Class and Object

Môn học: Kỹ thuật lập trình [Tuần 3]

Giảng viên: ThS. Trần Đức Hiếu

Giới thiệu

- Lịch sử phát triển của kĩ thuật lập trình
- Phương pháp lập trình hướng đối tượng (OOP)
- Khái niệm Class và Object

Computer Programming

- **Thuật toán** (*Algorithm*) là một qui trình tuần tự các bước để giải quyết một vấn đề

- **Chương trình máy tính** (*Computer Program*) là một tập các chỉ dẫn cho máy tính để thực hiện một công việc nào đó

➔Mọi chương trình máy tính đều có thể được xem là một thuật toán

Programming Languages

- Ngôn ngữ lập trình là loại ngôn ngữ cho phép những người lập trình có thể tạo ra các phần mềm để giao tiếp và điều khiển máy tính
- Có 3 dạng ngôn ngữ lập trình:
  - Ngôn ngữ máy (Machine Languages)
  - Hợp ngữ (Assembly Languages)
  - Ngôn ngữ cấp cao (High-Level Languages)

Ngôn ngữ máy

- Chỉ bao gồm các con số 0 và 1
- Đây là "ngôn ngữ tự nhiên" của máy tính
- Rất khó lập trình vì chỉ cần sai sót một con số 0 hay 1 sẽ làm chương trình bị lỗi
- Ví dụ

1110001001010101

0000101001001000010

0101011000010010

1010101011101010100

Hợp ngữ

- Là loại ngôn ngữ được phát triển trên nền ngôn ngữ máy giúp cho việc lập trình trở nên dễ dàng hơn
- Hợp ngữ bao gồm một tập các lệnh cơ bản gắn với một bộ vi xử lý nhất định
- Mã hợp ngữ sẽ được chuyển đổi thành mã máy trước máy tính thi hành
- Ví dụ

Add 1001010, 1011010

Ngôn ngữ cấp cao

- Là một bước tiến rất lớn, cho phép tạo ra các sản phẩm phần mềm có độ phức tạp và tính ứng dụng cao
- Cú pháp tựa ngôn ngữ tự nhiên của con người
- Ví dụ **hocPhi = soTinChi * donGiaTinChi**
- Có thể được chia thành 2 nhóm:
  - NNLT hướng cấu trúc (Structured Programming)
  - NNLT hướng đối tượng (Object-Oriented Programming)

Ngôn ngữ hướng cấu trúc

- Các NNLT cấp cao thời kì đầu được xây dựng theo mô hình lập trình hướng cấu trúc
- Chương trình được phân thành:
  - Các cấu trúc điều khiển (if-else, switch-case, while-do, for, …)
  - Các thủ tục hàm (void method(), int method(),…)
- Ví dụ C, COBOL, Fortran, LISP, Pearl, HTML, VBScript

Lập trình hướng cấu trúc

- Ví dụ: viết chương trình tính diện tích hình tròn

```
public class Circle {
    public static double tinhDienTichHinhTron(double banKinh) {
        double PI = 3.14;
        double dienTich = banKinh * banKinh * PI;
        return dienTich;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Nhập bán kính hình tròn: ");
        double r = input.nextDouble();
        double dienTich = tinhDienTichHinhTron(r);
        System.out.println("Diện tích hình tròn: " + dienTich);
    }
}
```

Toàn bộ code nằm trong hàm main()

Các biến được định nghĩa theo suy nghĩ cá nhân

Khó nhận biết được mối liên quan giữa các biến

Structured Programming

Ngôn ngữ hướng đối tượng

- NNLT sau này được xây dựng dựa trên mô hình lập trình hướng đối tượng (OOP)
- OOP không tập trung vào việc xây dựng cấu trúc chương trình mà đặt trọng tâm vào việc ***mô hình hóa dữ liệu***
- Phù hợp cho việc phát triển phần mềm có tính đa ứng dụng và có khả năng tái sử dụng cao
- Ví dụ C++, Java, Visual Basic .NET, C#, Python,…

Lập trình hướng đối tượng là gì ?

```java
public class Circle {
    private double banKinh;
    private double PI = 3.14;
    private double dienTich;

    public void input() {
        Scanner input = new Scanner(System.in);
        System.out.print("Nhập bán kính hình tròn: ");
        this.banKinh = input.nextDouble();
        input.close();
    }
    public void tinhDienTich() {
        this.dienTich = this.banKinh * this.banKinh * this.PI;
    }

    public void output() {
        System.out.println("Diện tích hình tròn: " + this.dienTich);
    }

    public static void main(String[] args) {
        Circle hinhTron = new Circle();
        hinhTron.input();
        hinhTron.tinhDienTich();
        hinhTron.output();
    }
}
```

Dữ liệu của chương trình sẽ được liên kết thành các đối tượng

Chương trình sẽ quản lý dựa trên các đối tượng

Hệ thống sẽ được mô tả cách trực quan hơn, phản ánh đúng thế giới hơn

Object-Oriented Programming

Khái niệm class và object

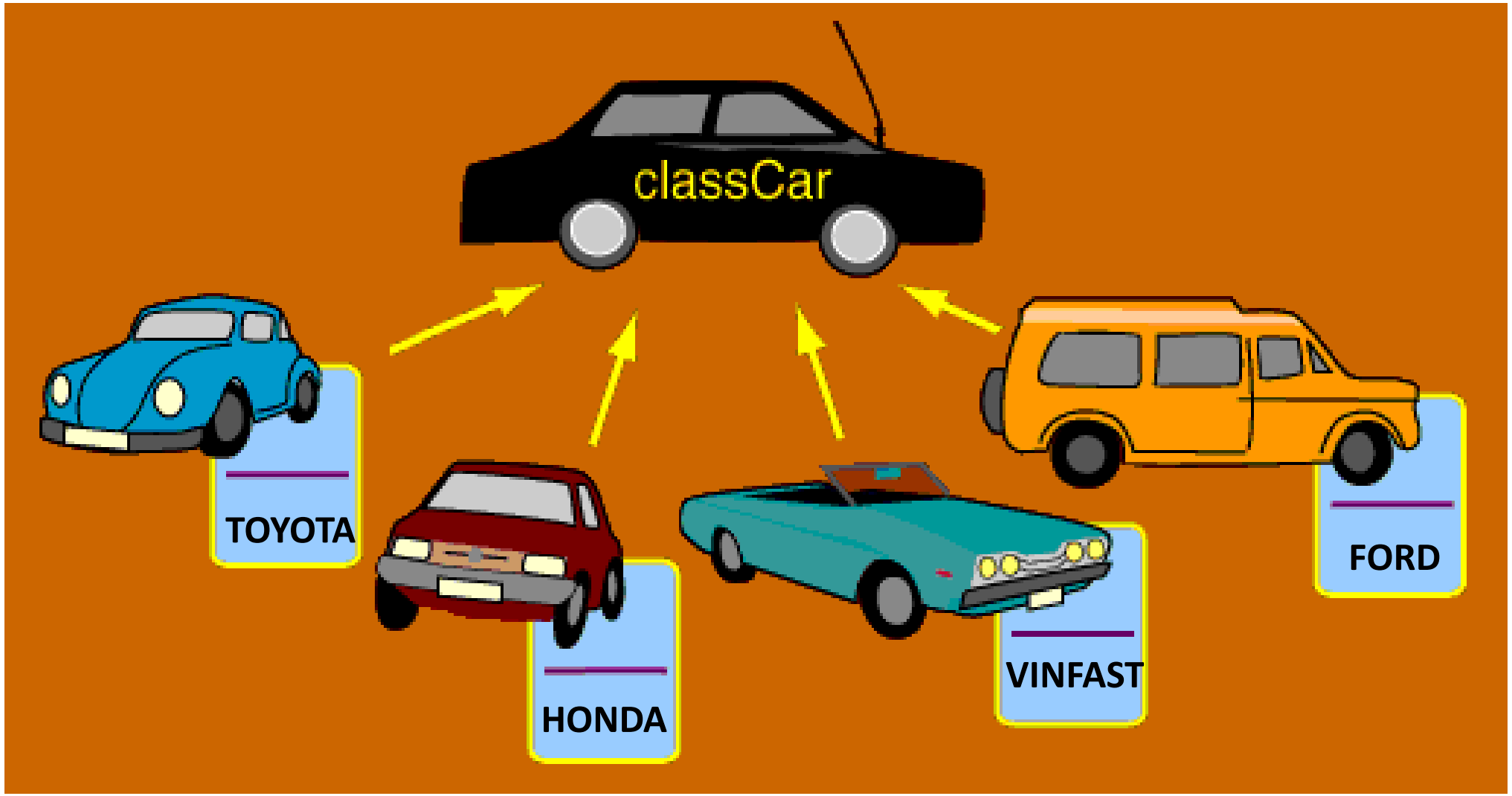
- Đối tượng chính là các khái niệm bên ngoài: Con người, xe cộ, máy móc,…
- Trong một đối tượng sẽ bao gồm hai thông tin: **đặc điểm** và **hành vi**
  - **Đặc điểm**: là những thông tin đặc trưng của đối tượng. Ví dụ: Sinh viên có họ tên, MSSV, khóa học, điểm số, lớp học,…
  - **Hành vi**: là những thao tác, hành động mà đối tượng đó có thể thực hiện. Ví dụ: Sinh viên có những hành động như đăng kí, đóng tiền, tham gia lớp, nộp bài,…

Khái niệm class và object

- Lớp (Class) chính là định nghĩa của đối tượng
- Lớp là tập hợp các thuộc tính (*properties*) cùng với những phương thức (*methods*) đặc trưng cho một loại đối tượng nào đó mà nó biểu diễn
- Ví dụ: Bạn **Nguyễn Văn A** và **Trần Văn B** đều là **sinh viên**, mà sinh viên thì đều có tên, tuổi, mã số, lớp học, điểm số,…Tuy nhiên, thông tin của mỗi bạn lại khác nhau, bạn A học lớp 19DTH3A còn bạn B học lớp 19DTH2D

➢Như vậy, **sinh viên** chính là **lớp**, còn **Nguyễn Văn A** và **Trần Văn B** là những **đối tượng**

Minh hoạ cho Class - Objects



Cú pháp khai báo class

- Trong Java để khai báo một lớp ta có cú pháp như sau:

```
class <<ClassName>>
{
    <<type>> <<field1>>;
    ...
    <<type>> <<fieldN>>;

    <<type>> <<method1>>([parameters]) {
            // body of method
    }
    ...
    <<type>> <<methodN>>([parameters]) {
            // body of method
    }
}
```

Khai báo các trường

Khai báo các phương thức

Thành phần dữ liệu

Thành phần xử lý

Minh họa: Khai báo Class Employee

```
public class Employee{
    public String fullname;
    public double salary;

    public void input(){
        Scanner scanner = new Scanner(System.in);
        System.out.print(" >> Full Name: ");
        this.fullname = scanner.nextLine();

        System.out.print(" >> Salary: ");
        this.salary = scanner.nextDouble();
    }

    public void output(){
        System.out.println(this.fullname);
        System.out.println(this.salary);
    }
}
```

Trường

Phương thức

Lớp Employee có 2 thuộc tính là fullname và salary và 2 phương thức là input() và output()

Tạo & sử dụng đối tượng

- Tạo đối tượng của 1 class đã định nghĩa
- Cú pháp

  <ClassName> <object>;

  <object> = **new** <ClassName>();

- Cú pháp

  <ClassName> <object> = **new** <ClassName>();

Tạo & sử dụng đối tượng

- Truy xuất đến thành phần dữ liệu và xử lý thuộc class
- Cú pháp

  <object>.<thuocTinh>;

  <object>.<phuongThuc>();

Ví dụ: tạo và sử dụng object

```
public static void main(String[] args) {

    Employee emp = new Employee();
    emp.input();
    emp.output();
}
```

- Chú ý:
  - Toán tử **new** được sử dụng để tạo đối tượng
  - Biến emp chứa tham chiếu tới đối tượng
  - Sử dụng dấu chấm (.) để truy xuất các thành viên của lớp (trường và phương thức).

Bài tập tạo class

- Khai báo class **sinhVien**
- Khai báo class **hangHoa**
- Khai báo class **phuongTienGiaoThong**
- Khai báo class **nhanVien**
- Bài toán: Viết chương trình nhập vào một phân số. Rút gọn phân số đó và xuất ra kết quả

Phương thức khởi tạo - Constructor

- **Constructor** là một loại phương thức đặc biệt của lớp, dùng để khởi tạo một đối tượng

```
class Student{
    // Thành phần dữ liệu
    String fullName;
    String studentID;

    // Thành phần xử lý
    Student() {
        // Đây là phương thức khởi tạo của lớp Student
        fullName = "Trần Văn A";
        studentID = "ST3849275";
    }
    // Các phương thức khác ...
}
```

Phương thức khởi tạo - Constructor

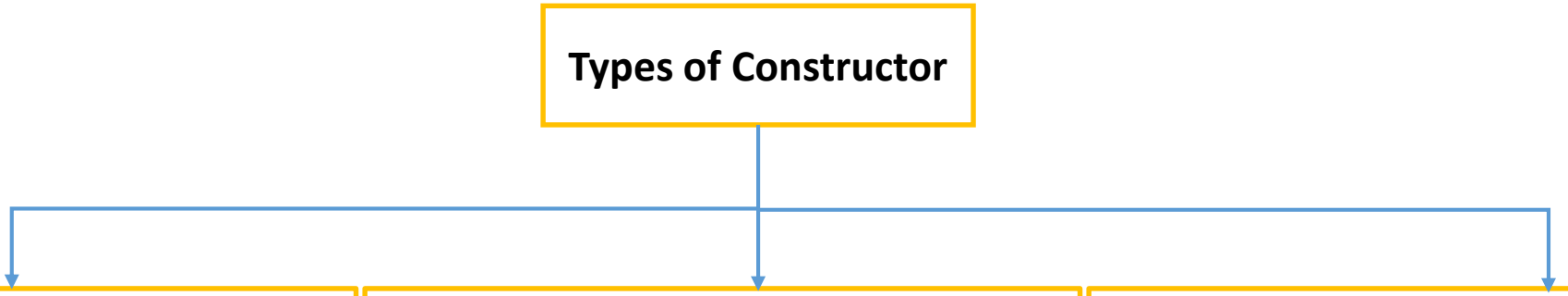
- Đặc điểm của **Constructor**
  - Cùng tên với tên lớp
  - Không có giá trị trả về
  - Tự động thi hành khi đối tượng được tạo ra bằng từ khóa new
  - Có thể có hoặc không có tham số

Phương thức khởi tạo - Constructor

- Khi nào nên sử dụng **constructor**
  - Khởi tạo giá trị cho các thuộc tính của đối tượng
  - Khởi tạo các đối tượng bên trong một đối tượng khác

Phương thức khởi tạo - Constructor

- Có bao nhiêu loại **constructor**




```java
public class Employee {

    String employeeName;
    int employeeID;

    // No Constructor

    public static void main(String[] args) {
        Employee emp = new Employee();
    }
}
```

Không có Constructor

```java
public class Employee {

    String employeeName;
    int employeeID;

    Employee(){
        // Constructor without Parameter
    }

    public static void main(String[] args) {
        Employee emp = new Employee();
    }
}
```

Constructor không tham số

```java
public class Employee {

    String employeeName;
    int employeeID;

    Employee(String name, int id){
        // Constructor with Parameters
        employeeName = name;
        employeeID = id;
    }

    public static void main(String[] args) {
        Employee emp = new Employee("Đặng Văn Lâm", 23);
    }
}
```

Constructor có tham số

Một số lưu ý khi xây dựng constructor

- Tên của **Constructors** phải trùng với tên của lớp mà nó trực thuộc
- **Constructors** phải được khai báo **public** và không có kiểu trả về, đồng thời cũng không được sử dụng khai báo **void** ở phía trước tên hàm
- Có thể định nghĩa nhiều **Constructors** khác nhau theo nguyên tắc về hàm nạp chồng (Function Overloading)
- Trong trường hợp có nhiều **Constructors** được định nghĩa cho lớp thì khi khai báo các đối tượng của lớp đó, ta phải chỉ ra các tham số sao cho phù hợp với từng **Constructors** muốn sử dụng để trình biên dịch có thể nhận biết và phân biệt giúp cho việc gọi chính xác **Constructors** theo yêu cầu của ngươi lập trình.

Nhớ gì ?!!!

- Khái niệm về Class và Object
- Cách khai báo và sử dụng Class và Object
- Phương thức khởi tạo Constructor

Tài liệu tham khảo

- Y. Daniel Lang, "**Introduction to Java Programming Comprehension Version**" 10th Edition.
- Jose M. Garrido, "**Object-Oriented Programming: From Problem Solving to Java**"
- Paul Deitel, Harvey Deitel, "**Java : How to program**", 9th edition, 2012
- Oracle, "**The Java™ Tutorials**", https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html, 6:20PM, 18/01/2018
- Java tutorial, https://howtodoinjava.com/java/basics/, 15:00PM, 20/02/2020