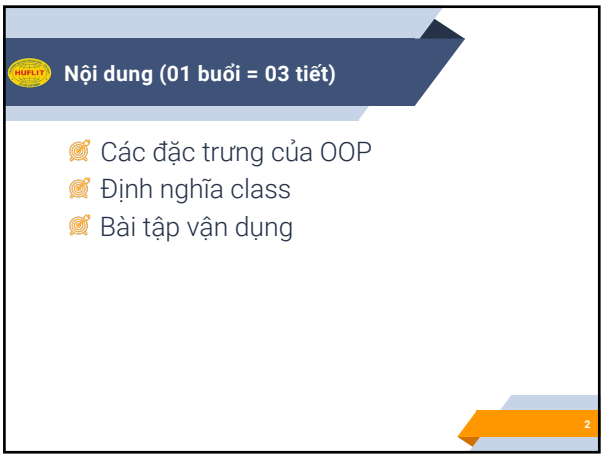

KHOA CÔNG NGHỆ THÔNG TIN
Bài 6
OOP -
Tính Trừu tượng và
Đóng gói
Biên soạn: Huỳnh Thành Lộc
Cập nhật: Tháng 01/2022

1

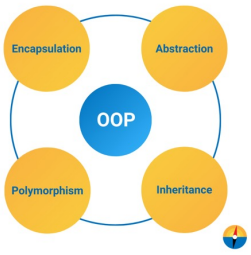

Nội dung (01 buổi = 03 tiết)
Các đặc trưng của OOP
Định nghĩa class
Bài tập vận dụng
2

2


1
CÁC ĐẶC TRƯNG
CỦA OOP
3

3

Các đặc trưng của OOP

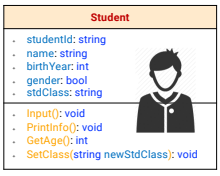


4

4

Các đặc trưng của OOP

- Tính Trừu tượng (*Abstraction*): giúp làm giảm đi sự phức tạp của chương trình bằng cách:
  - Ẩn đi các cài đặt chi tiết
  - Làm nổi bật lên các chức năng người dùng có thể thực hiện để thao tác với đối tượng



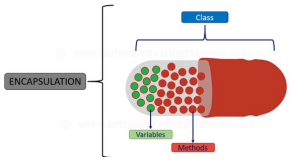
Ví dụ: Khi làm việc với một đối tượng của lớp Student chúng ta chỉ quan tâm có thể thực hiện được những phương thức nào mà không cần biết việc cài đặt chi tiết bên trong như thế nào

5

5

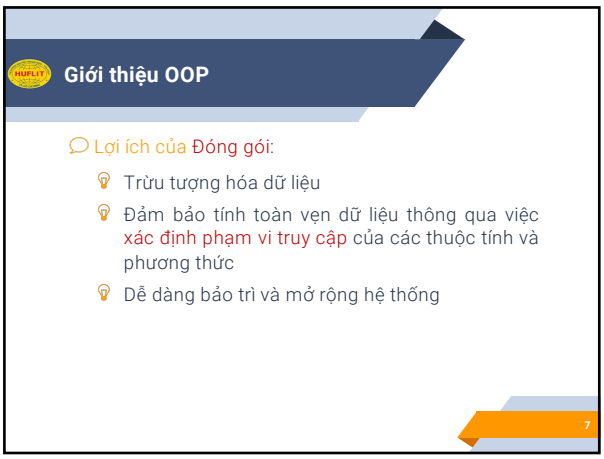
Các đặc trưng của OOP

- Tính Đóng gói (*Encapsulation*): các dữ liệu và phương thức có liên quan với nhau được đóng gói thành một lớp để tiện quản lý và sử dụng → việc tạo các class chính là đang thực hiện tính đóng gói

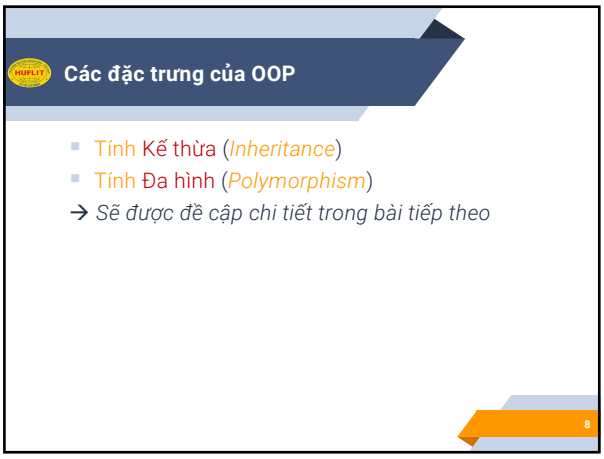


6

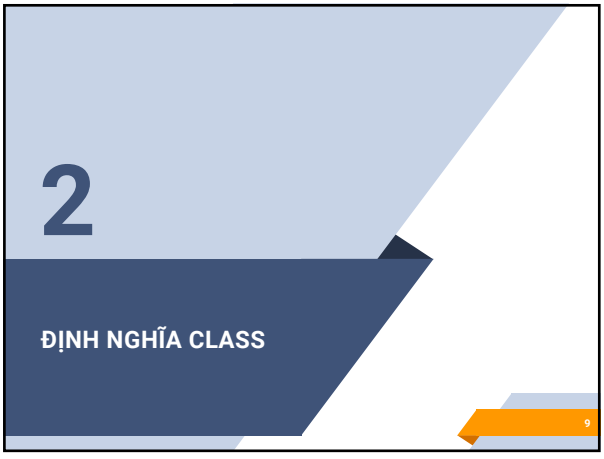
6


Giới thiệu OOP
Lợi ích của Đóng gói:
Trừu tượng hóa dữ liệu
Đảm bảo tính toàn vẹn dữ liệu thông qua việc xác định phạm vi truy cập của các thuộc tính và phương thức
Dễ dàng bảo trì và mở rộng hệ thống
7

7


Các đặc trưng của OOP
Tính Kế thừa (Inheritance)
Tính Đa hình (Polymorphism)
→ Sẽ được đề cập chi tiết trong bài tiếp theo
8

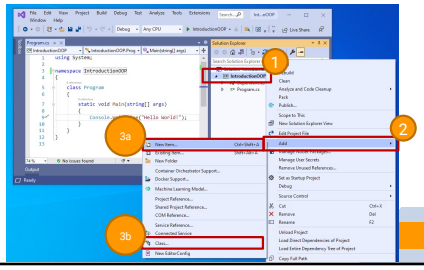
8


2
ĐỊNH NGHĨA CLASS
9

9

Tạo class mới trong C#

- Right-click vào **Project Name → Add → New Item… → Class**

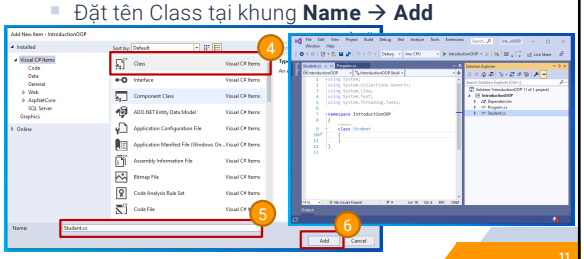


10

10

Tạo class mới trong C#

- Trong hộp thoại **Add New Item** chọn **Class**
- Đặt tên Class tại khung **Name → Add**



11

11

Cú pháp chung của một class

- Cú pháp chung của một class:

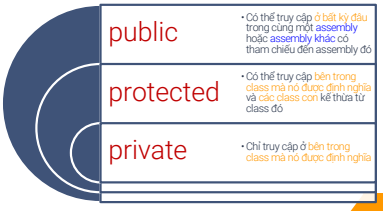
```
[access modifier] class <ClassName>[:BaseClassName]
{
    // Khai báo các thuộc tính (Attributes)
    [access modifier] <type> attributeName;

    // Các phương thức khởi tạo (Constructors)
    [access modifier] ClassName([Parameters])
    {...}

    // Các phương thức (Methods)
    [access modifier] <return-type> MethodName([Parameters])
    {...}
}
```

12

12

Access modifier

- Phạm vi truy cập (*access modifier*): dùng để xác định phạm vi cho phép truy cập các thành phần của một class



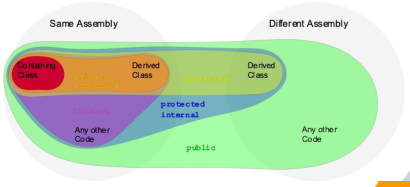
13

Ghi chú: *Kết quả biên dịch mọi project của C# Compiler đều được gọi chung là assembly*

13

Access modifier

- Ngoài ra, C# còn các access modifier như: internal, protected internal, private protected *(Available since C# 7.2)*



14

14

Access modifier

- Ngoài ra, C# còn các access modifier như: internal, protected internal, private protected *(Available since C# 7.2)*

Caller's location	public	protected internal	protected	internal	private protected	private
Within the class	✓	✓	✓	✓	✓	✓
Derived class (same assembly)	✓	✓	✓	✓	✓	✗
Non-derived class (same assembly)	✓	✓	✗	✓	✗	✗
Derived class (different assembly)	✓	✓	✓	✗	✗	✗
Non-derived class (different assembly)	✓	✗	✗	✗	✗	✗

15

15

Constructors

- Phương thức khởi tạo (*Constructors*): là phương thức đặc biệt dùng để khởi tạo nên các đối tượng của một lớp:
  - Constructors cho phép thiết lập giá trị ban đầu cho các attributes của đối tượng
  - Một class có thể có nhiều constructors với số lượng tham số khác nhau

16

16

Constructors

- Constructors có đặc điểm:
  - Không có kiểu trả về (return type)
  - Tên constructors giống với tên class
  - Nếu không định nghĩa constructor thì class có một constructors mặc định không có tham số (default constructor)
  - Khi định nghĩa bất kỳ constructor nào thì phải định nghĩa lại default constructor nếu muốn sử dụng (do trình biên dịch không tự tạo default constructor nữa)

17

17

Định nghĩa class

```
public class Student
{
    // Attributes
    public string studentId;
    public string name;
    public int birthYear;
    public bool gender;
    public string stdClass;

    // Constructors
    public Student() {}
    public Student(string stdId, string stdName)
    {
        studentId = stdId;
        name = stdName;
    }

    // Methods
    public void Input() {. . .}
    public void PrintInfo() {. . .}
    public int GetAge() {. . .}
    public void SetClass(string newClass) {. . .}
}
```

Constructor khởi tạo một SV không có giá trị ban đầu cho các attributes

Constructor khởi tạo một SV có giá trị cho `studentId` và `name`

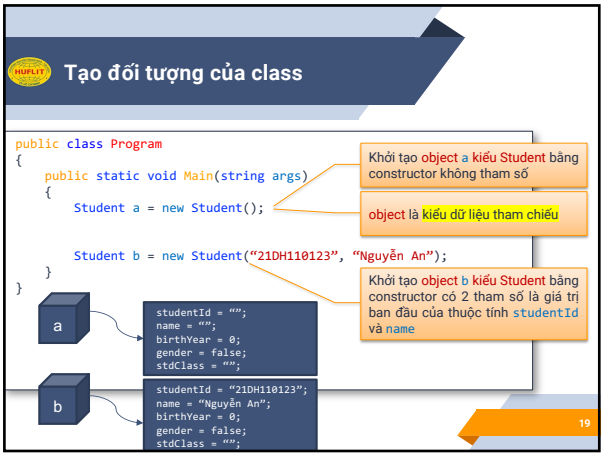
Student

- studentId: string
- name: string
- birthYear: int
- gender: bool
- stdClass: string

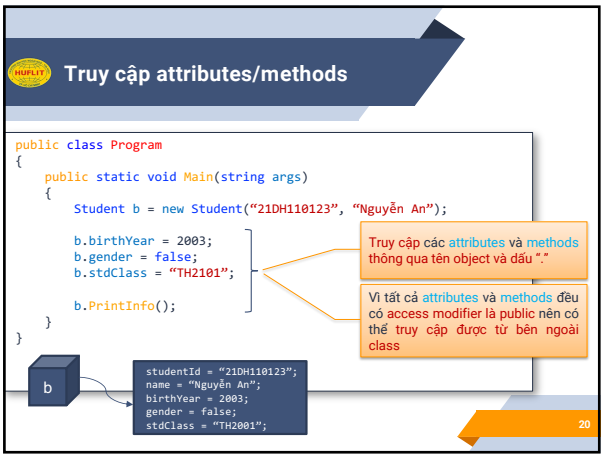
- Student()
- Student(string stdId, string stdName)
- Input(): void
- PrintInfo(): void
- GetAge(): int
- SetClass(string newStdClass): void

18

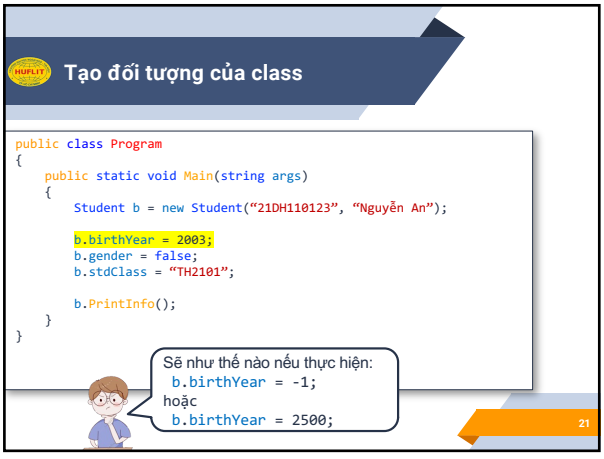
18


Tạo đối tượng của class
public class Program
{
public static void Main(string args)
{
Student a = new Student();
Student b = new Student("21DH110123", "Nguyễn An");
}
}
Khởi tạo object a kiểu Student bằng constructor không tham số
object là kiểu dữ liệu tham chiếu
Khởi tạo object b kiểu Student bằng constructor có 2 tham số là giá trị ban đầu của thuộc tính studentId và name
a
studentId = "";
name = "";
birthYear = 0;
gender = false;
stdClass = "";
b
studentId = "21DH110123";
name = "Nguyễn An";
birthYear = 0;
gender = false;
stdClass = "";
19

19


Truy cập attributes/methods
public class Program
{
public static void Main(string args)
{
Student b = new Student("21DH110123", "Nguyễn An");
b.birthYear = 2003;
b.gender = false;
b.stdClass = "TH2101";
b.PrintInfo();
}
}
Truy cập các attributes và methods thông qua tên object và dấu "."
Vì tất cả attributes và methods đều có access modifier là public nên có thể truy cập được từ bên ngoài class
b
studentId = "21DH110123";
name = "Nguyễn An";
birthYear = 2003;
gender = false;
stdClass = "TH2001";
20

20


Tạo đối tượng của class
public class Program
{
public static void Main(string args)
{
Student b = new Student("21DH110123", "Nguyễn An");
b.birthYear = 2003;
b.gender = false;
b.stdClass = "TH2101";
b.PrintInfo();
}
}
Sẽ như thế nào nếu thực hiện:
b.birthYear = -1;
hoặc
b.birthYear = 2500;
21

21

Phương thức Get/Set

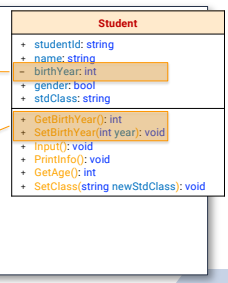
- Các thuộc tính có ràng buộc giá trị *(cần phải kiểm tra tính hợp lệ của dữ liệu trước khi gán cho thuộc tính)* thường có access modifier là private
- Việc lấy/thiết lập giá trị của thuộc tính được thực hiện gián tiếp thông qua phương thức get/set

22

22

Phương thức Get/Set

```
public class Student
{
    // Attributes
    private int birthYear;
    . . .

    // Methods
    public int GetBirthYear()
    {
        return birthYear;
    }
    public void SetBirthYear(int year)
    {
        birthyear = year;
    }
}
```

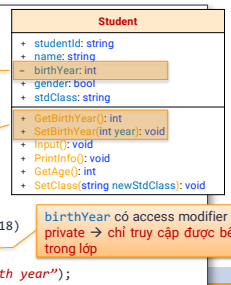


23

23

Phương thức Get/Set

```
public class Student
{
    // Attributes
    private int birthYear;
    . . .
    // Methods
    public int GetBirthYear()
    {
        return birthYear;
    }
    public void SetBirthYear(int year)
    {
        int curYear = DateTime.Now.Year;
        if (year > 0 && curYear - year >= 18)
            birthYear = year;
        else
            Console.WriteLine("Invalid birth year");
    }
}
```



birthYear có access modifier là private → chỉ truy cập được bên trong lớp

24

24

Phương thức Get/Set

```
public class Program
{
    public static void Main(string args)
    {
        Student a = new Student();
        a.studentId = "21DH110123";
        a.name = "Nguyễn An";
        a.gender = false;
        a.stdClass = "TH2101";

        a.birthYear = 2003;

        a.SetBirthYear(-1);
        a.SetBirthYear(2500);
        a.SetBirthYear(2003);

        Console.WriteLine(a.GetBirthYear());
    }
}
```

Không thể truy cập thuộc tính `birthYear` ở bên ngoài lớp

Dữ liệu có giá trị không hợp lệ sẽ không được ghi nhận vào `birthYear`

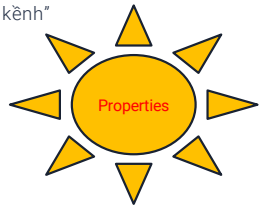
Thiết lập giá trị cho `birthYear` thành công

25

25

Phương thức Get/Set

- Nhận xét:
  - Nhu cầu sử dụng phương thức get/set rất nhiều
  - Nhiều lần gọi get/set làm cho code trở nên "cồng kềnh"



26

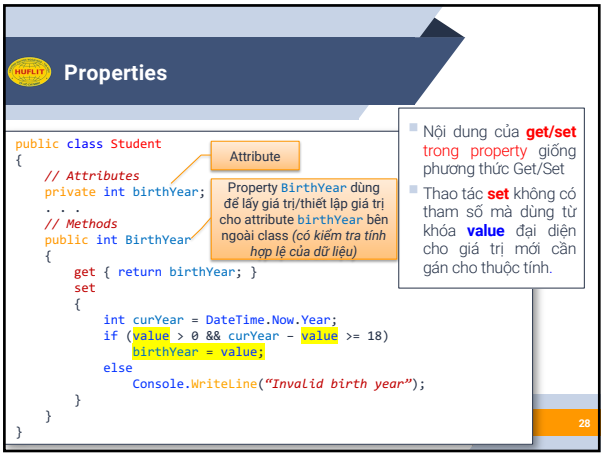
26

Properties

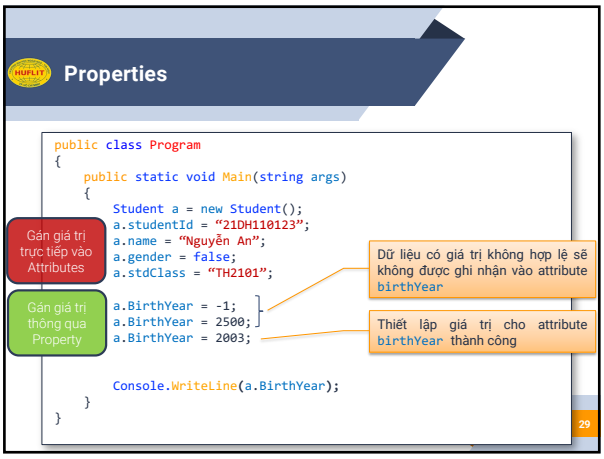
- Properties là sự kết hợp giữ attributes và methods:
  - Người dùng có thể truy cập properties với cú pháp giống như attributes
  - Thao tác lấy/thiết lập giá trị của properties có thể được xử lý như methods

27

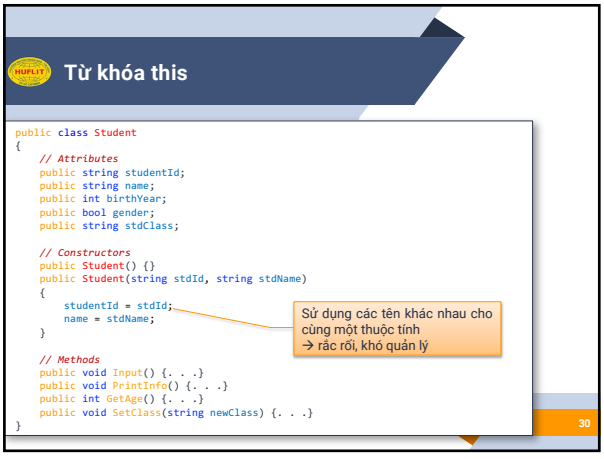
27


Properties
public class Student
{
// Attributes
private int birthYear;
. . .
// Methods
public int BirthYear
{
get { return birthYear; }
set
{
int curYear = DateTime.Now.Year;
if (value > 0 && curYear - value >= 18)
birthYear = value;
else
Console.WriteLine("Invalid birth year");
}
}
}
Attribute
Property BirthYear dùng để lấy giá trị/thiết lập giá trị cho attribute birthYear bên ngoài class (có kiểm tra tính hợp lệ của dữ liệu)
Nội dung của get/set trong property giống phương thức Get/Set
Thao tác set không có tham số mà dùng từ khóa value đại diện cho giá trị mới cần gán cho thuộc tính.
28

28


Properties
public class Program
{
public static void Main(string args)
{
Student a = new Student();
a.studentId = "21DH110123";
a.name = "Nguyễn An";
a.gender = false;
a.stdClass = "TH2101";
a.BirthYear = -1;
a.BirthYear = 2500;
a.BirthYear = 2003;
Console.WriteLine(a.BirthYear);
}
}
Gán giá trị trực tiếp vào Attributes
Gán giá trị thông qua Property
Dữ liệu có giá trị không hợp lệ sẽ không được ghi nhận vào attribute birthYear
Thiết lập giá trị cho attribute birthYear thành công
29

29


Từ khóa this
public class Student
{
// Attributes
public string studentId;
public string name;
public int birthYear;
public bool gender;
public string stdClass;
// Constructors
public Student() {}
public Student(string stdId, string stdName)
{
studentId = stdId;
name = stdName;
}
// Methods
public void Input() {. . .}
public void PrintInfo() {. . .}
public int GetAge() {. . .}
public void SetClass(string newClass) {. . .}
}
Sử dụng các tên khác nhau cho cùng một thuộc tính
→ rắc rối, khó quản lý
30

30

Từ khóa this

```
public class Student
{
    // Attributes
    public string studentId;
    public string name;
    public int birthYear;
    public bool gender;
    public string stdClass;

    // Constructors
    public Student() {}
    public Student(string studentId, string name)
    {
        studentId = studentId;
        name = name;
    }

    // Methods
    public void Input() {. . .}
    public void PrintInfo() {. . .}
    public int GetAge() {. . .}
    public void SetClass(string newClass) {. . .}
}
```

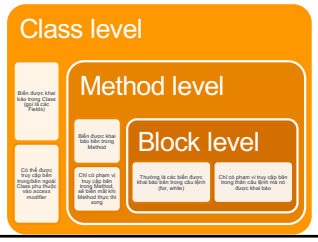
Viết như thế nào có được không?

31

31

Từ khóa this

- Phạm vi của biến (variable scope): cho biết phạm vi tồn tại của một biến trong chương trình, phụ thuộc vào vị trí khai báo biến



32

32

Từ khóa this

```
namespace Demo
{
    class Program
    {
        private int[] a = {1, 5, 8, 7, 2};

        public int FindX(int x)
        {
            for (int i = 0; i < a.Lenght; i++)
                if(a[i] == x)
                    return i;
            return -1;
        }
    }
}
```

Class level

Method level

Block level

33

33

Từ khóa this

```
public class Student
{
    // Attributes
    public string studentId;
    public string name;
    public int birthYear;
    public bool gender;
    public string stdClass;

    // Constructors
    public Student() {}
    public Student(string studentId, string name)
    {
        this.studentId = studentId;
        this.name = name;
    }

    // Methods
    public void Input() {. . .}
    public void PrintInfo() {. . .}
    public int GetAge() {. . .}
    public void SetClass(string newClass) {. . .}
}
```

Từ khóa this tham chiếu đến đối tượng hiện tại của lớp
→ this.studentId: xác định truy cập thuộc tính studentId của đối tượng

34

34

3

BÀI TẬP VẬN DỤNG

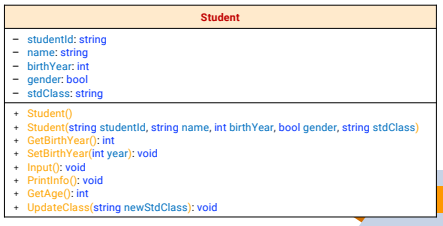
35

35

Bài tập vận dụng

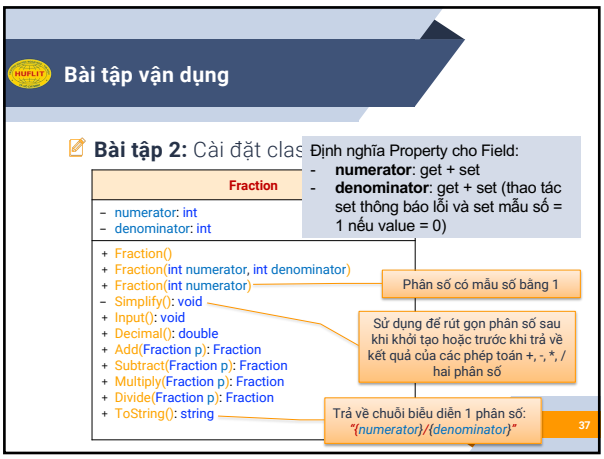
Bài tập 1: Cài đặt class **Student**

Cài đặt hoàn chỉnh class Student theo mô tả *(định nghĩa properties cho các attributes)*

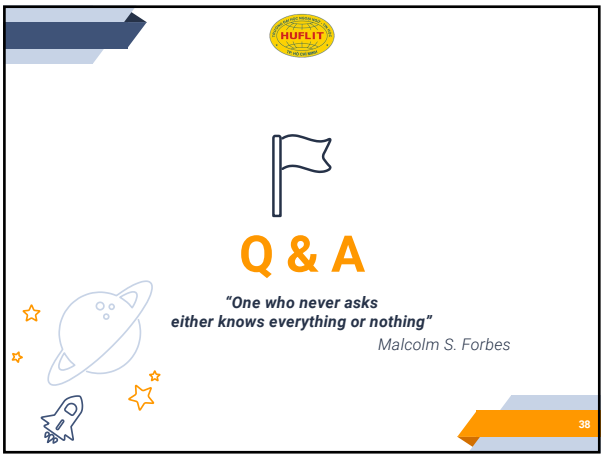


36

36


Bài tập vận dụng
Bài tập 2: Cài đặt clas
Fraction
- numerator: int
- denominator: int
+ Fraction()
+ Fraction(int numerator, int denominator)
+ Fraction(int numerator)
- Simplify(): void
+ Input(): void
+ Decimal(): double
+ Add(Fraction p): Fraction
+ Subtract(Fraction p): Fraction
+ Multiply(Fraction p): Fraction
+ Divide(Fraction p): Fraction
+ ToString(): string
Định nghĩa Property cho Field:
- numerator: get + set
- denominator: get + set (thao tác set thông báo lỗi và set mẫu số = 1 nếu value = 0)
Phân số có mẫu số bằng 1
Sử dụng để rút gọn phân số sau khi khởi tạo hoặc trước khi trả về kết quả của các phép toán +, -, *, / hai phân số
Trả về chuỗi biểu diễn 1 phân số: "{numerator}/{denominator}"
37

37


Q & A
"One who never asks
either knows everything or nothing"
Malcolm S. Forbes
38

38