CodeForge - B01 - Polymorphism Compile-time (Overloading)

Độ khó: ★ Easy

Đề bài

Tạo class `Calculator` với method overloading (compile-time polymorphism):

- `int add(int a, int b)` return a + b
- `double add(double a, double b)` return a + b
- `int add(int a, int b, int c)` return a + b + c

Trong main():

1. Tạo Calculator
2. Gọi cả 3 versions của add()
3. Compiler quyết định method nào được gọi (compile-time)

◇ Input

- Dòng 1: 2 số nguyên
- Dòng 2: 2 số thực
- Dòng 3: 3 số nguyên

◇ Output

- 3 dòng: kết quả của 3 method calls

◇ Constraints

- `-1000 ≤ các số ≤ 1000`

Ví dụ

Test case 1

Input:

```
5 3
2.5 3.7
1 2 3
```

Output:

```
8
6.20
6
```

Test case 2

Input:

```
10 20
5.5 4.5
7 8 9
```

Output:

```
30
10.00
24
```

Tags: `polymorphism`, `compile-time`, `overloading`, `method-resolution`

CodeForge - B02 - Polymorphism Runtime (Overriding)

Độ khó: ★ Easy

Đề bài

Tạo hierarchy với runtime polymorphism:

- Class `Animal` với method `void sound()` in "Animal sound"
- Class `Dog` extends Animal, override `sound()` in "Woof"
- Class `Cat` extends Animal, override `sound()` in "Meow"

Trong main():

1. Tạo Animal reference
2. Gán Dog object: `Animal a = new Dog();`
3. Gọi `a.sound()` → runtime quyết định gọi Dog's version

◇ Input

- Một dòng: "D" (Dog) hoặc "C" (Cat)

◇ Output

- Sound của animal type tương ứng

◇ Constraints

- Input chỉ là D hoặc C

Ví dụ

Test case 1

Input:

```
D
```

Output:

```
Woof
```

Test case 2

Input:

```
C
```

Output:

```
Meow
```

Tags: `polymorphism`, `runtime`, `overriding`, `dynamic-dispatch`

CodeForge - B03 - Upcasting Cơ Bản

Độ khó: ★ Easy

Đề bài

Tạo hierarchy:

- Class `Vehicle` với `void start()` in "Vehicle starting"
- Class `Car` extends Vehicle, override `start()` in "Car starting"

Trong main():

1. **Upcasting**: `Vehicle v = new Car();` (implicit, safe)
2. Gọi `v.start()` → gọi Car's version (runtime polymorphism)

◇ Input

- Không có input

◇ Output

- "Car starting"

◇ Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Car starting
```

Tags: `polymorphism`, `upcasting`, `implicit`, `safe`

CodeForge - B04 - Parent Reference To Multiple Child Types

Độ khó: ★ Easy

Đề bài

Tạo hierarchy:

- Class `Shape` với `void draw()` in "Drawing shape"
- Class `Circle` extends Shape, override `draw()` in "Drawing circle"
- Class `Rectangle` extends Shape, override `draw()` in "Drawing rectangle"

Trong main():

1. Tạo 2 Shape references
2. Assign Circle và Rectangle objects
3. Gọi draw() cho cả 2

◇ Input

- Không có input

◇ Output

- Dòng 1: "Drawing circle"
- Dòng 2: "Drawing rectangle"

◇ Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Drawing circle
Drawing rectangle
```

Tags: `polymorphism`, `parent-reference`, `multiple-children`

CodeForge - B05 - Method Resolution At Runtime

Độ khó: ★★ Medium

Đề bài

Tạo hierarchy:

- Class `Employee` với:
  - `double calculateSalary()` return 50000.0
- Class `Manager` extends Employee với:
  - Override `calculateSalary()` return 80000.0
- Class `Developer` extends Employee với:
  - Override `calculateSalary()` return 70000.0

Trong main():

1. Tạo array: `Employee[] employees = new Employee[2]`
2. `employees[0] = new Manager()`
3. `employees[1] = new Developer()`
4. Loop và gọi calculateSalary() → runtime resolution

Input

- Không có input

Output

- 2 dòng: salaries

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
80000.00
70000.00
```

Tags: `polymorphism`, `runtime`, `method-resolution`, `array`

CodeForge - B06 - Virtual Method Invocation

Độ khó: ★★ Medium

Đề bài

Minh họa virtual method invocation:

- Class `A` với `void method()` in "A's method"
- Class `B` extends A, override `void method()` in "B's method"
- Class `C` extends B, override `void method()` in "C's method"

Trong main():

1. `A ref = new C();` (upcasting 2 levels)
2. `ref.method()` → gọi C's version (virtual invocation)

◇ Input

- Không có input

◇ Output

- "C's method"

◇ Constraints

- N/A

Ví dụ

Test case 1

Output:

```
C's method
```

Tags: `polymorphism`, `virtual`, `method-invocation`, `runtime`

CodeForge - B07 - Polymorphic Behavior Demo

Độ khó: ★★ Medium

Đề bài

Tạo hierarchy:

- Class `Instrument` với `void play()` in "Playing instrument"
- Class `Piano` extends Instrument, override `play()` in "Playing piano"
- Class `Guitar` extends Instrument, override `play()` in "Playing guitar"
- Class `Drum` extends Instrument, override `play()` in "Playing drum"

Trong main():

1. Tạo method `void performConcert(Instrument i)` gọi `i.play()`
2. Pass các objects khác nhau → polymorphic behavior

Input

- Không có input

Output

- 3 dòng từ 3 instruments

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Playing piano
Playing guitar
Playing drum
```

Tags: `polymorphism`, `behavior`, `method-parameter`

CodeForge - B08 - Compile-time Vs Runtime Polymorphism

Độ khó: ★★ Medium

Đề bài

Demo cả 2 types:

- Class `Demo` với:
  - `void show(int x)` in "Int: [x]" (overloading)
  - `void show(String s)` in "String: [s]" (overloading)
- Class `SubDemo` extends Demo với:
  - Override `void show(int x)` in "SubDemo Int: [x]" (overriding)

Trong main():

1. Overloading: compiler quyết định (compile-time)
2. Overriding: JVM quyết định (runtime)

◇ Input

- Dòng 1: Số nguyên
- Dòng 2: String

◇ Output

- 2 dòng từ method calls

◇ Constraints

- `-100 ≤ số ≤ 100`

Ví dụ

Test case 1

Input:

```
42
Hello
```

Output:

```
SubDemo Int: 42
String: Hello
```

Tags: polymorphism, compile-time, runtime, comparison

CodeForge - B09 - Downcasting Cơ Bản

Độ khó: ★★ Medium

Đề bài

Tạo hierarchy:

- Class `Animal` với `void eat()` in "Animal eating"
- Class `Dog` extends Animal với:
  - Override `eat()` in "Dog eating"
  - Own method `void bark()` in "Woof"

Trong main():

1. Upcasting: `Animal a = new Dog();`
2. `a.eat()` works (inherited)
3. `a.bark()` NOT works (not in Animal)
4. **Downcasting**: `Dog d = (Dog) a;`
5. `d.bark()` works now

Input

- Không có input

Output

- Dòng 1: "Dog eating"
- Dòng 2: "Woof"

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Dog eating
Woof
```

Tags: `polymorphism`, `downcasting`, `explicit`, `casting`

CodeForge - B10 - instanceof Operator

Độ khó: ★★ Medium

Đề bài

Tạo hierarchy:

- Class `Vehicle`
- Class `Car` extends Vehicle
- Class `Truck` extends Vehicle

Trong main():

1. Tạo Vehicle reference với Car object
2. Kiểm tra `instanceof` trước khi downcast
3. Safe downcasting

Input

- Một dòng: "C" (Car) hoặc "T" (Truck)

Output

- Dòng 1: "instanceof Vehicle: true"
- Dòng 2: "instanceof Car: true/false"
- Dòng 3: "instanceof Truck: true/false"

Constraints

- Input chỉ là C hoặc T

Ví dụ

Test case 1

Input:

```
C
```

Output:

```
instanceof Vehicle: true
instanceof Car: true
instanceof Truck: false
```

Test case 2

Input:

```
T
```

Output:

```
instanceof Vehicle: true
instanceof Car: false
instanceof Truck: true
```

Tags: polymorphism, instanceof, type-checking, safe-casting

CodeForge - B11 - ClassCastException Prevention

Độ khó: ★★ Medium

Đề bài

Tạo hierarchy:

- Class `Shape`
- Class `Circle` extends Shape với `double radius`
- Class `Rectangle` extends Shape với `double width, height`

Trong main():

1. Tạo Shape reference với random object
2. Dùng instanceof kiểm tra trước khi downcast
3. Nếu không check → ClassCastException

Input

- Một dòng: "C" (Circle) hoặc "R" (Rectangle)

Output

- Nếu Circle: "Circle detected"
- Nếu Rectangle: "Rectangle detected"

Constraints

- Input chỉ là C hoặc R

Ví dụ

Test case 1

Input:

```
C
```

Output:

```
Circle detected
```

Test case 2

Input:

```
R
```

Output:

```
Rectangle detected
```

Tags: `polymorphism`, `exception`, `prevention`, `instanceof`

CodeForge - B12 - Safe Downcasting Pattern

Độ khó: ★★ Medium

Đề bài

Tạo hierarchy:

- Class `Employee` với `String name`
- Class `Manager` extends Employee với `void manage()` in "Managing team"
- Class `Developer` extends Employee với `void code()` in "Writing code"

Trong main():

1. Tạo Employee array với mixed types
2. Loop, check instanceof, downcast an toàn
3. Gọi specific methods

Input

- Dòng 1: N (số employees)
- N dòng: Type ("M" hoặc "D")

Output

- N dòng: actions của mỗi employee

Constraints

- `1 ≤ N ≤ 10`

Ví dụ

Test case 1

Input:

```
3
M
D
M
```

Output:

```
Managing team
Writing code
```

```
Managing team
```

Tags: polymorphism, safe-downcasting, pattern, instanceof

CodeForge - B13 - Upcasting Implicit Conversion

Độ khó: ★★ Medium

Đề bài

Minh họa implicit upcasting:

- Class `Number` (tự tạo, không dùng Java's Number)
- Class `Integer` extends Number với `int value`
- Class `Double` extends Number với `double value`

Trong main():

1. Method `void printNumber(Number n)` accept bất kỳ subclass nào
2. Pass Integer, Double objects → automatic upcasting
3. Polymorphic parameter

Input

- Dòng 1: Số nguyên
- Dòng 2: Số thực

Output

- 2 dòng: "Number received"

Constraints

- `-1000 ≤ values ≤ 1000`

Ví dụ

Test case 1

Input:

```
42
3.14
```

Output:

```
Number received
Number received
```

Tags: `polymorphism`, `upcasting`, `implicit`, `automatic`

CodeForge - B14 - Downcasting Explicit Conversion

Độ khó: ★ ★ ★ Hard

Đề bài

Tạo hierarchy:

- Class `BankAccount` với `double balance`
- Class `SavingsAccount` extends BankAccount với `void addInterest()`
- Class `CheckingAccount` extends BankAccount với `void writeCheck()`

Trong main():

1. Tạo BankAccount[] với mixed types
2. Loop, check instanceof
3. Downcast và gọi specific methods

◇ Input

- Dòng 1: N
- N dòng: Type ("S" hoặc "C")

◇ Output

- N dòng: operations

◇ Constraints

- `1 ≤ N ≤ 10`

Ví dụ

Test case 1

Input:

```
3
S
C
S
```

Output:

```
Interest added
Check written
```

```
Interest added
```

Tags: polymorphism, downcasting, explicit, banking

CodeForge - B15 - instanceof Với Inheritance Chain

Độ khó: ★ ★ ★ Hard

Đề bài

Tạo hierarchy:

- Class `A`
- Class `B` extends A
- Class `C` extends B

Trong main():

1. Tạo object C
2. Check instanceof với tất cả types trong chain
3. Object C là instance của C, B, và A (tất cả)

Input

- Không có input

Output

- 3 dòng: instanceof results

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
instanceof A: true
instanceof B: true
instanceof C: true
```

Tags: `polymorphism`, `instanceof`, `inheritance-chain`, `hierarchy`

CodeForge - B16 - Pattern Matching Với instanceof (Java 16+)

Độ khó: ★★ Medium

Đề bài

Tạo hierarchy:

- Class `Shape`
- Class `Circle` extends Shape với `double radius`
- Class `Rectangle` extends Shape với `double width, height`

Sử dụng pattern matching (nếu Java 16+):

```
if (shape instanceof Circle c) {
    // c tự động cast
}
```

Trong main():

1. Tạo Shape reference
2. Dùng pattern matching để access fields

Input

- Dòng 1: Type ("C" hoặc "R")
- Dòng 2+: Dimensions

Output

- Shape info

Constraints

- `0 < dimensions ≤ 100`

Ví dụ

Test case 1

Input:

```
C
5.0
```

Output:

```
Circle with radius: 5.00
```

Test case 2

Input:

```
R
4.0
6.0
```

Output:

```
Rectangle: 4.00 x 6.00
```

Tags: polymorphism, pattern-matching, java16, instanceof

CodeForge - B17 - Dynamic Method Dispatch Demo

Độ khó: ★★ Medium

Đề bài

Minh họa dynamic dispatch:

- Class `Base` với `void display()` in "Base display"
- Class `Derived` extends Base, override `display()` in "Derived display"

Trong main():

1. `Base b = new Derived();`
2. `b.display()` → JVM quyết định lúc runtime gọi Derived's version
3. Không phải compiler quyết định

◇ Input

- Không có input

◇ Output

- "Derived display"

◇ Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Derived display
```

Tags: `polymorphism`, `dynamic-dispatch`, `runtime`, `virtual`

CodeForge - B18 - Virtual Method Table Concept

Độ khó: ★ ★ ★ Hard

Đề bài

Minh họa virtual method table:

- Class `Animal` với `void eat()`, `void sleep()`
- Class `Dog` extends Animal, override `eat()` only
- Class `Cat` extends Animal, override cả 2

Trong main():

1. Tạo Animal[] với mixed types
2. Call methods → JVM lookup trong vtable
3. Overridden methods gọi child version, non-overridden gọi parent

Input

- Không có input

Output

- Method calls từ Dog và Cat

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Dog eating
Animal sleeping
Cat eating
Cat sleeping
```

Tags: `polymorphism`, `vtable`, `virtual-method`, `dispatch`

CodeForge - B19 - Method Overriding Với Different Return Types

Độ khó: ★★ Medium

Đề bài

Tạo hierarchy với covariant return types:

- Class `Animal` với `Animal reproduce()` return new Animal()
- Class `Dog` extends Animal, override `Dog reproduce()` return new Dog() (covariant)

Lưu ý: Return type có thể là subtype (Java 5+)

Trong main():

1. Animal reference với Dog object
2. Call reproduce()
3. Result là Dog object

Input

- Không có input

Output

- "Dog reproduced"

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Dog reproduced
```

Tags: `polymorphism`, `covariant`, `return-type`, `overriding`

CodeForge - B20 - Polymorphic Method Parameters

Độ khó: ★★ Medium

Đề bài

Tạo hierarchy:

- Class `Payment` với `void process()` in "Processing payment"
- Class `CreditCard` extends Payment, override `process()` in "Credit card payment"
- Class `PayPal` extends Payment, override `process()` in "PayPal payment"
- Class `Bitcoin` extends Payment, override `process()` in "Bitcoin payment"

Method `void checkout(Payment p)` accepts bất kỳ payment type nào.

Input

- Dòng 1: N
- N dòng: Payment type ("C", "P", hoặc "B")

Output

- N dòng: payment processing messages

Constraints

- `1 ≤ N ≤ 10`

Ví dụ

Test case 1

Input:

```
3
C
P
B
```

Output:

```
Credit card payment
PayPal payment
Bitcoin payment
```

Tags: `polymorphism`, `parameters`, `payment`, `real-world`

CodeForge - B21 - Polymorphic Return Types

Độ khó: ★★ Medium

Đề bài

Tạo factory pattern với polymorphism:

- Class `Vehicle`
- Class `Car` extends Vehicle
- Class `Bike` extends Vehicle

Method `Vehicle createVehicle(String type)` return different subtypes.

Input

- Dòng 1: N
- N dòng: Vehicle type ("C" hoặc "B")

Output

- N dòng: vehicle created messages

Constraints

- `1 ≤ N ≤ 10`

Ví dụ

Test case 1

Input:

```
3
C
B
C
```

Output:

```
Car created
Bike created
Car created
```

Tags: `polymorphism`, `return-type`, `factory`, `pattern`

CodeForge - B22 - Runtime Type Information

Độ khó: ★ ★ ★ Hard

Đề bài

Tạo hierarchy:

- Class `Shape` với `String getName()` return "Shape"
- Class `Circle` extends Shape, override `getName()` return "Circle"
- Class `Rectangle` extends Shape, override `getName()` return "Rectangle"

Trong main():

1. Tạo Shape[] với mixed types
2. Call getName() → runtime type information
3. Use getClass().getSimpleName() để confirm

◇ Input

- Dòng 1: N
- N dòng: Shape type ("C" hoặc "R")

◇ Output

- N nhóm 2 dòng: getName() và getClass() results

◇ Constraints

- `1 ≤ N ≤ 10`

Ví dụ

Test case 1

Input:

```
2
C
R
```

Output:

```
Circle
Circle
```

```
Rectangle
Rectangle
```

Tags: `polymorphism`, `rtti`, `runtime`, `type-information`

CodeForge - B23 - Polymorphic Array Cơ Bản

Độ khó: ★★ Medium

Đề bài

Tạo hierarchy:

- Class `Animal` với `void makeSound()`
- Class `Dog`, `Cat`, `Cow` extends Animal với override

Trong main():

1. Tạo `Animal[] animals = new Animal[3]`
2. Assign different subtypes
3. Loop và call makeSound() → polymorphic behavior

Input

- Không có input

Output

- 3 dòng: sounds

Constraints

- N/A

Ví dụ

Test case 1

Output:

```
Woof
Meow
Moo
```

Tags: `polymorphism`, `array`, `collection`, `heterogeneous`

CodeForge - B24 - Processing Heterogeneous Collections

Độ khó: ★ ★ ★ Hard

Đề bài

Tạo hierarchy:

- Class `Employee` với `double getSalary()`
- Class `Manager`, `Developer`, `Intern` extends Employee với different salaries

Trong main():

1. Tạo Employee[] với mixed types
2. Calculate total payroll (tổng lương)
3. Polymorphic collection processing

◇ Input

- Dòng 1: N
- N dòng: Employee type ("M", "D", hoặc "I")

◇ Output

- Total payroll

◇ Constraints

- `1 ≤ N ≤ 100`

Ví dụ

Test case 1

Input:

```
5
M
D
D
I
M
```

Output:

```
340000.00
```

Giải thích: M=80000, D=70000, I=30000 Total = 80000 + 70000 + 70000 + 30000 + 80000 = 330000

Tags: polymorphism, collection, processing, payroll

CodeForge - B25 - ArrayList With Polymorphism

Độ khó: ★ ★ ★ Hard

Đề bài

Tạo hierarchy:

- Class `Product` với `String name`, `double price`
- Class `Electronics`, `Clothing`, `Food` extends Product với specific features

Trong main():

1. Tạo `ArrayList<Product>`
2. Add different product types
3. Process polymorphically

Input

- Dòng 1: N
- N nhóm: Type và data (E/C/F)

Output

- Total price

Constraints

- `1 ≤ N ≤ 50`

Ví dụ

Test case 1

Input:

```
3
E Laptop 1000.00
C Shirt 50.00
F Rice 5.00
```

Output:

```
1055.00
```

Tags: `polymorphism`, `arraylist`, `collection`, `generics-preview`

CodeForge - B26 - Polymorphic Interface Preview

Độ khó: ★ ★ ★ Hard

Đề bài

Tạo simple hierarchy (interface học Buổi 16):

- Class `Drawable` (giả lập interface) với `void draw()`
- Classes `Circle`, `Rectangle`, `Triangle` implement behavior

Trong main():

1. Tạo Drawable[] với mixed types
2. Call draw() polymorphically

Input

- Dòng 1: N
- N dòng: Shape type ("C", "R", hoặc "T")

Output

- N dòng: drawing messages

Constraints

- `1 ≤ N ≤ 20`

Ví dụ

Test case 1

Input:

```
3
C
R
T
```

Output:

```
Drawing circle
Drawing rectangle
Drawing triangle
```

Tags: `polymorphism`, `interface-preview`, `drawable`, `shapes`

CodeForge - B27A - Zoo Management System

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo complete zoo system:

- Class `Animal` với:
  - `String name`, `int age`
  - `void eat()`, `void sleep()`, `void makeSound()`
- Classes `Lion`, `Elephant`, `Monkey` extends Animal với:
  - Override all methods với specific behaviors
  - Own methods (e.g., Lion.roar(), Elephant.spray(), Monkey.climb())

Trong main():

1. Tạo Animal[] zoo
2. Input N animals với types
3. Perform daily routine: eat → makeSound → sleep
4. Use instanceof để call specific methods

Input

- Dòng 1: N
- N dòng: Animal type ("L", "E", hoặc "M") và name

Output

- Daily routine cho mỗi animal
- Specific actions nếu downcast thành công

Constraints

- `1 ≤ N ≤ 20`

Ví dụ

Test case 1

Input:

```
3
L Simba
E Dumbo
M George
```

Output:

```
Simba is eating meat
Simba roars loudly
Simba is sleeping
Dumbo is eating plants
Dumbo sprays water
Dumbo is sleeping
George is eating fruits
George is climbing trees
George is sleeping
```

Tags: `polymorphism`, `zoo`, `system`, `real-world`, `advanced`

CodeForge - B28A - Payment Processing System

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo payment system:

- Class `Payment` với:
  - `double amount`
  - `boolean process()` (default return false)
  - `void printReceipt()`
- Classes `CreditCard`, `DebitCard`, `Cash`, `Cryptocurrency` extends Payment với:
  - Override `process()` với specific validation
  - Different processing fees
  - Own fields (cardNumber, walletAddress, etc.)

Trong main():

1. Input N transactions
2. Process each payment
3. Calculate total processed, total fees
4. Print receipts for successful transactions

Input

- Dòng 1: N
- N nhóm: Type, amount, và extra data

Output

- Processing status cho mỗi transaction
- Total processed
- Total fees

Constraints

- `1 ≤ N ≤ 50`
- `0 < amount ≤ 1000000`

Ví dụ

Test case 1

Input:

```
4
CC 1000.00 1234-5678-9012-3456
DC 500.00 9876-5432-1098-7654
CASH 200.00
CRYPTO 1500.00 0xABC123
```

Output:

```
Credit Card processed: $1000.00 (Fee: $30.00)
Debit Card processed: $500.00 (Fee: $10.00)
Cash processed: $200.00 (Fee: $0.00)
Crypto processed: $1500.00 (Fee: $45.00)
Total Processed: $3200.00
Total Fees: $85.00
```

Tags: polymorphism, payment, system, transaction, advanced

CodeForge - B29A - Vehicle Rental System

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo rental system:

- Class `Vehicle` với:
  - `String plateNumber`
  - `double dailyRate`
  - `boolean isAvailable`
  - `double calculateRentalCost(int days)`
- Classes `Car`, `Motorcycle`, `Truck`, `Van` extends Vehicle với:
  - Different pricing strategies
  - Extra costs (insurance, mileage, etc.)
  - Capacity restrictions

Trong main():

1. Initialize fleet (various vehicles)
2. Process N rental requests
3. Check availability, calculate cost
4. Update availability status

◇ Input

- Dòng 1: M (fleet size)
- M dòng: Vehicle data
- Dòng M+2: N (rental requests)
- N dòng: Vehicle type và days

◇ Output

- Rental confirmations hoặc rejections
- Total revenue

◇ Constraints

- `1 ≤ M ≤ 20`
- `1 ≤ N ≤ 50`

Ví dụ

Test case 1

Input:

```
3
CAR ABC123 50.00
MOTO XYZ789 30.00
TRUCK DEF456 100.00
3
CAR 3
MOTO 5
CAR 2
```

Output:

```
Car ABC123 rented for 3 days: $150.00
Motorcycle XYZ789 rented for 5 days: $150.00
Car ABC123 not available
Total Revenue: $300.00
```

Tags: `polymorphism`, `rental`, `system`, `business-logic`, `advanced`

CodeForge - B30A - Game Character Combat System

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo game combat system:

- Class `Character` với:
  - `String name`, `int health`, `int attackPower`, `int defense`
  - `void attack(Character target)` (virtual method)
  - `void takeDamage(int damage)`
  - `boolean isAlive()`
- Classes `Warrior`, `Mage`, `Archer`, `Healer` extends Character với:
  - Different attack mechanisms (physical, magical, ranged)
  - Special abilities (override attack)
  - Unique stats distribution

Trong main():

1. Create 2 teams (ArrayList)
2. Simulate turn-based combat
3. Use polymorphism để handle different character types
4. Display combat log và winner

◇ Input

- Dòng 1-2: Team sizes
- Team 1 characters (type, name, stats)
- Team 2 characters (type, name, stats)

◇ Output

- Combat log
- Winner announcement

◇ Constraints

- `1 ≤ team size ≤ 5`
- `0 < stats ≤ 1000`

Ví dụ

Test case 1

Input:

```
2
2
Warrior Knight 500 80 50
Mage Wizard 300 100 20
Archer Robin 400 70 30
Healer Priest 350 40 25
```

Output:

```
Knight attacks Robin for 65 damage
Wizard casts fireball on Priest for 95 damage
Robin shoots Knight for 55 damage
Priest heals Robin for 50 HP
...
Team 1 wins!
```

Tags: polymorphism, game, combat, rpg, advanced

CodeForge - B31A - E-commerce Product Catalog

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo product catalog system:

- Class `Product` với:
  - `String id`, `String name`, `double basePrice`
  - `double getFinalPrice()` (có thể override)
  - `String getDescription()`
- Classes `Electronics`, `Clothing`, `Books`, `Groceries` extends Product với:
  - Category-specific discounts
  - Tax calculations (different rates)
  - Shipping costs
  - Warranty, size, author, expiry date (specific fields)

Trong main():

1. Load catalog (ArrayList)
2. Process shopping cart
3. Calculate subtotal, taxes, shipping
4. Apply polymorphic pricing

◇ Input

- Dòng 1: N (catalog size)
- N dòng: Product data với type
- Dòng N+2: M (cart items)
- M dòng: Product IDs và quantities

◇ Output

- Cart details
- Subtotal, taxes, shipping, total

◇ Constraints

- `1 ≤ N ≤ 100`
- `1 ≤ M ≤ 20`

Ví dụ

Test case 1

Input:

```
4
E P001 Laptop 1000.00
C P002 Shirt 50.00
B P003 JavaBook 45.00
G P004 Rice 5.00
3
P001 1
P002 2
P003 1
```

Output:

```
Laptop x1: $1000.00
Shirt x2: $100.00
Java Book x1: $45.00
Subtotal: $1145.00
Tax: $91.60
Shipping: $15.00
Total: $1251.60
```

Tags: polymorphism, ecommerce, catalog, pricing, advanced

CodeForge - B32A - Media Library System

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo media library:

- Class `Media` với:
  - `String title`, `int year`, `double rating`
  - `void play()` (virtual)
  - `String getInfo()`
- Classes `Movie`, `TVShow`, `Podcast`, `Audiobook` extends Media với:
  - Specific metadata (duration, episodes, chapters)
  - Different play behaviors
  - Rating systems

Trong main():

1. Build library (ArrayList)
2. Search/filter operations
3. Create playlists
4. Play media polymorphically

◇ Input

- Dòng 1: N (library size)
- N dòng: Media data
- Dòng N+2: M (playlist items)
- M dòng: Media indices

◇ Output

- Library contents
- Playlist playback

◇ Constraints

- $1 \leq N \leq 50$
- $1 \leq M \leq 20$

Ví dụ

Test case 1

Input:

```
4
MOVIE Inception 2010 8.8 148
TV BreakingBad 2008 9.5 62
PODCAST TechTalk 2023 4.5 45
AUDIO Dune 1965 4.7 21
3
1
2
4
```

Output:

```
Playing: Inception (2010) - Duration: 148 min
Playing: Breaking Bad S01E01
Playing: Dune - Chapter 1
```

Tags: polymorphism, media, library, streaming, advanced

CodeForge - B33A - Banking System With Transactions

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo complete banking system:

- Class `Account` với:
  - `String accountNumber`, `double balance`
  - `boolean deposit(double amount)`
  - `boolean withdraw(double amount)`
  - `String getStatement()`
- Classes `SavingsAccount`, `CheckingAccount`, `BusinessAccount` extends Account với:
  - Different interest rates
  - Transaction fees
  - Overdraft limits
  - Minimum balance requirements
- Class `Transaction` để log activities

Trong main():

1. Create multiple accounts (polymorphic array)
2. Process transactions
3. Calculate interest, fees
4. Generate statements

◇ Input

- Dòng 1: N (accounts)
- N dòng: Account data
- Dòng N+2: M (transactions)
- M dòng: Account index, type (D/W), amount

◇ Output

- Transaction results
- Final balances
- Statements

◇ Constraints

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

Ví dụ

Test case 1

Input:

```
3
SAVINGS SA001 10000.00 0.05
CHECKING CA001 5000.00 1000.00
BUSINESS BA001 50000.00
5
0 D 5000.00
1 W 6000.00
2 D 10000.00
0 W 2000.00
1 D 1000.00
```

Output:

```
SA001: Deposit $5000.00 - Balance: $15000.00
CA001: Withdraw $6000.00 (Overdraft) - Balance: -$1000.00
BA001: Deposit $10000.00 - Balance: $60000.00
SA001: Withdraw $2000.00 - Balance: $13000.00
CA001: Deposit $1000.00 - Balance: $0.00
```

Tags: polymorphism, banking, transactions, complete-system, advanced

CodeForge - B34A - Notification System

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo notification system:

- Class `Notification` với:
  - `String message`, `String timestamp`, `String priority`
  - `void send()` (virtual)
  - `boolean isDelivered()`
- Classes `EmailNotification`, `SMSNotification`, `PushNotification`, `SlackNotification` extends Notification với:
  - Different delivery mechanisms
  - Retry logic
  - Formatting rules
  - Delivery confirmations

Trong main():

1. Create notification queue (ArrayList)
2. Add various notification types
3. Process queue polymorphically
4. Track delivery status
5. Implement retry for failed notifications

◇ Input

- Dòng 1: N (notifications)
- N dòng: Type, message, priority

◇ Output

- Delivery status cho mỗi notification
- Success/failure summary

◇ Constraints

- $1 \leq N \leq 50$

Ví dụ

Test case 1

Input:

```
5
EMAIL "Meeting at 3pm" HIGH
SMS "Code: 123456" URGENT
PUSH "New message" LOW
SLACK "Deploy completed" HIGH
EMAIL "Report ready" MEDIUM
```

Output:

```
[HIGH] Email sent to user@email.com: Meeting at 3pm
[URGENT] SMS sent to +1234567890: Code: 123456
[LOW] Push notification sent: New message
[HIGH] Slack message posted: Deploy completed
[MEDIUM] Email sent to user@email.com: Report ready
Successfully delivered: 5/5
```

Tags: polymorphism, notification, messaging, system, advanced

CodeForge - B35A - Complete Polymorphic System - Restaurant Management

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo complete restaurant system:

- Class `MenuItem` với:
  - `String name`, `double price`, `String category`
  - `double calculatePrice()` (can be overridden)
  - `String getDescription()`
- Classes `Appetizer`, `MainCourse`, `Dessert`, `Beverage` extends MenuItem với:
  - Specific pricing (tax, service charge)
  - Preparation time
  - Special dietary flags (vegan, gluten-free)
- Class `Order` với:
  - `ArrayList<MenuItem> items`
  - `double calculateTotal()` polymorphically
  - `void printReceipt()`
- Classes `DineIn`, `Takeout`, `Delivery` extends Order với:
  - Different fees and discounts
  - Tip handling
  - Delivery charges

Trong main():

1. Load menu (polymorphic collection)
2. Take N orders (mixed types)
3. Process orders
4. Calculate totals với polymorphic pricing
5. Generate receipts

◇ Input

- Dòng 1: M (menu items)
- M dòng: MenuItem data
- Dòng M+2: N (orders)
- N nhóm: Order type, số items, item indices

◇ Output

- Order confirmations
- Receipts
- Daily revenue

◇ Constraints

- $1 \leq M \leq 50$
- $1 \leq N \leq 20$

📊 Ví dụ

Test case 1

Input:

```
6
APP "Spring Rolls" 8.00
MAIN "Pad Thai" 15.00
MAIN "Green Curry" 16.00
DESSERT "Mango Sticky Rice" 7.00
BEV "Thai Iced Tea" 5.00
BEV "Coconut Water" 4.00
3
DINEIN 3 0 1 4
TAKEOUT 2 2 3
DELIVERY 4 0 1 3 4
```

Output:

```
Order #1 (Dine-In):
- Spring Rolls: $8.00
- Pad Thai: $15.00
- Thai Iced Tea: $5.00
Subtotal: $28.00
Tax: $2.52
Service: $2.80
Total: $33.32

Order #2 (Takeout):
- Green Curry: $16.00
- Mango Sticky Rice: $7.00
Subtotal: $23.00
Tax: $2.07
Discount: -$2.30
Total: $22.77

Order #3 (Delivery):
```

```
- Spring Rolls: $8.00
- Pad Thai: $15.00
- Mango Sticky Rice: $7.00
- Thai Iced Tea: $5.00
Subtotal: $35.00
Tax: $3.15
Delivery Fee: $5.00
Total: $43.15

Daily Revenue: $99.24
```

Tags: polymorphism, restaurant, complete-system, real-world, capstone, advanced