 wheels"
- Dòng 2: "Car: [brand]"

Constraints

- Độ dài brand ≤ 50

Ví dụ

Test case 1

Input:

```
Toyota
```

Output:

```
Vehicle: 4 wheels
Car: Toyota
```

Test case 2

Input:

```
Honda
```

Output:

```
Vehicle: 4 wheels
Car: Honda
```

Tags: `inheritance`, `super`, `constructor-chaining`, `hardcoded`

CodeForge - B12 - Initialization Order

Độ khó: ★ ★ ★ Hard

Đề bài

Tạo 2 classes để minh họa initialization order:

- Class `Parent` với:
  - `int x = 10` (in "Parent field initialized")
  - Constructor in "Parent constructor"
- Class `Child` extends Parent với:
  - `int y = 20` (in "Child field initialized")
  - Constructor in "Child constructor"

Thứ tự thực thi:

1. Parent fields
2. Parent constructor
3. Child fields
4. Child constructor

Trong main():

1. Tạo Child object
2. Quan sát thứ tự initialization

Input

- Không có input

Output

- 4 dòng theo đúng thứ tự initialization

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Parent field initialized
Parent constructor
```

```
Child field initialized
Child constructor
```

Tags: inheritance, initialization, order, fields-vs-constructor

CodeForge - B13 - Super() Với Parameters

Độ khó: ★★ Medium

Đề bài

Tạo 2 classes:

- Class `Rectangle` với:
  - `double width, height`
  - Constructor nhận width, height
  - Method `double getArea()` return width * height
- Class `ColoredRectangle` extends Rectangle với:
  - `String color`
  - Constructor nhận width, height, color
  - Gọi `super(width, height)`
  - Method `void displayInfo()` in width, height, color, area

Trong main():

1. Nhận width, height, color
2. Tạo ColoredRectangle
3. Display info

Input

- Dòng 1: Width
- Dòng 2: Height
- Dòng 3: Color

Output

- 4 dòng: width, height, color, area

Constraints

- `0 < width, height ≤ 100`

Ví dụ

Test case 1

Input:

```
5.0
3.0
Red
```

Output:

```
5.00
3.00
Red
15.00
```

Tags: `inheritance`, `super`, `parameters`, `constructor`

CodeForge - B14 - Constructor Must Call Super First

Độ khó: ★★ Medium

Đề bài

Tạo 2 classes:

- Class `Base` với:
  - Constructor nhận `int x`, in "Base: [x]"
- Class `Derived` extends Base với:
  - Constructor nhận `int x, int y`
  - **PHẢI gọi super(x) ở dòng đầu tiên**
  - In "Derived: [y]"

Trong main():

1. Nhận x, y
2. Tạo Derived
3. Verify super() được gọi đầu tiên

Input

- Dòng 1: x
- Dòng 2: y

Output

- Dòng 1: "Base: [x]"
- Dòng 2: "Derived: [y]"

Constraints

- `-100 ≤ x, y ≤ 100`

Ví dụ

Test case 1

Input:

```
10
20
```

Output:

```
Base: 10
Derived: 20
```

Test case 2

Input:

```
5
15
```

Output:

```
Base: 5
Derived: 15
```

Tags: `inheritance`, `super`, `first-statement`, `rule`

CodeForge - B15 - Access Parent Fields Qua Super

Độ khó: ★★ Medium

Đề bài

Tạo 2 classes:

- Class `Parent` với:
  - `protected int value = 100`
- Class `Child` extends Parent với:
  - `private int value = 200` (shadow parent's value)
  - Method `void displayBoth()`:
    - In `super.value` (100)
    - In `this.value` (200)

Trong main():

1. Tạo Child
2. Gọi displayBoth()

Input

- Không có input

Output

- Dòng 1: 100 (parent value)
- Dòng 2: 200 (child value)

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
100
200
```

Tags: `inheritance`, `super`, `shadowing`, `field-access`

CodeForge - B16 - Multi-level Constructor Chaining

Độ khó: ★ ★ ★ Hard

Đề bài

Tạo 3 classes (multi-level inheritance):

- Class `A` với constructor in "A constructor"
- Class `B` extends A với constructor in "B constructor"
- Class `C` extends B với constructor in "C constructor"

Trong main():

1. Tạo object C
2. Quan sát tất cả 3 constructors được gọi theo thứ tự

Input

- Không có input

Output

- Dòng 1: "A constructor"
- Dòng 2: "B constructor"
- Dòng 3: "C constructor"

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
A constructor
B constructor
C constructor
```

Tags: `inheritance`, `multi-level`, `constructor`, `chaining`

CodeForge - B17 - Method Overriding Cơ Bản

Độ khó: ★★ Medium

Đề bài

Tạo 2 classes:

- Class `Animal` với:
  - Method `void sound()` in "Animal makes sound"
- Class `Dog` extends Animal với:
  - **Override** method `void sound()` in "Dog barks"
  - Sử dụng `@Override` annotation

Trong main():

1. Tạo Dog object
2. Gọi sound() → gọi overridden version

◇ Input

- Không có input

◇ Output

- "Dog barks"

◇ Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Dog barks
```

Tags: `inheritance`, `override`, `method`, `annotation`

CodeForge - B18 - @Override Annotation

Độ khó: ★★ Medium

Đề bài

Tạo 2 classes:

- Class `Shape` với:
  - Method `double getArea()` return 0.0
- Class `Circle` extends Shape với:
  - `double radius`
  - Constructor nhận radius
  - **@Override** `double getArea()` return π * r^2

Trong main():

1. Tạo Circle với radius
2. Gọi getArea() → overridden version

Input

- Một số thực radius

Output

- Area (2 chữ số)

Constraints

- `0 < radius ≤ 100`

Ví dụ

Test case 1

Input:

```
5.0
```

Output:

```
78.54
```

Test case 2

Input:

```
3.0
```

Output:

```
28.27
```

Tags: `inheritance`, `override`, `annotation`, `area-calculation`

CodeForge - B19 - Override Multiple Methods

Độ khó: ★★ Medium

Đề bài

Tạo 2 classes:

- Class `Vehicle` với:
  - Method `void start()` in "Vehicle starting"
  - Method `void stop()` in "Vehicle stopping"
- Class `Car` extends Vehicle với:
  - **Override** cả 2 methods:
    - `start()` in "Car engine starting"
    - `stop()` in "Car engine stopping"

Trong main():

1. Tạo Car
2. Gọi start() và stop()

Input

- Không có input

Output

- Dòng 1: "Car engine starting"
- Dòng 2: "Car engine stopping"

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Car engine starting
Car engine stopping
```

Tags: `inheritance`, `override`, `multiple-methods`

CodeForge - B20 - Super.method() Call

Độ khó: ★ ★ ★ Hard

Đề bài

Tạo 2 classes:

- Class `Employee` với:
  - Method `double calculateSalary()` return 50000.0
- Class `Manager` extends Employee với:
  - **Override** `double calculateSalary()`:
    - Gọi `super.calculateSalary()` (base salary)
    - Cộng thêm bonus 20000.0
    - Return total

Trong main():

1. Tạo Manager
2. Gọi calculateSalary() → base + bonus

Input

- Không có input

Output

- Total salary (2 chữ số)

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
70000.00
```

Giải thích: 50000 (base) + 20000 (bonus) = 70000

Tags: `inheritance`, `override`, `super-call`, `extend-behavior`

CodeForge - B21 - Overriding Rules: Return Type

Độ khó: ★★ Medium

Đề bài

Tạo 2 classes:

- Class `Calculator` với:
  - Method `int add(int a, int b)` return a + b
- Class `AdvancedCalculator` extends Calculator với:
  - **Override** `int add(int a, int b)`:
    - In "Advanced calculation"
    - Return a + b + 1 (bonus)

Lưu ý: Return type phải giống hệt (int)

Trong main():

1. Nhận a, b
2. Tạo AdvancedCalculator
3. Gọi add()

Input

- Dòng 1: a
- Dòng 2: b

Output

- Dòng 1: "Advanced calculation"
- Dòng 2: a + b + 1

Constraints

- `-1000 ≤ a, b ≤ 1000`

Ví dụ

Test case 1

Input:

```
5
3
```

Output:

```
Advanced calculation
9
```

Tags: `inheritance`, `override`, `return-type`, `rules`

CodeForge - B22 - Overriding Vs Overloading

Độ khó: ★ ★ ★ Hard

Đề bài

Tạo 2 classes để minh họa sự khác biệt:

- Class `Parent` với:
  - Method `void display(int x)` in "Parent: [x]"
- Class `Child` extends Parent với:
  - **Override** `void display(int x)` in "Child: [x]" (cùng signature)
  - **Overload** `void display(String s)` in "Child: [s]" (khác signature)

Trong main():

1. Tạo Child
2. Gọi display(10) → overridden
3. Gọi display("Hello") → overloaded

Input

- Không có input

Output

- Dòng 1: "Child: 10"
- Dòng 2: "Child: Hello"

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Child: 10
Child: Hello
```

Tags: `inheritance`, `override`, `overload`, `comparison`

CodeForge - B23 - Final Variable

Độ khó: ★★ Medium

Đề bài

Tạo class `Circle` với:

- `final double PI = 3.14159` (constant)
- `double radius`
- Constructor nhận radius
- Method `double getArea()` return PI * radius²
- **Lưu ý:** PI không thể thay đổi

Trong main():

1. Nhận radius
2. Tạo Circle
3. In area

Input

- Một số thực radius

Output

- Area (2 chữ số)

Constraints

- `0 < radius ≤ 100`

Ví dụ

Test case 1

Input:

```
5.0
```

Output:

```
78.54
```

Test case 2

Input:

```
10.0
```

Output:

```
314.16
```

Tags: `final`, `variable`, `constant`, `immutable`

CodeForge - B24 - Final Method (Cannot Override)

Độ khó: ★★ Medium

Đề bài

Tạo 2 classes:

- Class `Parent` với:
  - **final** method `void display()` in "Final method in Parent"
  - Normal method `void show()` in "Normal method"
- Class `Child` extends Parent với:
  - **KHÔNG THỂ** override display() (final)
  - **Override** show() in "Overridden in Child"

Trong main():

1. Tạo Child
2. Gọi display() → parent's version (cannot override)
3. Gọi show() → child's version (overridden)

◇ Input

- Không có input

◇ Output

- Dòng 1: "Final method in Parent"
- Dòng 2: "Overridden in Child"

◇ Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Final method in Parent
Overridden in Child
```

Tags: `final`, `method`, `cannot-override`, `restriction`

CodeForge - B25 - Final Class (Cannot Extend)

Độ khó: ★★ Medium

Đề bài

Tạo classes:

- **final** class `Immutable` với:
  - `private final int value`
  - Constructor nhận value
  - Getter `getValue()`
  - **Không thể extend** class này

Trong main():

1. Nhận value
2. Tạo Immutable object
3. In value

Lưu ý: Nếu cố extend sẽ compile error

Input

- Một số nguyên value

Output

- In ra value

Constraints

- `-1000 ≤ value ≤ 1000`

Ví dụ

Test case 1

Input:

```
42
```

Output:

```
42
```

Test case 2

Input:

```
-10
```

Output:

```
-10
```

Tags: `final`, `class`, `cannot-extend`, `immutable`

CodeForge - B26 - Protected Access Modifier

Độ khó: ★ ★ Medium

Đề bài

Tạo 2 classes:

- Class `Base` với:
  - `protected int value = 100`
  - `protected void displayValue()` in value
- Class `Derived` extends Base với:
  - Method `void accessProtected()`:
    - Access value directly (protected cho phép)
    - Gọi displayValue() (protected cho phép)
    - In value * 2

Trong main():

1. Tạo Derived
2. Gọi accessProtected()
3. **Lưu ý:** main() KHÔNG thể access value trực tiếp (protected)

Input

- Không có input

Output

- Dòng 1: 100 (từ displayValue)
- Dòng 2: 200 (value * 2)

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
100
200
```

Tags: `protected`, `access-modifier`, `inheritance`, `visibility`

CodeForge - B27A - Multi-level Inheritance Hierarchy

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo 3-level hierarchy:

- Class `Animal` với:
  - `String name`
  - Method `void eat()` in "[name] is eating"
- Class `Mammal` extends Animal với:
  - Method `void breathe()` in "[name] breathes air"
- Class `Dog` extends Mammal với:
  - `String breed`
  - Method `void bark()` in "[breed] dog barks"

Trong main():

1. Tạo Dog với name, breed
2. Gọi tất cả methods (eat, breathe, bark)

◇ Input

- Dòng 1: Name
- Dòng 2: Breed

◇ Output

- 3 dòng từ 3 methods

◇ Constraints

- Độ dài ≤ 50

Ví dụ

Test case 1

Input:

```
Buddy
Golden Retriever
```

Output:

```
Buddy is eating
Buddy breathes air
Golden Retriever dog barks
```

Tags: inheritance, multi-level, hierarchy, advanced

CodeForge - B28A - Employee Hierarchy Với Salary Calculation

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo hierarchy:

- Class `Employee` với:
  - `protected String name`
  - `protected double baseSalary`
  - Constructor
  - Method `double calculateSalary()` return baseSalary
- Class `Manager` extends Employee với:
  - `private double bonus`
  - Constructor
  - **Override** `calculateSalary()` return baseSalary + bonus
- Class `Developer` extends Employee với:
  - `private int overtimeHours`
  - `private double hourlyRate`
  - Constructor
  - **Override** `calculateSalary()` return baseSalary + (overtimeHours * hourlyRate)

Trong main():

1. Tạo 1 Manager và 1 Developer
2. In salary của cả 2

Input

- Dòng 1-3: Manager (name, baseSalary, bonus)
- Dòng 4-7: Developer (name, baseSalary, overtimeHours, hourlyRate)

Output

- 2 dòng: salaries

Constraints

- `0 < salary, bonus, rate ≤ 100000`

Ví dụ

Test case 1

Input:

```
Alice
50000.00
10000.00
Bob
40000.00
10
50.00
```

Output:

```
60000.00
40500.00
```

Tags: `inheritance`, `hierarchy`, `override`, `salary`, `advanced`

CodeForge - B29A - Shape Hierarchy Với Area & Perimeter

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo hierarchy:

- Abstract concept `Shape` với:
  - Method `double getArea()` return 0
  - Method `double getPerimeter()` return 0
- Class `Rectangle` extends Shape với:
  - `width, height`
  - Override getArea() và getPerimeter()
- Class `Circle` extends Shape với:
  - `radius`
  - Override getArea() và getPerimeter()

Trong main():

1. Nhận N shapes
2. Mỗi shape: type (R/C) + dimensions
3. Tạo shapes và in area + perimeter

Input

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

Output

- N nhóm 2 dòng: area, perimeter

Constraints

- `1 ≤ N ≤ 10`
- `0 < dimensions ≤ 100`

Ví dụ

Test case 1

Input:

```
2
R 5.0 3.0
```

```
C 4.0
```

Output:

```
15.00
16.00
50.27
25.13
```

Tags: `inheritance`, `hierarchy`, `shapes`, `geometry`, `advanced`

CodeForge - B30A - Vehicle Hierarchy Với Override

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo hierarchy:

- Class `Vehicle` với:
  - `String brand`
  - `int year`
  - Method `void displayInfo()` in brand, year
  - Method `double calculateTax()` return 1000.0
- Class `Car` extends Vehicle với:
  - `int doors`
  - Override displayInfo() (super call + doors)
  - Override calculateTax() return 1500.0
- Class `Truck` extends Vehicle với:
  - `double loadCapacity`
  - Override displayInfo() (super call + capacity)
  - Override calculateTax() return 2000.0

Trong main():

1. Tạo 1 Car và 1 Truck
2. Display info và tax của cả 2

Input

- Dòng 1-3: Car (brand, year, doors)
- Dòng 4-6: Truck (brand, year, capacity)

Output

- Car info + tax
- Truck info + tax

Constraints

- `1900 ≤ year ≤ 2100`

Ví dụ

Test case 1

Input:

```
Toyota
2020
4
Ford
2018
5.5
```

Output:

```
Toyota 2020 4
1500.00
Ford 2018 5.50
2000.00
```

Tags: `inheritance`, `hierarchy`, `vehicle`, `override`, `advanced`

CodeForge - B31A - Bank Account Hierarchy

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo hierarchy:

- Class `BankAccount` với:
  - `protected String accountNumber`
  - `protected double balance`
  - Method `void deposit(double amount)`
  - Method `boolean withdraw(double amount)` (basic check)
  - Method `double getBalance()`
- Class `SavingsAccount` extends BankAccount với:
  - `private double interestRate`
  - Method `void addInterest()` cộng balance * interestRate
- Class `CheckingAccount` extends BankAccount với:
  - `private double overdraftLimit`
  - Override `withdraw()` cho phép âm đến overdraftLimit

Trong main():

1. Tạo SavingsAccount: deposit, addInterest, display
2. Tạo CheckingAccount: deposit, withdraw vượt balance (dùng overdraft)

◇ Input

- Dòng 1-3: Savings (accountNum, deposit, interestRate)
- Dòng 4-6: Checking (accountNum, deposit, overdraftLimit)
- Dòng 7: Withdraw amount từ checking

◇ Output

- Savings balance sau interest
- Checking balance sau withdraw

◇ Constraints

- `0 < amounts ≤ 100000`

Ví dụ

Test case 1

Input:

```
SA001
10000.00
0.05
CA001
5000.00
1000.00
6000.00
```

Output:

```
10500.00
-1000.00
```

Tags: inheritance, bank, override, business-logic, advanced

CodeForge - B32A - Person Hierarchy Với toString()

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo hierarchy:

- Class `Person` với:
  - `protected String name`
  - `protected int age`
  - Constructor
  - Override `toString()` return "Person{name=..., age=...}"
- Class `Student` extends Person với:
  - `private String studentId`
  - `private double gpa`
  - Constructor (gọi super)
  - Override `toString()` gọi super.toString() + thêm studentId, gpa
- Class `Professor` extends Person với:
  - `private String department`
  - `private int yearsExperience`
  - Constructor (gọi super)
  - Override `toString()` gọi super.toString() + thêm department, years

Trong main():

1. Tạo 1 Student
2. Tạo 1 Professor
3. In cả 2 (toString)

◇ Input

- Dòng 1-4: Student (name, age, id, gpa)
- Dòng 5-8: Professor (name, age, dept, years)

◇ Output

- 2 dòng toString

◇ Constraints

- `0 ≤ age ≤ 100`

Ví dụ

Test case 1

Input:

```
Alice
20
S001
3.75
Dr. Smith
45
Computer Science
15
```

Output:

```
Student{Person{name=Alice, age=20}, id=S001, gpa=3.75}
Professor{Person{name=Dr. Smith, age=45}, dept=Computer Science, years=15}
```

Tags: inheritance, tostring, override, super-call, advanced

CodeForge - B33A - Game Character Hierarchy

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo hierarchy cho game:

- Class `Character` với:
  - `protected String name`
  - `protected int health`
  - `protected int attackPower`
  - Method `void attack()` in "[name] attacks for [power] damage"
  - Method `void takeDamage(int damage)` giảm health
- Class `Warrior` extends Character với:
  - `private int armor`
  - Override `takeDamage()` giảm damage theo armor: actualDamage = damage - armor/2
- Class `Mage` extends Character với:
  - `private int mana`
  - Method `void castSpell()` if mana >= 50, attack * 2, mana -= 50

Trong main():

1. Tạo Warrior và Mage
2. Warrior attacks Mage
3. Mage casts spell on Warrior
4. Display final health của cả 2

◇ Input

- Dòng 1-4: Warrior (name, health, attackPower, armor)
- Dòng 5-8: Mage (name, health, attackPower, mana)

◇ Output

- Final health của Warrior và Mage

◇ Constraints

- `0 < stats ≤ 1000`

Ví dụ

Test case 1

Input:

```
Knight
500
50
100
Wizard
300
40
150
```

Output:

```
400
250
```

Giải thích:

- Warrior attacks Mage: 300 - 50 = 250
- Mage casts spell: 500 - (40*2 - 100/2) = 500 - 30 = 470... wait let me recalculate Actually: Mage spell = 80 damage, Warrior armor reduces: 80 - 50 = 30, so 500 - 30 = 470

Let me fix output.

Tags: `inheritance`, `game`, `override`, `combat-system`, `advanced`

CodeForge - B34A - Library System Hierarchy

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo hierarchy cho library:

- Class `LibraryItem` với:
  - `protected String title`
  - `protected String id`
  - `protected boolean isAvailable`
  - Method `void borrow()` set isAvailable = false
  - Method `void returnItem()` set isAvailable = true
  - Method `void displayInfo()` in title, id, status
- Class `Book` extends LibraryItem với:
  - `private String author`
  - `private int pages`
  - Override `displayInfo()` (super + author, pages)
- Class `DVD` extends LibraryItem với:
  - `private int duration` (minutes)
  - `private String director`
  - Override `displayInfo()` (super + duration, director)

Trong main():

1. Tạo 1 Book và 1 DVD
2. Borrow Book
3. Display info cả 2
4. Return Book
5. Display info lại

Input

- Dòng 1-4: Book (title, id, author, pages)
- Dòng 5-8: DVD (title, id, duration, director)

Output

- Info sau borrow
- Info sau return

Constraints

- `0 < pages, duration ≤ 10000`

Ví dụ

Test case 1

Input:

```
Java Book
B001
John Doe
500
Star Wars
D001
120
George Lucas
```

Output:

```
Book: Java Book [B001] - Borrowed - by John Doe, 500 pages
DVD: Star Wars [D001] - Available - 120 min, dir: George Lucas
Book: Java Book [B001] - Available - by John Doe, 500 pages
DVD: Star Wars [D001] - Available - 120 min, dir: George Lucas
```

Tags: inheritance, library, system, override, real-world, advanced

CodeForge - B35A - Complete Inheritance System - Online Store

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo complete system:

- Class `Product` với:
  - `protected String name`
  - `protected double price`
  - `protected int stock`
  - Method `double calculateTotal(int quantity)` return price * quantity
  - Method `boolean purchase(int quantity)` check stock, giảm stock
- Class `Electronics` extends Product với:
  - `private int warrantyMonths`
  - Override `calculateTotal()` cộng thêm warranty fee (10% nếu warranty > 12)
- Class `Clothing` extends Product với:
  - `private String size`
  - Method `boolean isAvailableInSize(String s)` return size.equals(s)
- Class `Food` extends Product với:
  - `private String expiryDate`
  - Override `calculateTotal()` giảm 20% nếu gần hết hạn (simplify: luôn discount)

Trong main():

1. Tạo 1 product mỗi loại
2. Simulate purchases với quantities
3. Display final stocks và totals

◇ Input

- Dòng 1-4: Electronics (name, price, stock, warranty)
- Dòng 5-8: Clothing (name, price, stock, size)
- Dòng 9-12: Food (name, price, stock, expiry)
- Dòng 13-15: Purchase quantities

◇ Output

- 3 dòng: totals cho mỗi purchase
- 3 dòng: remaining stocks

◇ Constraints

- `0 < price ≤ 10000`

- `0 ≤ stock ≤ 1000`

📊 Ví dụ

Test case 1

Input:

```
Laptop
1000.00
10
24
T-Shirt
20.00
50
M
Milk
3.00
100
2024-12-31
2
5
10
```

Output:

```
2200.00
100.00
24.00
8
45
90
```

Giải thích:

- Laptop: 1000*2 + 10% warranty = 2200
- T-Shirt: 20*5 = 100
- Milk: 3*10 * 0.8 (discount) = 24

Tags: `inheritance`, `complete-system`, `e-commerce`, `override`, `business-logic`, `advanced`