

CodeForge - B01 - Class Rỗng & Object Creation

Độ khó: ★ Easy

Đề bài

Tạo class **Box** rỗng (không có fields, không có methods).

Trong main(), tạo một object của class **Box** và in ra "Object created successfully".

◊ Input

- Không có input

◊ Output

- In ra "Object created successfully"

◊ Constraints

- Class phải có tên là **Box**
- Tạo object bằng **new Box()**

Ví dụ

Test case 1

Input:

(không có)

Output:

Object created successfully

Tags: **oop, class, object, basic**

CodeForge - B02 - Class Với Một Field

Độ khó: ★ Easy

Đề bài

Tạo class **Car** với:

- Field public: **String brand**

Trong main():

1. Tạo object **Car**
2. Gán giá trị cho field **brand**
3. In ra giá trị của **brand**

◊ Input

- Một dòng chứa brand name

◊ Output

- In ra brand name

◊ Constraints

- Field phải là **public**
- **1 ≤ độ dài brand ≤ 50**

Ví dụ

Test case 1

Input:

```
Toyota
```

Output:

```
Toyota
```

Test case 2

Input:

Honda

Output:

Honda

Tags: oop, class, field, public

CodeForge - B03 - Class Với Multiple Fields

Độ khó: ★ Easy

Đề bài

Tạo class **Person** với:

- Field public: **String name**
- Field public: **int age**

Trong main():

1. Tạo object **Person**
2. Gán giá trị cho cả 2 fields
3. In ra name và age

◊ Input

- Dòng 1: Name
- Dòng 2: Age

◊ Output

- Dòng 1: Name
- Dòng 2: Age

◊ Constraints

- Fields phải là **public**
- **0 ≤ age ≤ 150**

Ví dụ

Test case 1

Input:

```
Alice  
25
```

Output:

```
Alice  
25
```

Test case 2

Input:

```
Bob  
30
```

Output:

```
Bob  
30
```

Tags: oop, class, multiple-fields, public

CodeForge - B04 - Access Object Fields

Độ khó: ★ Easy

Đề bài

Tạo class **Book** với:

- `public String title`
- `public String author`
- `public double price`

Trong main():

1. Tạo object, gán giá trị
2. In ra tất cả fields

◊ Input

- Dòng 1: Title
- Dòng 2: Author
- Dòng 3: Price

◊ Output

- Dòng 1: Title
- Dòng 2: Author
- Dòng 3: Price (làm tròn 2 chữ số thập phân)

◊ Constraints

- `0 < price ≤ 10000`

Ví dụ

Test case 1

Input:

```
Java Programming
John Doe
49.99
```

Output:

```
Java Programming  
John Doe  
49.99
```

Test case 2

Input:

```
Clean Code  
Robert Martin  
45.50
```

Output:

```
Clean Code  
Robert Martin  
45.50
```

Tags: oop, class, fields, access

CodeForge - B05 - Instance Method (Void, No Params)

Độ khó: ★ Easy

Đề bài

Tạo class **Dog** với:

- `public String name`
- Method `void bark()` in ra "Woof! Woof!"

Trong main():

1. Tạo object, gán name
2. Gọi method `bark()`

◊ Input

- Một dòng chứa dog name

◊ Output

- Dòng 1: Dog name
- Dòng 2: "Woof! Woof!"

◊ Constraints

- Method phải là instance method (không static)

Ví dụ

Test case 1

Input:

```
Buddy
```

Output:

```
Buddy
Woof! Woof!
```

Test case 2

Input:

```
Max
```

Output:

```
Max  
Woof! Woof!
```

Tags: oop, instance-method, void, no-params

CodeForge - B06 - Instance Method VỚI PARAMETERS

Độ khó: ★ Easy

Đề bài

Tạo class **Calculator** với:

- Method `void add(int a, int b)` in ra tổng $a + b$

Trong main():

- Tạo object
- Nhận 2 số từ input
- Gọi method `add()`

◊ Input

- Dòng 1: Số nguyên a
- Dòng 2: Số nguyên b

◊ Output

- In ra tổng $a + b$

◊ Constraints

- $-10^9 \leq a, b \leq 10^9$

Ví dụ

Test case 1

Input:

```
5
3
```

Output:

```
8
```

Test case 2

Input:

```
-10  
15
```

Output:

```
5
```

Tags: oop, instance-method, parameters, void

CodeForge - B07 - Instance Method Với Return Value

Độ khó: ★ Easy

Đề bài

Tạo class **MathHelper** với:

- Method `int multiply(int a, int b)` return `a * b`

Trong main():

1. Tạo object
2. Nhận 2 số từ input
3. Gọi method và in kết quả

◊ Input

- Dòng 1: Số nguyên a
- Dòng 2: Số nguyên b

◊ Output

- In ra tích a * b

◊ Constraints

- $-10^6 \leq a, b \leq 10^6$

Ví dụ

Test case 1

Input:

```
4
5
```

Output:

```
20
```

Test case 2

Input:

```
-3  
7
```

Output:

```
-21
```

Tags: oop, instance-method, return-value

CodeForge - B08 - Multiple Objects

Độ khó: ★ Easy

Đề bài

Tạo class **Student** với:

- `public String name`
- `public int score`

Trong main():

1. Tạo 3 objects Student
2. Nhập thông tin cho cả 3
3. In ra thông tin của student có điểm cao nhất

◊ Input

- 3 nhóm, mỗi nhóm 2 dòng:
 - Name
 - Score

◊ Output

- Dòng 1: Name của student có điểm cao nhất
- Dòng 2: Score của student đó

◊ Constraints

- `0 ≤ score ≤ 100`
- Không có 2 students cùng điểm

Ví dụ

Test case 1

Input:

```
Alice  
85  
Bob  
92  
Charlie  
78
```

Output:

```
Bob  
92
```

Test case 2

Input:

```
John  
70  
Jane  
95  
Jack  
88
```

Output:

```
Jane  
95
```

Tags: oop, multiple-objects, comparison

CodeForge - B09 - Default Constructor

Độ khó: ★ Easy

Đề bài

Tạo class **Counter** với:

- `public int count`
- Default constructor khởi tạo `count = 0`

Trong main():

1. Tạo object bằng default constructor
2. In ra giá trị count

◊ Input

- Không có input

◊ Output

- In ra `0`

◊ Constraints

- Phải tạo default constructor

Ví dụ

Test case 1

Output:

```
0
```

Tags: oop, constructor, default

CodeForge - B10 - Parameterized Constructor (1 Param)

Độ khó: ★ Easy

Đề bài

Tạo class `Circle` với:

- `public double radius`
- Constructor `Circle(double r)` gán `radius = r`

Trong main():

1. Nhận `radius` từ input
2. Tạo object bằng parameterized constructor
3. In ra `radius`

◊ Input

- Một số thực `radius`

◊ Output

- In ra `radius` (làm tròn 2 chữ số thập phân)

◊ Constraints

- `0 < radius ≤ 1000`

Ví dụ

Test case 1

Input:

```
5.5
```

Output:

```
5.50
```

Test case 2

Input:

```
10.0
```

Output:

```
10.00
```

Tags: oop, constructor, parameterized, one-param

CodeForge - B11 - Parameterized Constructor (Multiple Params)

Độ khó: ★ Easy

Đề bài

Tạo class `Rectangle` với:

- `public double width`
- `public double height`
- Constructor `Rectangle(double w, double h)`

Trong main():

1. Nhận width và height
2. Tạo object
3. In ra width và height

◊ Input

- Dòng 1: Width
- Dòng 2: Height

◊ Output

- Dòng 1: Width
- Dòng 2: Height

◊ Constraints

- `0 < width, height ≤ 1000`

Ví dụ

Test case 1

Input:

```
5.0
3.0
```

Output:

```
5.00  
3.00
```

Test case 2

Input:

```
10.5  
7.2
```

Output:

```
10.50  
7.20
```

Tags: oop, constructor, multiple-params

CodeForge - B12 - Constructor Overloading (2 Constructors)

Độ khó: ★★ Medium

Đề bài

Tạo class **Point** với:

- `public int x, y`
- Constructor `Point()` → $x=0, y=0$
- Constructor `Point(int x, int y)` → gán giá trị

Trong main():

1. Nhận N (số constructors sử dụng: 1 hoặc 2)
2. Nếu $N=1$: dùng default constructor
3. Nếu $N=2$: nhận x, y và dùng parameterized constructor
4. In ra x và y

◊ Input

- Dòng 1: N (1 hoặc 2)
- Nếu $N=2$: Dòng 2-3 là x, y

◊ Output

- Dòng 1: x
- Dòng 2: y

◊ Constraints

- $-1000 \leq x, y \leq 1000$

Ví dụ

Test case 1

Input:

```
1
```

Output:

```
0  
0
```

Test case 2

Input:

```
2  
5  
3
```

Output:

```
5  
3
```

Tags: oop, constructor, overloading

CodeForge - B13 - Constructor Overloading (3+ Constructors)

Độ khó: ★★ Medium

Đề bài

Tạo class **Box** với:

- `public double length, width, height`
- Constructor `Box()` → tất cả = 1.0
- Constructor `Box(double side)` → tất cả = side (cube)
- Constructor `Box(double l, double w, double h)` → gán từng giá trị

Trong main():

1. Nhận N (số parameters: 0, 1, hoặc 3)
2. Tạo object theo constructor phù hợp
3. In ra length, width, height

◊ Input

- Dòng 1: N (0, 1, hoặc 3)
- Nếu N=1: Dòng 2 là side
- Nếu N=3: Dòng 2-4 là length, width, height

◊ Output

- 3 dòng: length, width, height

◊ Constraints

- `0 < các giá trị ≤ 100`

Ví dụ

Test case 1

Input:

```
0
```

Output:

```
1.00  
1.00  
1.00
```

Test case 2

Input:

```
1  
5.0
```

Output:

```
5.00  
5.00  
5.00
```

Test case 3

Input:

```
3  
2.0  
3.0  
4.0
```

Output:

```
2.00  
3.00  
4.00
```

Tags: oop, constructor, overloading, multiple

CodeForge - B14 - This Keyword Trong Method

Độ khó: ★ ★ Medium

Đề bài

Tạo class `Employee` với:

- `public String name`
- `public double salary`
- Method `void display()` in ra name và salary sử dụng `this.name` và `this.salary`

Trong main():

1. Nhận name và salary
2. Tạo object
3. Gọi method display()

◊ Input

- Dòng 1: Name
- Dòng 2: Salary

◊ Output

- Dòng 1: Name
- Dòng 2: Salary (làm tròn 2 chữ số)

◊ Constraints

- `0 < salary ≤ 1000000`

Ví dụ

Test case 1

Input:

```
Alice  
50000.00
```

Output:

```
Alice  
50000.00
```

Test case 2

Input:

```
Bob  
75500.50
```

Output:

```
Bob  
75500.50
```

Tags: oop, this, method, keyword

CodeForge - B15 - This Keyword Trong Constructor

Độ khó: ★ ★ Medium

📝 Đề bài

Tạo class **Product** với:

- `private String name`
- `private double price`
- Constructor `Product(String name, double price)` sử dụng `this.name = name` và `this.price = price`
- Getter methods: `getName()`, `getPrice()`

Trong main():

1. Nhận name và price
2. Tạo object
3. In ra bằng getters

◊ Input

- Dòng 1: Name
- Dòng 2: Price

◊ Output

- Dòng 1: Name
- Dòng 2: Price

◊ Constraints

- `0 < price ≤ 100000`

📊 Ví dụ

Test case 1

Input:

```
Laptop  
999.99
```

Output:

```
Laptop  
999.99
```

Test case 2

Input:

```
Mouse  
25.50
```

Output:

```
Mouse  
25.50
```

Tags: oop, this, constructor, private, getter

CodeForge - B16 - Constructor Chaining (this())

Độ khó: ★ ★ Medium

📝 Đề bài

Tạo class `Account` với:

- `public String owner`
- `public double balance`
- Constructor `Account(String owner)` gọi `this(owner, 0.0)`
- Constructor `Account(String owner, double balance)` gán giá trị

Trong main():

1. Nhận N (1 hoặc 2 parameters)
2. Tạo object theo constructor phù hợp
3. In ra owner và balance

◊ Input

- Dòng 1: N
- Dòng 2: Owner
- Nếu N=2: Dòng 3 là balance

◊ Output

- Dòng 1: Owner
- Dòng 2: Balance

◊ Constraints

- `0 ≤ balance ≤ 1000000`

📊 Ví dụ

Test case 1

Input:

```
1
Alice
```

Output:

```
Alice  
0.00
```

Test case 2

Input:

```
2  
Bob  
5000.00
```

Output:

```
Bob  
5000.00
```

Tags: oop, constructor-chaining, this()

CodeForge - B17 - Private Fields + Public Methods

Độ khó: ★★ Medium

📝 Đề bài

Tạo class **Car** với:

- `private String brand`
- `private int year`
- Method `void setBrand(String b)` và `void setYear(int y)`
- Method `void display()` in ra brand và year

Trong main():

1. Tạo object
2. Dùng setters để gán giá trị
3. Gọi display()

◊ Input

- Dòng 1: Brand
- Dòng 2: Year

◊ Output

- Dòng 1: Brand
- Dòng 2: Year

◊ Constraints

- `1900 ≤ year ≤ 2100`

📊 Ví dụ

Test case 1

Input:

```
Toyota  
2024
```

Output:

```
Toyota  
2024
```

Test case 2

Input:

```
Honda  
2020
```

Output:

```
Honda  
2020
```

Tags: oop, private, encapsulation, setter

CodeForge - B18 - Getter Methods

Độ khó: ★ ★ Medium

📝 Đề bài

Tạo class `Person` với:

- `private String name`
- `private int age`
- Constructor để khởi tạo
- Getter methods: `getName()`, `getAge()`

Trong `main()`:

1. Nhận `name` và `age`
2. Tạo object bằng constructor
3. Dùng getters để in ra

◊ Input

- Dòng 1: Name
- Dòng 2: Age

◊ Output

- Dòng 1: Name
- Dòng 2: Age

◊ Constraints

- `0 ≤ age ≤ 150`

📊 Ví dụ

Test case 1

Input:

```
Alice  
25
```

Output:

```
Alice  
25
```

Test case 2

Input:

```
Bob  
30
```

Output:

```
Bob  
30
```

Tags: oop, getter, encapsulation, private

CodeForge - B19 - Setter Methods

Độ khó: ★ ★ Medium

📝 Đề bài

Tạo class `Temperature` với:

- `private double celsius`
- Default constructor → `celsius = 0`
- Setter `void setCelsius(double c)`
- Getter `double getCelsius()`

Trong main():

1. Tạo object (default constructor)
2. Nhận temperature từ input
3. Dùng setter để gán
4. Dùng getter để in ra

◊ Input

- Một số thực temperature

◊ Output

- In ra temperature (2 chữ số thập phân)

◊ Constraints

- `-273.15 ≤ celsius ≤ 1000`

💻 Ví dụ

Test case 1

Input:

```
25.5
```

Output:

```
25.50
```

Test case 2

Input:

```
-10.0
```

Output:

```
-10.00
```

Tags: oop, setter, getter, encapsulation

CodeForge - B20 - Getters + Setters (Full Encapsulation)

Độ khó: ★ ★ Medium

Đề bài

Tạo class **BankAccount** với:

- `private String accountNumber`
- `private double balance`
- Default constructor → `balance = 0`
- Setters: `setAccountNumber()`, `setBalance()`
- Getters: `getAccountNumber()`, `getBalance()`

Trong main():

1. Tạo object
2. Dùng setters gán giá trị
3. Dùng getters in ra

◊ Input

- Dòng 1: Account number
- Dòng 2: Balance

◊ Output

- Dòng 1: Account number
- Dòng 2: Balance

◊ Constraints

- `0 ≤ balance ≤ 10^9`

Ví dụ

Test case 1

Input:

```
ACC001
1000.00
```

Output:

```
ACC001  
1000.00
```

Test case 2

Input:

```
ACC002  
5500.50
```

Output:

```
ACC002  
5500.50
```

Tags: oop, full-encapsulation, getters-setters

CodeForge - B21 - Validation Trong Setter

Độ khó: ★ ★ Medium

📝 Đề bài

Tạo class **Age** với:

- **private int age**
- Setter **void setAge(int a)**: chỉ gán nếu $0 \leq a \leq 150$, nếu không gán **age = 0**
- Getter **int getAge()**

Trong main():

1. Tạo object
2. Nhận age từ input
3. Dùng setter gán
4. Dùng getter in ra

◊ Input

- Một số nguyên age

◊ Output

- In ra age (sau validation)

◊ Constraints

- Input có thể nằm ngoài range

💻 Ví dụ

Test case 1

Input:

```
25
```

Output:

```
25
```

Test case 2

Input:

```
200
```

Output:

```
0
```

Test case 3

Input:

```
-5
```

Output:

```
0
```

Tags: oop, setter, validation, encapsulation

CodeForge - B22 - Read-Only Fields (Getters Only)

Độ khó: ★★ Medium

📝 Đề bài

Tạo class `ImmutablePerson` với:

- `private String name`
- `private int birthYear`
- Constructor để khởi tạo
- Getters: `getName()`, `getBirthYear()`
- Method `int calculateAge(int currentYear)` return `currentYear - birthYear`
- **KHÔNG CÓ setters** (immutable)

Trong main():

1. Nhận name, birthYear, currentYear
2. Tạo object
3. In ra name, birthYear, và age

◊ Input

- Dòng 1: Name
- Dòng 2: Birth year
- Dòng 3: Current year

◊ Output

- Dòng 1: Name
- Dòng 2: Birth year
- Dòng 3: Age

◊ Constraints

- `1900 ≤ birthYear ≤ currentYear ≤ 2100`

📊 Ví dụ

Test case 1

Input:

```
Alice  
2000  
2024
```

Output:

```
Alice  
2000  
24
```

Test case 2

Input:

```
Bob  
1990  
2024
```

Output:

```
Bob  
1990  
34
```

Tags: oop, read-only, immutable, getter-only

CodeForge - B23 - Copy Constructor

Độ khó: ★ ★ Medium

📝 Đề bài

Tạo class `Book` với:

- `private String title`
- `private double price`
- Constructor `Book(String t, double p)`
- Copy constructor `Book(Book other)` copy từ object khác
- Getters: `getTitle()`, `getPrice()`

Trong main():

1. Nhận title và price
2. Tạo object book1
3. Tạo book2 từ book1 (copy constructor)
4. In ra thông tin book2

◊ Input

- Dòng 1: Title
- Dòng 2: Price

◊ Output

- Dòng 1: Title
- Dòng 2: Price

◊ Constraints

- `0 < price ≤ 10000`

📊 Ví dụ

Test case 1

Input:

```
Java Book  
99.99
```

Output:

```
Java Book  
99.99
```

Test case 2

Input:

```
Clean Code  
45.50
```

Output:

```
Clean Code  
45.50
```

Tags: oop, copy-constructor, constructor

CodeForge - B24 - Access Modifiers Demo

Độ khó: ★ ★ ★ Hard

📝 Đề bài

Tạo class **AccessDemo** với:

- `public int publicVar`
- `private int privateVar`
- Constructor khởi tạo cả 2
- `public int getPrivateVar()` getter cho `privateVar`
- Method `public void display()` in ra cả 2 variables

Trong main():

1. Tạo object
2. Access `publicVar` trực tiếp và in ra
3. Access `privateVar` qua getter và in ra

◊ Input

- Dòng 1: Giá trị cho `publicVar`
- Dòng 2: Giá trị cho `privateVar`

◊ Output

- Dòng 1: `publicVar` (accessed trực tiếp)
- Dòng 2: `privateVar` (accessed qua getter)

◊ Constraints

- `-1000 ≤ values ≤ 1000`

📊 Ví dụ

Test case 1

Input:

```
10
20
```

Output:

```
10  
20
```

Test case 2

Input:

```
-5  
15
```

Output:

```
-5  
15
```

Tags: oop, access-modifiers, public, private

CodeForge - B25 - Complex Constructor Chaining

Độ khó: ★ ★ ★ Hard

Đề bài

Tạo class **Student** với:

- `private String name`
- `private int age`
- `private double gpa`
- Constructor `Student()` gọi `this("Unknown", 0, 0.0)`
- Constructor `Student(String name)` gọi `this(name, 0, 0.0)`
- Constructor `Student(String name, int age)` gọi `this(name, age, 0.0)`
- Constructor `Student(String name, int age, double gpa)` gán tất cả
- Getters cho tất cả fields

Trong main():

1. Nhận N (số parameters: 0, 1, 2, hoặc 3)
2. Tạo object theo constructor phù hợp
3. In ra tất cả fields

◊ Input

- Dòng 1: N
- Nếu $N \geq 1$: Dòng 2 là name
- Nếu $N \geq 2$: Dòng 3 là age
- Nếu $N=3$: Dòng 4 là gpa

◊ Output

- 3 dòng: name, age, gpa

◊ Constraints

- $0 \leq age \leq 100$
- $0.0 \leq gpa \leq 4.0$

Ví dụ

Test case 1

Input:

```
0
```

Output:

```
Unknown  
0  
0.00
```

Test case 2

Input:

```
2  
Alice  
20
```

Output:

```
Alice  
20  
0.00
```

Test case 3

Input:

```
3  
Bob  
22  
3.75
```

Output:

```
Bob  
22  
3.75
```

Tags: oop, constructor-chaining, this(), complex

CodeForge - B26 - Immutable Object (Final Fields)

Độ khó: ★ ★ ★ Hard

📝 Đề bài

Tạo class **Point** với:

- `private final int x`
- `private final int y`
- Constructor để khởi tạo x, y (final fields phải init trong constructor)
- Getters: `getX()`, `getY()`
- Method `double distanceFromOrigin()` return khoảng cách từ (0,0)
- **KHÔNG CÓ setters** vì fields là final

Trong main():

1. Nhận x, y
2. Tạo object
3. In ra x, y và distance

◊ Input

- Dòng 1: x
- Dòng 2: y

◊ Output

- Dòng 1: x
- Dòng 2: y
- Dòng 3: Distance (làm tròn 2 chữ số)

◊ Constraints

- `-1000 ≤ x, y ≤ 1000`

💻 Ví dụ

Test case 1

Input:

```
3  
4
```

Output:

```
3  
4  
5.00
```

Test case 2

Input:

```
0  
5
```

Output:

```
0  
5  
5.00
```

Tags: oop, immutable, final, encapsulation

CodeForge - B27A - Person Class (Full Implementation)

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo class **Person** với:

- `private String name`
- `private int age`
- `private String email`
- Constructor đầy đủ
- Copy constructor
- Getters/setters với validation:
 - age: 0-150
 - email: phải chứa '@'
- Method `void displayInfo()` in tất cả thông tin

Trong main():

1. Nhận thông tin person1
2. Tạo person1
3. Tạo person2 từ person1 (copy constructor)
4. Modify person2 qua setters
5. Display cả 2 persons

◊ Input

- Dòng 1-3: name, age, email của person1
- Dòng 4-6: new values cho person2

◊ Output

- 3 dòng: person1 info
- 3 dòng: person2 info

◊ Constraints

- Email phải chứa '@'
- Age: 0-150

Ví dụ

Test case 1

Input:

```
Alice  
25  
alice@email.com  
Bob  
30  
bob@email.com
```

Output:

```
Alice  
25  
alice@email.com  
Bob  
30  
bob@email.com
```

Tags: oop, person, validation, copy-constructor, advanced

CodeForge - B28A - Student Class (GPA Calculation)

Độ khó: ★ ★ ★ Hard (Advanced)

📝 Đề bài

Tạo class **Student** với:

- `private String name`
- `private double[] scores` (3 môn)
- Constructor nhận name và 3 điểm
- Getters cho name
- Method `double calculateGPA()` return trung bình 3 môn
- Method `String getGrade()` return:
 - "A" nếu GPA ≥ 3.5
 - "B" nếu GPA ≥ 2.5
 - "C" nếu GPA ≥ 1.5
 - "D" nếu GPA ≥ 1.0
 - "F" nếu GPA < 1.0

Trong main():

1. Nhận name và 3 điểm
2. Tạo student
3. In ra name, GPA, và grade

◊ Input

- Dòng 1: Name
- Dòng 2-4: 3 scores

◊ Output

- Dòng 1: Name
- Dòng 2: GPA (2 chữ số)
- Dòng 3: Grade

◊ Constraints

- `0.0 ≤ scores ≤ 4.0`

📊 Ví dụ

Test case 1

Input:

```
Alice  
3.5  
3.8  
3.6
```

Output:

```
Alice  
3.63  
A
```

Test case 2**Input:**

```
Bob  
2.0  
2.5  
2.3
```

Output:

```
Bob  
2.27  
C
```

Tags: oop, student, gpa, calculation, advanced

CodeForge - B29A - BankAccount (Deposit/Withdraw)

Độ khó: ★ ★ ★ Hard (Advanced)

📝 Đề bài

Tạo class `BankAccount` với:

- `private String accountNumber`
- `private double balance`
- Constructor khởi tạo `accountNumber` và `balance = 0`
- Method `void deposit(double amount)`: cộng vào `balance` (nếu `amount > 0`)
- Method `boolean withdraw(double amount)`:
 - Trừ `balance` nếu `amount > 0` và `balance` đủ, return true
 - Nếu không đủ hoặc `amount ≤ 0`, return false
- Getter `getBalance()`

Trong `main()`:

1. Tạo account
2. Nhận N operations
3. Mỗi operation: "D amount" (deposit) hoặc "W amount" (withdraw)
4. In ra `balance` cuối cùng

◊ Input

- Dòng 1: Account number
- Dòng 2: N operations
- N dòng: operation type và amount

◊ Output

- Balance cuối cùng (2 chữ số)

◊ Constraints

- `1 ≤ N ≤ 100`
- `0 < amount ≤ 10000`

📊 Ví dụ

Test case 1

Input:

```
ACC001
3
```

```
D 1000.00  
W 300.00  
D 500.00
```

Output:

```
1200.00
```

Test case 2**Input:**

```
ACC002  
4  
D 500.00  
W 200.00  
W 400.00  
D 100.00
```

Output:

```
400.00
```

Giải thích: $500 - 200 = 300$, W 400 failed (không đủ), + 100 = 400

Tags: oop, bank-account, deposit, withdraw, advanced

CodeForge - B30A - Rectangle (Area & Perimeter)

Độ khó: ★ ★ ★ Hard (Advanced)

📝 Đề bài

Tạo class `Rectangle` với:

- `private double width`
- `private double height`
- Constructor với validation: `width, height > 0`
- Getters/setters với validation
- Method `double getArea()` return `width * height`
- Method `double getPerimeter()` return `2 * (width + height)`
- Method `boolean isSquare()` return true nếu `width == height`

Trong main():

1. Nhận `width` và `height`
2. Tạo `rectangle`
3. In ra area, perimeter, và có phải square không

◊ Input

- Dòng 1: Width
- Dòng 2: Height

◊ Output

- Dòng 1: Area (2 chữ số)
- Dòng 2: Perimeter (2 chữ số)
- Dòng 3: "SQUARE" hoặc "NOT SQUARE"

◊ Constraints

- `0 < width, height ≤ 1000`

📊 Ví dụ

Test case 1

Input:

```
5.0
5.0
```

Output:

```
25.00  
20.00  
SQUARE
```

Test case 2

Input:

```
4.0  
6.0
```

Output:

```
24.00  
20.00  
NOT SQUARE
```

Tags: oop, rectangle, geometry, calculation, advanced

CodeForge - B31A - Circle (Area & Circumference)

Độ khó: ★ ★ ★ Hard (Advanced)

📝 Đề bài

Tạo class `Circle` với:

- `private double radius`
- Constructor với validation: `radius > 0`
- Getter/setter với validation
- Method `double getArea()` return $\pi * r^2$
- Method `double getCircumference()` return $2 * \pi * r$
- Method `double getDiameter()` return $2 * r$
- Sử dụng `Math.PI`

Trong main():

1. Nhận radius
2. Tạo circle
3. In ra area, circumference, diameter

◊ Input

- Một số thực radius

◊ Output

- Dòng 1: Area (2 chữ số)
- Dòng 2: Circumference (2 chữ số)
- Dòng 3: Diameter (2 chữ số)

◊ Constraints

- `0 < radius ≤ 1000`

💻 Ví dụ

Test case 1

Input:

```
5.0
```

Output:

```
78.54  
31.42  
10.00
```

Test case 2

Input:

```
10.0
```

Output:

```
314.16  
62.83  
20.00
```

Tags: oop, circle, geometry, math, advanced

CodeForge - B32A - Point (Distance Calculation)

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo class **Point** với:

- **private int x, y**
- Constructor khởi tạo x, y
- Getters cho x, y
- Method **double distanceTo(Point other)** tính khoảng cách đến point khác
 - Công thức: $\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$
- Method **double distanceFromOrigin()** tính khoảng cách từ (0, 0)
- Method **boolean isOnAxis()** return true nếu x=0 hoặc y=0

Trong main():

1. Nhận tọa độ 2 points
2. Tạo 2 objects
3. In ra khoảng cách giữa chúng
4. In ra khoảng cách mỗi point từ origin
5. Check xem có point nào trên trực không

◊ Input

- Dòng 1-2: x, y của point1
- Dòng 3-4: x, y của point2

◊ Output

- Dòng 1: Distance giữa 2 points
- Dòng 2: Distance point1 từ origin
- Dòng 3: Distance point2 từ origin
- Dòng 4: "YES" nếu có point trên trực, "NO" nếu không

◊ Constraints

- $-1000 \leq x, y \leq 1000$

Ví dụ

Test case 1

Input:

```
0  
0  
3  
4
```

Output:

```
5.00  
0.00  
5.00  
YES
```

Test case 2**Input:**

```
1  
1  
4  
5
```

Output:

```
5.00  
1.41  
6.40  
NO
```

Tags: [oop](#), [point](#), [distance](#), [geometry](#), [advanced](#)

CodeForge - B33A - Time (Custom Time Class)

Độ khó: ★ ★ ★ Hard (Advanced)

📝 Đề bài

Tạo class `Time` với:

- `private int hours, minutes, seconds`
- Constructor với validation:
 - $0 \leq \text{hours} < 24$
 - $0 \leq \text{minutes} < 60$
 - $0 \leq \text{seconds} < 60$
 - Nếu invalid, set về 0
- Getters/setters với validation
- Method `int toSeconds()` convert to total seconds
- Method `String toStandardFormat()` return "HH:MM:SS"
- Method `void addSeconds(int s)` cộng s giây (handle overflow)

Trong main():

1. Nhận time (hours, minutes, seconds)
2. Tạo Time object
3. Nhận N giây để cộng
4. Add seconds
5. In ra time mới theo format HH:MM:SS

◊ Input

- Dòng 1-3: hours, minutes, seconds
- Dòng 4: N (giây cần cộng)

◊ Output

- Time sau khi cộng (HH:MM:SS)

◊ Constraints

- $0 \leq \text{initial time} < 24:00:00$
- $0 \leq N \leq 100000$

📊 Ví dụ

Test case 1

Input:

```
10  
30  
45  
75
```

Output:

```
10:32:00
```

Giải thích: 10:30:45 + 75s = 10:32:00

Test case 2

Input:

```
23  
59  
50  
20
```

Output:

```
00:00:10
```

Giải thích: Qua ngày mới

Tags: oop, time, validation, overflow, advanced

CodeForge - B34A - Book (With Discount)

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo class **Book** với:

- `private String title`
- `private String author`
- `private double price`
- `private int quantity`
- Constructor đầy đủ
- Getters/setters với validation (`price > 0, quantity ≥ 0`)
- Method `double calculateTotal()` return `price * quantity`
- Method `double calculateDiscountedTotal(double discountPercent)`
 - Return total sau discount ($0 \leq \text{discount} \leq 100$)
- Method `boolean isInStock()` return `quantity > 0`

Trong main():

1. Nhận book info và discount percent
2. Tạo book
3. In ra total, discounted total, stock status

◊ Input

- Dòng 1: Title
- Dòng 2: Author
- Dòng 3: Price
- Dòng 4: Quantity
- Dòng 5: Discount percent

◊ Output

- Dòng 1: Total (2 chữ số)
- Dòng 2: Discounted total (2 chữ số)
- Dòng 3: "IN STOCK" hoặc "OUT OF STOCK"

◊ Constraints

- `0 < price ≤ 10000`
- `0 ≤ quantity ≤ 1000`
- `0 ≤ discount ≤ 100`

Ví dụ

Test case 1

Input:

```
Java Book  
John Doe  
50.00  
10  
20.0
```

Output:

```
500.00  
400.00  
IN STOCK
```

Test case 2

Input:

```
Clean Code  
Robert Martin  
45.50  
0  
10.0
```

Output:

```
0.00  
0.00  
OUT OF STOCK
```

Tags: oop, book, discount, inventory, advanced

CodeForge - B35A - Product Inventory System

Độ khó: ★★☆ Hard (Advanced)

Đề bài

Tạo class **Product** với:

- `private String code`
- `private String name`
- `private double price`
- `private int stock`
- Constructor đầy đủ
- Getters/setters với validation
- Method `void addStock(int amount)` tăng stock
- Method `boolean sellProduct(int quantity)` giảm stock nếu đủ, return success/fail
- Method `double getTotalValue()` return price * stock

Trong main():

1. Nhận N products
2. Nhập info cho từng product
3. Nhận M operations (A: add stock, S: sell)
4. In ra tổng value của tất cả products

◊ Input

- Dòng 1: N (số products)
- N nhóm 4 dòng: code, name, price, initial stock
- Dòng N+2: M (số operations)
- M dòng: operation (format: "productCode A/S amount")

◊ Output

- Total inventory value (2 chữ số)

◊ Constraints

- `1 ≤ N ≤ 10`
- `1 ≤ M ≤ 50`
- `0 < price ≤ 10000`
- `0 ≤ stock ≤ 1000`

Ví dụ

Test case 1

Input:

```
2
P001
Laptop
1000.00
5
P002
Mouse
25.00
10
3
P001 A 2
P002 S 3
P001 S 1
```

Output:

```
6175.00
```

Giải thích:

- P001: $5+2-1=6$, value=6000
 - P002: $10-3=7$, value=175
 - Total=6175
-

Tags: oop, inventory, multiple-objects, system, advanced