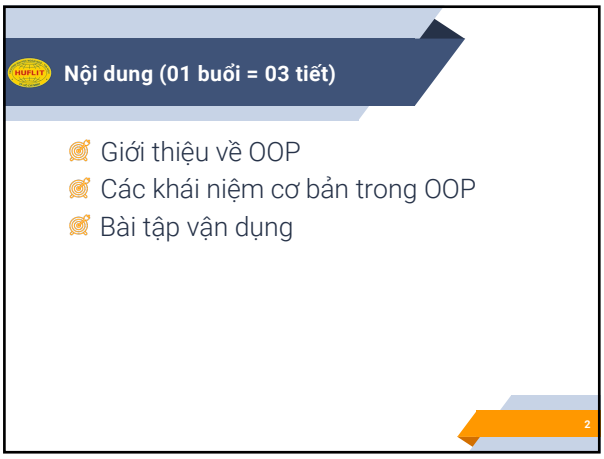

KHOA CÔNG NGHỆ THÔNG TIN
Bài 5
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (Tổng quan)
Biên soạn: Huỳnh Thành Lộc
Cập nhật: Tháng 01/2022

1


Nội dung (01 buổi = 03 tiết)
Giới thiệu về OOP
Các khái niệm cơ bản trong OOP
Bài tập vận dụng
2

2


1
GIỚI THIỆU VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
(Object-Oriented Programming)
3

3

Giới thiệu OOP

- Mô hình lập trình (*Programming Paradigm*): là **hướng tiếp cận** để giải quyết bài toán và cũng là **cách thức tổ chức code** của chương trình trên một ngôn ngữ lập trình cụ thể.
- Một số mô hình lập trình thông dụng:
  - Lập trình tuần tự (Sequential Programming)
  - Lập trình hướng thủ tục (Procedure-Oriented Programming – POP)
  - Lập trình hướng đối tượng (Object-Oriented Programming – OOP)
  - . . .

4

4

Giới thiệu OOP

Xét bài toán: Cho 2 số thực a và b.

Tính và in ra màn hình tổng, hiệu, tích, thương của chúng.

Có thể giải bài toán thông qua 3 bước:

- Nhập giá trị cho a và b
- Tính tổng, hiệu, tích, thương
- In kết quả ra màn hình

5

5

Giới thiệu OOP

Hướng tiếp cận 1: dùng các biến để lưu dữ liệu và viết một dãy các câu lệnh tuần tự để giải bài toán

```
static void Main(string[] args)
{
    double a = int.Parse(Console.ReadLine());
    double b = int.Parse(Console.ReadLine());

    double sum = a + b;
    double different = a - b;
    double product = a * b;
    double quotient = a / b;

    Console.WriteLine($"{a} + {b} = {sum}");
    Console.WriteLine($"{a} - {b} = {different}");
    Console.WriteLine($"{a} x {b} = {product}");
    Console.WriteLine($"{a} / {b} = {quotient}");
}
```

Sequential Programming

6

6

Giới thiệu OOP

- **Hướng tiếp cận 2**: dùng các biến để lưu dữ liệu và phân chia chương trình thành các hàm chức năng để giải bài toán
  - Nếu cần tính với nhiều lần với các giá trị a, b khác nhau, hướng tiếp cận này giúp hạn chế được viết lặp lại các đoạn code cùng thực hiện một chức năng

7

7

Giới thiệu OOP

```
static double Sum(double a, double b) { return a + b; }
static double Subtract(double a, double b) { return a - b; }
static double Multiply(double a, double b) { return a * b; }
static double Divide(double a, double b) { return a / b; }

static void PrintResult(double a, double b)
{
    Console.WriteLine($"{a} + {b} = {Sum(a,b)}");
    Console.WriteLine($"{a} - {b} = {Subtract(a,b)}");
    Console.WriteLine($"{a} x {b} = {Multiply(a,b)}");
    Console.WriteLine($"{a} / {b} = {Divide(a,b)}");
}

static void Main(string[] args)
{
    double a = int.Parse(Console.ReadLine());
    double b = int.Parse(Console.ReadLine());
    PrintResult(a, b);
}
```

Procedure-Oriented Programming

8

8

Giới thiệu OOP

- **Hướng tiếp cận 3**: thay vì tập trung vào các chức năng, chúng ta xác định các đối tượng xuất hiện trong bài toán và hành động mà các đối tượng đó cần thực hiện để cho ra kết quả bài toán

```
//Xây dựng lớp đối tượng Caculator!!!

static void Main(string[] args)
{
    double a = int.Parse(Console.ReadLine());
    double b = int.Parse(Console.ReadLine());

    Caculator myCaculator = new Caculator(a, b);
    myCaculator.Add();
    myCaculator.Subtract();
    myCaculator.Multiply();
    myCaculator.Divide();
}
```

Object-Oriented Programming

Tạo đối tượng tính toán tên *myCaculator*

Yêu cầu *myCaculator* thực hiện các chức năng tính toán

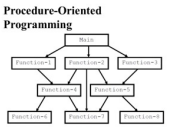
9

9

Giới thiệu OOP

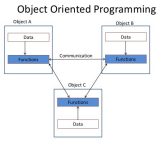
POP

- Tiếp cận giải quyết bài toán bằng cách xác định các chức năng cần thực hiện
- Các chức năng sẽ được định nghĩa thành các thủ tục/hàm



OOP

- Tiếp cận giải quyết bài toán bằng cách xác định các đối tượng có thể xuất hiện trong bài toán
- Mỗi đối tượng sẽ có dữ liệu và hành động của mình; và có thể tương tác với các đối tượng khác



10

10

Giới thiệu OOP

- Lập trình hướng đối tượng (*Object-Oriented Programming*): là mô hình lập trình mà bài toán được phân chia thành các đối tượng. Mỗi đối tượng sẽ *có dữ liệu riêng* và có khả năng *thực hiện các hành động* để tạo nên kết quả cho bài toán.

11

11

Giới thiệu OOP

Bài toán: Quản lý sinh viên

Viết một chương trình quản lý sinh viên với các tính năng:

- Lưu trữ thông tin một SV: *mã số SV, tên, năm sinh, giới tính, lớp*
- Nhập/xuất thông tin của một SV, tính tuổi của một SV, chuyển lớp cho một SV
- Quản lý một danh sách sinh viên: thêm một sinh viên vào danh sách, in toàn bộ SV trong danh sách, sắp xếp danh sách SV theo thứ tự tên tăng dần (theo alphabet)

12

12

Giới thiệu OOP

- Thay vì tập trung vào các chức năng, hãy phân tích:
  - Trong bài toán xuất hiện các đối tượng nào?
  - Cần lưu trữ thông tin gì trong mỗi đối tượng?
  - Những hành động mà đối tượng cần thực hiện?
  - Mối quan hệ giữa các đối tượng?

13

13

Giới thiệu OOP

- Bài toán có các đối tượng "sinh viên":

14

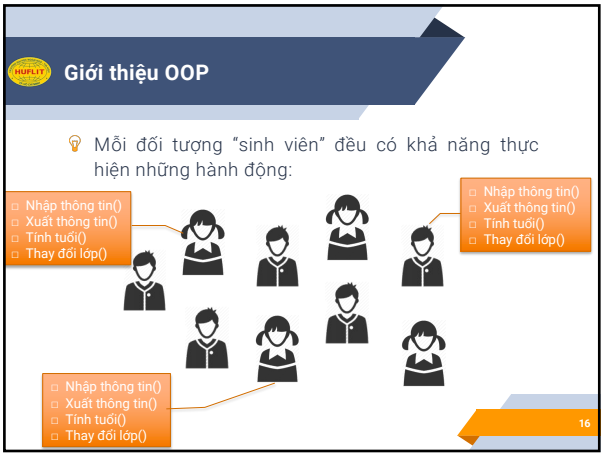
14

Giới thiệu OOP

- Trong mỗi đối tượng "sinh viên" cần lưu thông tin:

15

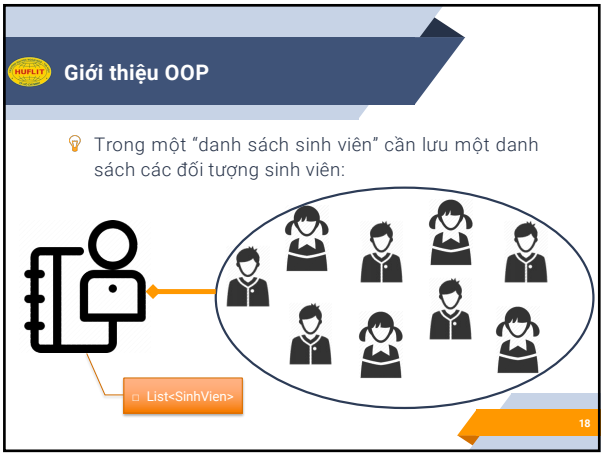
15


Giới thiệu OOP
Mỗi đối tượng "sinh viên" đều có khả năng thực hiện những hành động:
Nhập thông tin()
Xuất thông tin()
Tính tuổi()
Thay đổi lớp()
Nhập thông tin()
Xuất thông tin()
Tính tuổi()
Thay đổi lớp()
Nhập thông tin()
Xuất thông tin()
Tính tuổi()
Thay đổi lớp()
16

16


Giới thiệu OOP
Bài toán có đối tượng "danh sách sinh viên":
17

17


Giới thiệu OOP
Trong một "danh sách sinh viên" cần lưu một danh sách các đối tượng sinh viên:
List<SinhVien>
18

18

Giới thiệu OOP

- Đối tượng "danh sách sinh viên" có khả năng thực hiện những hành động:

19

19

Giới thiệu OOP

- Nhận xét:
  - Lập trình hướng thủ tục giải quyết trực tiếp bài toán bằng cách gọi các hàm
  - Lập trình hướng đối tượng giải quyết bài toán gián tiếp thông qua việc yêu cầu các đối tượng thực hiện các hành động

20

20

2

CÁC KHÁI NIỆM CƠ BẢN

21

21

Các khái niệm cơ bản

- Đối tượng (*Object*): là đơn vị cơ bản nhất trong OOP. Một đối tượng bao gồm:
  - Thuộc tính (Attribute/Field): những thông tin lưu trữ trong đối tượng
  - Phương thức (Method): những hành động mà đối tượng có thể thực hiện

22

22

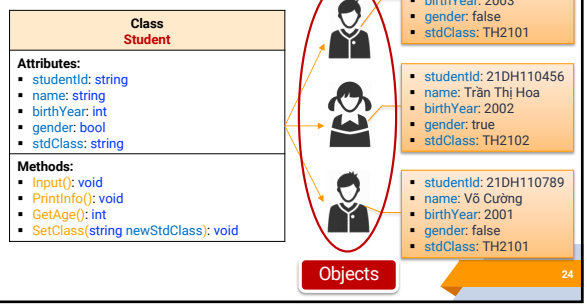
Các khái niệm cơ bản

- Lớp (Class): là kiểu dữ liệu do người dùng tự định nghĩa, mô tả bản thiết kế mẫu của các đối tượng cùng loại
  - Nội dung của lớp bao gồm các thuộc tính và phương thức
  - Từ định nghĩa lớp, chúng ta có thể tạo ra các đối tượng có cấu trúc chung nhưng thuộc tính của các đối tượng có thể khác nhau

23

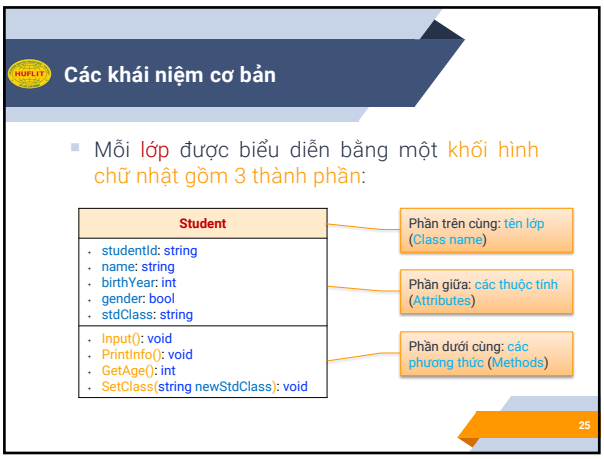
23

Các khái niệm cơ bản

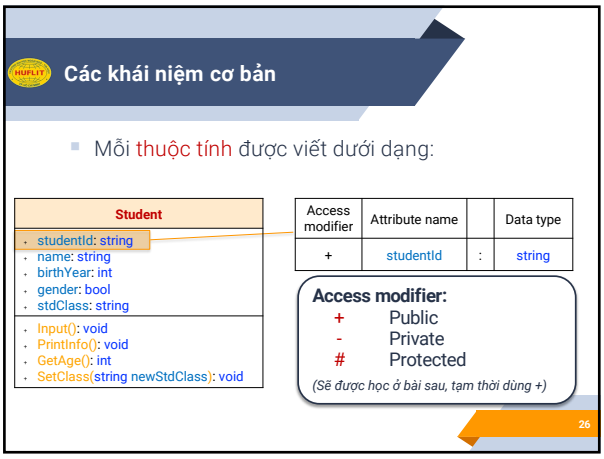


24

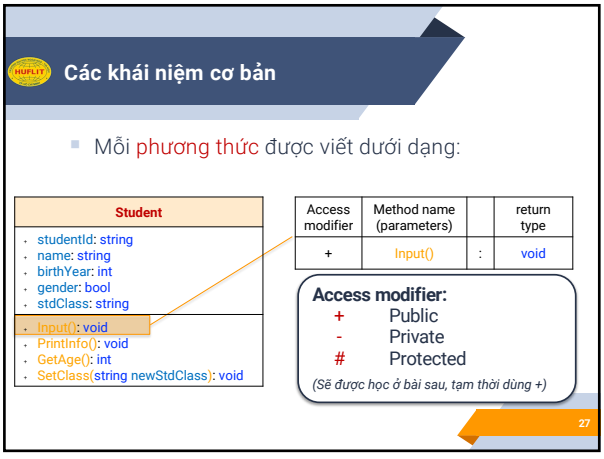
24


Các khái niệm cơ bản
Mỗi lớp được biểu diễn bằng một khối hình chữ nhật gồm 3 thành phần:
Student
studentId: string
name: string
birthYear: int
gender: bool
stdClass: string
Input(): void
PrintInfo(): void
GetAge(): int
SetClass(string newStdClass): void
Phần trên cùng: tên lớp (Class name)
Phần giữa: các thuộc tính (Attributes)
Phần dưới cùng: các phương thức (Methods)
25

25


Các khái niệm cơ bản
Mỗi thuộc tính được viết dưới dạng:
Student
studentId: string
name: string
birthYear: int
gender: bool
stdClass: string
Input(): void
PrintInfo(): void
GetAge(): int
SetClass(string newStdClass): void
Access modifier
Attribute name
Data type
+
studentId
:
string
Access modifier:
+ Public
- Private
Protected
(Sẽ được học ở bài sau, tạm thời dùng +)
26

26


Các khái niệm cơ bản
Mỗi phương thức được viết dưới dạng:
Student
studentId: string
name: string
birthYear: int
gender: bool
stdClass: string
Input(): void
PrintInfo(): void
GetAge(): int
SetClass(string newStdClass): void
Access modifier
Method name (parameters)
return type
+
Input()
:
void
Access modifier:
+ Public
- Private
Protected
(Sẽ được học ở bài sau, tạm thời dùng +)
27

27

3

BÀI TẬP VẬN DỤNG

28

28

Bài tập vận dụng

Bài tập 1: Class **Fraction**

Hãy xác định các thuộc tính và phương thức cho lớp Fraction mô tả các đối tượng là các phân số

29

29

Bài tập vận dụng

Bài tập 1: Class **Fraction**

Fraction
- numerator: int - denominator: int
- Input(): void - Simplify(): void - ToString(): string - Decimal(): double - Add(Fraction p): Fraction - Subtract(Fraction p): Fraction - Multiply(Fraction p): Fraction - Divide(Fraction p): Fraction

30

30

Bài tập vận dụng

Bài tập 2: Class **Rectangle**

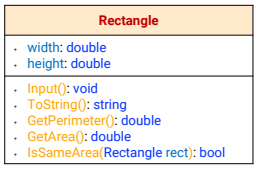
Hãy xác định các thuộc tính và phương thức cho lớp Rectangle mô tả các đối tượng là các hình chữ nhật

31

31

Bài tập vận dụng

Bài tập 2: Class **Rectangle**



32

32

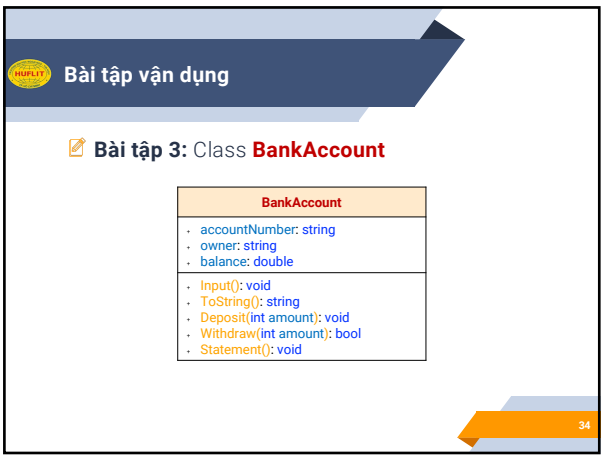
Bài tập vận dụng

Bài tập 3: Class **BankAccount**

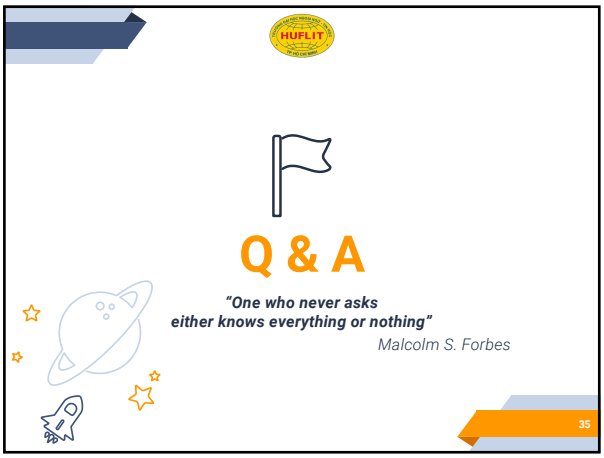
Hãy xác định các thuộc tính và phương thức cho lớp BankAccount mô tả các đối tượng là các tài khoản ngân hàng

33

33


Bài tập vận dụng
Bài tập 3: Class BankAccount
BankAccount
accountNumber: string
owner: string
balance: double
Input(): void
ToString(): string
Deposit(int amount): void
Withdraw(int amount): bool
Statement(): void
34

34


Q & A
"One who never asks
either knows everything or nothing"
Malcolm S. Forbes
35

35