CodeForge - B01 - Static Variable Đơn Giản

Độ khó: ★ Easy

Đề bài

Tạo class `Counter` với:

- `static int count` (khởi tạo = 0)
- Method `static void increment()` tăng count lên 1
- Method `static int getCount()` return giá trị count

Trong main():

1. Gọi increment() N lần
2. In ra count

◇ Input

- Một số nguyên N (số lần gọi increment)

◇ Output

- In ra giá trị count

◇ Constraints

- `1 ≤ N ≤ 1000`

Ví dụ

Test case 1

Input:

```
5
```

Output:

```
5
```

Test case 2

Input:

```
10
```

Output:

```
10
```

Test case 3

Input:

```
1
```

Output:

```
1
```

Tags: `static`, `variable`, `class-variable`, `basic`

CodeForge - B02 - Static Variable Shared Giữa Objects

Độ khó: ★ Easy

Đề bài

Tạo class `Student` với:

- `static int totalStudents = 0`
- `String name`
- Constructor nhận name, tăng totalStudents lên 1
- `static int getTotalStudents()` return totalStudents

Trong main():

1. Nhận N
2. Tạo N objects Student
3. In ra tổng số students

◇ Input

- Dòng 1: N (số students)
- N dòng tiếp: names

◇ Output

- In ra tổng số students

◇ Constraints

- `1 ≤ N ≤ 100`

Ví dụ

Test case 1

Input:

```
3
Alice
Bob
Charlie
```

Output:

```
3
```

Test case 2

Input:

```
5
John
Jane
Jack
Jill
James
```

Output:

```
5
```

Tags: `static`, `shared`, `counter`, `constructor`

CodeForge - B03 - Static Vs Instance Variables

Độ khó: ★ Easy

Đề bài

Tạo class `Car` với:

- `static int totalCars = 0` (class variable)
- `String brand` (instance variable)
- Constructor nhận brand, tăng totalCars
- Method `void display()` in ra brand và totalCars

Trong main():

1. Tạo 2 objects Car với brands khác nhau
2. Gọi display() cho cả 2

◇ Input

- Dòng 1: Brand của car1
- Dòng 2: Brand của car2

◇ Output

- 2 dòng: brand và totalCars của mỗi car

◇ Constraints

- Độ dài brand ≤ 50

Ví dụ

Test case 1

Input:

```
Toyota
Honda
```

Output:

```
Toyota 1
Honda 2
```

Giải thích: Car1 thấy total=1, Car2 thấy total=2

Test case 2

Input:

```
BMW
Audi
```

Output:

```
BMW 1
Audi 2
```

Tags: `static`, `instance`, `comparison`, `shared-state`

CodeForge - B04 - Access Static Qua Class Name

Độ khó: ★ Easy

Đề bài

Tạo class `MathUtils` với:

- `static double PI = 3.14159`
- `static int square(int n)` return n^2
- `static int cube(int n)` return n^3

Trong main():

1. Access PI qua `MathUtils.PI`
2. Nhận số N
3. Gọi square và cube qua `MathUtils.square()` và `MathUtils.cube()`
4. In ra kết quả

Input

- Một số nguyên N

Output

- Dòng 1: PI
- Dòng 2: Square của N
- Dòng 3: Cube của N

Constraints

- $1 \leq N \leq 100$

Ví dụ

Test case 1

Input:

```
5
```

Output:

```
3.14159
25
125
```

Test case 2

Input:

```
3
```

Output:

```
3.14159
9
27
```

Tags: `static`, `class-name-access`, `utility`

CodeForge - B05 - Modify Static Variable Qua Objects

Độ khó: ★ Easy

Đề bài

Tạo class `BankAccount` với:

- `static double interestRate = 5.0` (%)
- `String accountNumber`
- Constructor nhận accountNumber
- `static void setInterestRate(double rate)` modify interestRate
- `static double getInterestRate()` return interestRate

Trong main():

1. Tạo 2 accounts
2. Thay đổi interestRate qua static method
3. In ra interestRate từ cả 2 accounts (cùng giá trị)

◇ Input

- Dòng 1: Account1 number
- Dòng 2: Account2 number
- Dòng 3: New interest rate

◇ Output

- Dòng 1: Interest rate từ account1
- Dòng 2: Interest rate từ account2

◇ Constraints

- `0 < interestRate ≤ 20.0`

Ví dụ

Test case 1

Input:

```
ACC001
ACC002
7.5
```

Output:

```
7.50
7.50
```

Giải thích: Cả 2 accounts thấy cùng interestRate

Test case 2

Input:

```
ACC003
ACC004
6.0
```

Output:

```
6.00
6.00
```

Tags: `static`, `modify`, `shared`, `class-variable`

CodeForge - B06 - Count Objects Created

Độ khó: ★★ Medium

Đề bài

Tạo class `Product` với:

- `static int objectCount = 0`
- `String name`
- `int id` (tự động gán = objectCount khi tạo)
- Constructor nhận name, tăng objectCount, gán id
- Method `void display()` in ra id và name

Trong main():

1. Nhận N
2. Tạo N products
3. Display tất cả products
4. In ra tổng số products created

◇ Input

- Dòng 1: N
- N dòng: product names

◇ Output

- N dòng: id và name của mỗi product
- Dòng cuối: Total count

◇ Constraints

- `1 ≤ N ≤ 50`

Ví dụ

Test case 1

Input:

```
3
Laptop
Mouse
Keyboard
```

Output:

```
1 Laptop
2 Mouse
3 Keyboard
Total: 3
```

Test case 2

Input:

```
2
Book
Pen
```

Output:

```
1 Book
2 Pen
Total: 2
```

Tags: `static`, `counter`, `auto-increment`, `tracking`

CodeForge - B07 - Static Method Đơn Giản

Độ khó: ★ Easy

Đề bài

Tạo class `Calculator` với:

- `static int add(int a, int b)` return a + b
- `static int subtract(int a, int b)` return a - b
- `static int multiply(int a, int b)` return a * b

Trong main():

1. Nhận 2 số a, b
2. Gọi tất cả static methods
3. In ra kết quả

Input

- Dòng 1: a
- Dòng 2: b

Output

- Dòng 1: a + b
- Dòng 2: a - b
- Dòng 3: a * b

Constraints

- `-1000 ≤ a, b ≤ 1000`

Ví dụ

Test case 1

Input:

```
10
5
```

Output:

```
15
5
```

```
50
```

Test case 2

Input:

```
-3
7
```

Output:

```
4
-10
-21
```

Tags: `static`, `method`, `utility`, `basic`

CodeForge - B08 - Static Method Với Parameters

Độ khó: ★ Easy

Đề bài

Tạo class `StringUtils` với:

- `static int countVowels(String str)` đếm nguyên âm (a,e,i,o,u)
- `static boolean isPalindrome(String str)` kiểm tra palindrome
- `static String reverse(String str)` đảo ngược chuỗi

Trong main():

1. Nhận chuỗi
2. Gọi tất cả methods
3. In ra kết quả

Input

- Một chuỗi

Output

- Dòng 1: Số nguyên âm
- Dòng 2: "YES" hoặc "NO" (palindrome)
- Dòng 3: Chuỗi đảo ngược

Constraints

- `1 ≤ độ dài ≤ 100`
- Chỉ chữ cái thường

Ví dụ

Test case 1

Input:

```
hello
```

Output:

```
2
NO
olleh
```

Test case 2

Input:

```
racecar
```

Output:

```
3
YES
racecar
```

Tags: `static`, `method`, `string-utils`, `parameters`

CodeForge - B09 - Static Method Với Return Value

Độ khó: ★ Easy

Đề bài

Tạo class `ArrayUtils` với:

- `static int findMax(int[] arr)` tìm max
- `static int findMin(int[] arr)` tìm min
- `static double findAverage(int[] arr)` tính trung bình

Trong main():

1. Nhận N và mảng N phần tử
2. Gọi tất cả methods
3. In ra kết quả

Input

- Dòng 1: N
- Dòng 2: N số nguyên

Output

- Dòng 1: Max
- Dòng 2: Min
- Dòng 3: Average (2 chữ số)

Constraints

- `1 ≤ N ≤ 100`
- `-1000 ≤ phần tử ≤ 1000`

Ví dụ

Test case 1

Input:

```
5
3 7 2 9 5
```

Output:

```
9
2
5.20
```

Test case 2

Input:

```
3
-5 0 10
```

Output:

```
10
-5
1.67
```

Tags: `static`, `method`, `array`, `return-value`

CodeForge - B10 - Static Method Gọi Static Variable

Độ khó: ★★ Medium

Đề bài

Tạo class `Circle` với:

- `static double PI = 3.14159`
- `static double calculateArea(double radius)` return PI * r^2
- `static double calculateCircumference(double radius)` return 2 _ PI _ r

Trong main():

1. Nhận radius
2. Gọi cả 2 methods
3. In ra area và circumference

Input

- Một số thực radius

Output

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

Constraints

- `0 < radius ≤ 1000`

Ví dụ

Test case 1

Input:

```
5.0
```

Output:

```
78.54
31.42
```

Test case 2

Input:

```
10.0
```

Output:

```
314.16
62.83
```

Tags: `static`, `method`, `variable`, `access`

CodeForge - B11 - Static Method KHÔNG Access Instance Variable

Độ khó: ★★ Medium

Đề bài

Tạo class `Demo` với:

- `int instanceVar = 10` (instance variable)
- `static int staticVar = 20` (static variable)
- `static void staticMethod()` in ra staticVar (OK)
  - KHÔNG THỂ access instanceVar (compile error nếu thử)
- `void instanceMethod()` in ra cả instanceVar và staticVar (OK)

Trong main():

1. Gọi staticMethod()
2. Tạo object, gọi instanceMethod()

Input

- Không có input

Output

- Dòng 1: staticVar (từ staticMethod)
- Dòng 2: instanceVar và staticVar (từ instanceMethod)

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
20
10 20
```

Tags: `static`, `instance`, `access-restriction`, `demo`

CodeForge - B12 - Instance Method Gọi Static Method

Độ khó: ★★ Medium

Đề bài

Tạo class `Employee` với:

- `String name`
- `static double taxRate = 0.1` (10%)
- Constructor nhận name
- `static double calculateTax(double salary)` return salary * taxRate
- `void displaySalaryInfo(double salary)` in ra name, salary, và tax (gọi calculateTax)

Trong main():

1. Nhận name và salary
2. Tạo employee
3. Gọi displaySalaryInfo

Input

- Dòng 1: Name
- Dòng 2: Salary

Output

- Dòng 1: Name
- Dòng 2: Salary (2 chữ số)
- Dòng 3: Tax (2 chữ số)

Constraints

- `0 < salary ≤ 1000000`

Ví dụ

Test case 1

Input:

```
Alice
50000.00
```

Output:

```
Alice
50000.00
5000.00
```

Test case 2

Input:

```
Bob
75000.00
```

Output:

```
Bob
75000.00
7500.00
```

Tags: `static`, `instance`, `method-call`, `interaction`

CodeForge - B13 - Static Vs Instance Demo

Độ khó: ★★ Medium

Đề bài

Tạo class `Person` với:

- `static int population = 0` (class variable)
- `String name` (instance variable)
- `int age` (instance variable)
- Constructor nhận name, age; tăng population
- `static int getPopulation()` return population
- `void displayInfo()` in name, age, population

Trong main():

1. Tạo 3 persons
2. Display info của cả 3
3. In ra population từ static method

Input

- 3 nhóm, mỗi nhóm 2 dòng: name, age

Output

- 3 nhóm: name, age, population (tại thời điểm tạo)
- Dòng cuối: Total population

Constraints

- `0 ≤ age ≤ 150`

Ví dụ

Test case 1

Input:

```
Alice
25
Bob
30
Charlie
35
```

Output:

```
Alice 25 1
Bob 30 2
Charlie 35 3
Total: 3
```

Tags: `static`, `instance`, `demo`, `comparison`

CodeForge - B14 - Utility Class Với Static Methods

Độ khó: ★★ Medium

Đề bài

Tạo utility class `MathHelper` với:

- `static int factorial(int n)` tính n!
- `static boolean isPrime(int n)` kiểm tra số nguyên tố
- `static int gcd(int a, int b)` tìm ước chung lớn nhất
- **Private constructor** để ngăn tạo object

Trong main():

1. Nhận N
2. Gọi factorial(N)
3. Gọi isPrime(N)
4. Nhận M, gọi gcd(N, M)

Input

- Dòng 1: N
- Dòng 2: M

Output

- Dòng 1: N!
- Dòng 2: "PRIME" hoặc "NOT PRIME"
- Dòng 3: GCD(N, M)

Constraints

- $0 \leq N \leq 12$ (để factorial không overflow)
- $1 \leq M \leq 1000$

Ví dụ

Test case 1

Input:

```
5
10
```

Output:

```
120
PRIME
5
```

Test case 2

Input:

```
6
9
```

Output:

```
720
NOT PRIME
3
```

Tags: `static`, `utility`, `helper`, `private-constructor`

CodeForge - B15 - Constants Với Static Final

Độ khó: ★★ Medium

Đề bài

Tạo class `Constants` với:

- `static final double PI = 3.14159`
- `static final int MAX_SIZE = 100`
- `static final String APP_NAME = "MyApp"`
- Method `static void displayConstants()` in ra tất cả constants

Trong main():

1. Gọi displayConstants()
2. Access các constants qua Constants.PI, etc.

Lưu ý: final variable không thể thay đổi giá trị

Input

- Không có input

Output

- 3 dòng: PI, MAX_SIZE, APP_NAME

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
3.14159
100
MyApp
```

Tags: `static`, `final`, `constants`, `immutable`

CodeForge - B16 - Static Import Demo

Độ khó: ★★ Medium

Đề bài

Tạo class `GeometryUtils` với:

- `static final double PI = 3.14159`
- `static double circleArea(double r)` return PI _ r _ r
- `static double sphereVolume(double r)` return (4.0/3.0) _ PI _ r^3

Trong main():

1. Sử dụng `import static GeometryUtils.*`
2. Gọi methods trực tiếp mà không cần class name
3. Nhận radius, in area và volume

Input

- Một số thực radius

Output

- Dòng 1: Circle area (2 chữ số)
- Dòng 2: Sphere volume (2 chữ số)

Constraints

- `0 < radius ≤ 100`

Ví dụ

Test case 1

Input:

```
5.0
```

Output:

```
78.54
523.60
```

Test case 2

Input:

```
3.0
```

Output:

```
28.27
113.10
```

Tags: `static`, `import`, `static-import`, `utility`

CodeForge - B17 - toString() Override Đơn Giản

Độ khó: ★★ Medium

Đề bài

Tạo class `Book` với:

- `String title`
- `String author`
- Constructor nhận title, author
- Override `toString()` return format: "Book[title=..., author=...]"

Trong main():

1. Nhận title và author
2. Tạo object
3. In ra object (tự động gọi toString)

◇ Input

- Dòng 1: Title
- Dòng 2: Author

◇ Output

- In ra theo format toString()

◇ Constraints

- Độ dài ≤ 100

Ví dụ

Test case 1

Input:

```
Java Programming
John Doe
```

Output:

```
Book[title=Java Programming, author=John Doe]
```

Test case 2

Input:

```
Clean Code
Robert Martin
```

Output:

```
Book[title=Clean Code, author=Robert Martin]
```

Tags: tostring, override, basic, format

CodeForge - B18 - toString() Với Multiple Fields

Độ khó: ★★ Medium

Đề bài

Tạo class `Student` với:

- `String name`
- `int age`
- `double gpa`
- Constructor đầy đủ
- Override `toString()` return format: "Student{name='...', age=..., gpa=...}"

Trong main():

1. Nhận N
2. Tạo N students
3. In ra tất cả students (mỗi student tự động gọi toString)

Input

- Dòng 1: N
- N nhóm 3 dòng: name, age, gpa

Output

- N dòng: toString của mỗi student

Constraints

- `1 ≤ N ≤ 20`
- `0 ≤ age ≤ 100`
- `0.0 ≤ gpa ≤ 4.0`

Ví dụ

Test case 1

Input:

```
2
Alice
20
3.75
Bob
22
3.50
```

Output:

```
Student{name='Alice', age=20, gpa=3.75}
Student{name='Bob', age=22, gpa=3.50}
```

Tags: `tostring`, `override`, `multiple-fields`

CodeForge - B19 - toString() Format Custom

Độ khó: ★★ Medium

Đề bài

Tạo class `Rectangle` với:

- `double width`
- `double height`
- Constructor nhận width, height
- Method `double getArea()` return width * height
- Override `toString()` return format:
  - "Rectangle: W=... H=... Area=..."

Trong main():

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

Input

- Dòng 1: Width
- Dòng 2: Height

Output

- toString format với area calculated

Constraints

- `0 < width, height ≤ 1000`

Ví dụ

Test case 1

Input:

```
5.0
3.0
```

Output:

```
Rectangle: W=5.00 H=3.00 Area=15.00
```

Test case 2

Input:

```
10.5
7.2
```

Output:

```
Rectangle: W=10.50 H=7.20 Area=75.60
```

Tags: tostring, override, custom-format, calculation

CodeForge - B20A - Static Block Initialization

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo class `Configuration` với:

- `static String appName`
- `static int version`
- `static String[] supportedLanguages`
- **Static block** khởi tạo:
  - appName = "MyApp"
  - version = 1
  - supportedLanguages = {"English", "Vietnamese", "Japanese"}
- Method `static void displayConfig()` in tất cả config

Trong main():

1. Gọi displayConfig() (static block đã chạy trước)
2. Modify một vài config values
3. Display lại

Input

- Dòng 1: New app name
- Dòng 2: New version

Output

- Config ban đầu (từ static block)
- Config sau khi modify

Constraints

- N/A

Ví dụ

Test case 1

Input:

```
SuperApp
2
```

Output:

```
Initial Config:
App: MyApp
Version: 1
Languages: English, Vietnamese, Japanese

Modified Config:
App: SuperApp
Version: 2
Languages: English, Vietnamese, Japanese
```

Tags: `static`, `block`, `initialization`, `advanced`

CodeForge - B21A - Singleton Pattern Basic

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo class `Database` với Singleton pattern:

- `private static Database instance = null`
- `private String connectionString`
- **Private constructor** nhận connectionString
- `static Database getInstance(String connStr)`:
  - Nếu instance == null, tạo mới
  - Nếu không, return instance hiện tại
- Method `void connect()` in "Connected to: " + connectionString

Trong main():

1. Lần 1: gọi getInstance với connection string
2. Lần 2: gọi getInstance với connection string KHÁC (vẫn return instance cũ)
3. Connect từ cả 2 references
4. Check xem cả 2 có cùng object không

◇ Input

- Dòng 1: Connection string 1
- Dòng 2: Connection string 2

◇ Output

- Connection từ instance 1
- Connection từ instance 2 (cùng string với instance 1)
- "SAME INSTANCE"

◇ Constraints

- N/A

Ví dụ

Test case 1

Input:

```
localhost:3306
remotehost:5432
```

Output:

```
Connected to: localhost:3306
Connected to: localhost:3306
SAME INSTANCE
```

Giải thích: Instance 2 vẫn dùng connection cũ

Tags: `static`, `singleton`, `pattern`, `design-pattern`, `advanced`

CodeForge - B22A - Factory Pattern Với Static Methods

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo class `Shape` với:

- `String type`
- `double dimension1, dimension2`
- **Private constructor** nhận tất cả
- `static Shape createCircle(double radius)` return new Shape("Circle", radius, 0)
- `static Shape createRectangle(double w, double h)` return new Shape("Rectangle", w, h)
- `double getArea()`:
  - Circle: π * r^2
  - Rectangle: w * h
- Override `toString()` hiển thị type và area

Trong main():

1. Nhận N operations
2. Mỗi operation: "C radius" hoặc "R width height"
3. Tạo shapes bằng factory methods
4. In ra tất cả shapes

Input

- Dòng 1: N
- N dòng: operation

Output

- N dòng: toString của mỗi shape

Constraints

- `1 ≤ N ≤ 20`
- `0 < dimensions ≤ 100`

Ví dụ

Test case 1

Input:

```
3
C 5.0
```

```
R 4.0 6.0
C 3.0
```

Output:

```
Circle: Area=78.54
Rectangle: Area=24.00
Circle: Area=28.27
```

Tags: `static`, `factory`, `pattern`, `creation`, `advanced`

CodeForge - B23A - Counter Class Với Static Tracking

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo class `User` với:

- `static int totalUsers = 0`
- `static int activeUsers = 0`
- `int userId`
- `String username`
- `boolean isActive`
- Constructor nhận username:
  - Tăng totalUsers
  - Gán userId = totalUsers
  - Tăng activeUsers
  - Set isActive = true
- Method `void deactivate()`:
  - Set isActive = false
  - Giảm activeUsers
- `static void displayStats()` in totalUsers và activeUsers

Trong main():

1. Tạo N users
2. Deactivate một số users
3. Display stats

◇ Input

- Dòng 1: N (tổng users)
- N dòng: usernames
- Dòng N+2: M (số users deactivate)
- M dòng: userId cần deactivate

◇ Output

- Stats cuối cùng

◇ Constraints

- `1 ≤ N ≤ 100`
- `0 ≤ M ≤ N`

Ví dụ

Test case 1

Input:

```
5
Alice
Bob
Charlie
David
Eve
2
2
4
```

Output:

```
Total Users: 5
Active Users: 3
```

Giải thích: Bob và David deactivated

Tags: `static`, `counter`, `tracking`, `state-management`, `advanced`

CodeForge - B24A - Configuration Class Với Static Properties

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo class `AppConfig` với:

- `static String environment = "DEVELOPMENT"`
- `static boolean debugMode = true`
- `static int maxConnections = 10`
- Static block in ra "Config loaded"
- `static void setProduction()`:
  - environment = "PRODUCTION"
  - debugMode = false
  - maxConnections = 100
- `static void displayConfig()` in tất cả settings
- **Private constructor** ngăn instantiation

Trong main():

1. Display config ban đầu
2. Nhận command: "PROD" hoặc "DEV"
3. Nếu PROD: gọi setProduction()
4. Display config sau khi thay đổi

◇ Input

- Một dòng: "PROD" hoặc "DEV"

◇ Output

- Initial config
- Final config

◇ Constraints

- N/A

Ví dụ

Test case 1

Input:

```
PROD
```

Output:

```
Config loaded
Initial: DEVELOPMENT, Debug=true, MaxConn=10
Final: PRODUCTION, Debug=false, MaxConn=100
```

Test case 2

Input:

```
DEV
```

Output:

```
Config loaded
Initial: DEVELOPMENT, Debug=true, MaxConn=10
Final: DEVELOPMENT, Debug=true, MaxConn=10
```

Tags: `static`, `configuration`, `environment`, `settings`, `advanced`

CodeForge - B25A - Complex toString() Với Nested Formatting

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo class `Order` với:

- `static int orderCounter = 0`
- `int orderId`
- `String customerName`
- `String[] items` (tối đa 10 items)
- `double[] prices` (giá của từng item)
- `int itemCount`
- Constructor nhận customerName, tăng orderCounter, gán orderId
- Method `void addItem(String item, double price)` thêm item
- Method `double getTotal()` tính tổng giá
- Override `toString()` format:

```
Order #[orderId] - Customer: [name]
Items:
- [item1]: $[price1]
- [item2]: $[price2]
Total: $[total]
```

Trong main():

1. Nhận customer name
2. Nhận N items với prices
3. Tạo order, add items
4. In ra order (toString)

◇ Input

- Dòng 1: Customer name
- Dòng 2: N (số items)
- N nhóm 2 dòng: item name, price

◇ Output

- toString của order

◇ Constraints

- $1 \leq N \leq 10$
- $0 < price \leq 1000$

📊 Ví dụ

Test case 1

Input:

```
Alice
3
Laptop
999.99
Mouse
25.50
Keyboard
75.00
```

Output:

```
Order #1 - Customer: Alice
Items:
- Laptop: $999.99
- Mouse: $25.50
- Keyboard: $75.00
Total: $1100.49
```

Tags: tostring, override, complex, formatting, nested, advanced