JAVA OOP

Dẫn nhập - Các khai báo

Các ví dụ dẫn nhập

Game, Nhân Viên, Xe

Đối tượng

Thành phần chính của đối tượng

Lớp Đối tượng

Khai báo, định nghĩa lớp

Đối tượng bao gồm gì?

Mã nhân viên
Tên nhân viên
CMND
Email
SĐT
Giới tính
……

THUỘC TÍNH

tinhLuong();
tongGioLam();
tongNgayNghi();
….

PHƯƠNG THỨC

Đối tượng bao gồm gì?

Mã Xe
Tên Xe
Thương hiệu
Năm sản xuất
Màu xe
Trọng lượng
Công suất máy
Chiều dài
Chiều rộng
Chiều cao
Giá nhập
Giá bán
Số lượng trong kho

THUỘC TÍNH

tinhThue();
xuatKho();
…..

PHƯƠNG THỨC

Khai báo lớp trong Java

```
public class NhanVien
    //Thuộc tính
{   public String maNV;
    public String tenNV;
    public String cmnd;
    public String email;
    public String sdt;
    public Boolean
    gioiTinh;
    //Phương thức
    public void tinhLuong();
    public void tongGioLam();
    public void tongNgayNghi();
}
```

1. Class : từ khóa để khai báo lớp
2. Tên lớp cần tạo
3. Dẫn xuất (Modifier - trình bày sau)

NOW, Let’s do them - Giới thiệu UML - draw.io tool

Các bạn liệt kê LỚP đối tượng, các thuộc tính và phương thức có thể có cho các bài toán sau đây: (CODE TRÊN GIẤY hoặc draw.io tool)

1. Mô tả lớp đối tượng cho phép thực hiện các phép tính Cộng, Trừ, Nhân, Chia các Phân Số
2. Mô tả lớp đối tượng cho phép quản lý Tài khoản ngân hàng của người dùng.
3. Mô tả lớp đối tượng cho phép quản lý thông tin Sinh Viên
4. Mô tả lớp đối tượng cho phép quản lý các Bệnh án điện tử
5. Mô tả lớp đối tượng cho phép quản lý thông tin Chuyến bay.
6. Mô tả lớp đối tượng cho phép quản lý thông tin Cầu thủ.
7. Mô tả lớp đối tượng cho phép quản lý thông tin Sổ tiết kiệm
8. Mô tả lớp đối tượng cho phép quản lý Lớp học (thông tin lớp và DS Học sinh)
9. Mô tả lớp đối tượng cho phép quản lý Công ty (thông tin công ty và DS nhân viên)
10. Mô tả lớp đối tượng cho phép quản lý Trường học (thông tin trường và DS các Lớp)

JAVA OOP

Instance - Code mẫu

Instance

Thể hiện lớp đối tượng

Một số lưu ý

Các best practice khi lập trình hướng đối tượng

Code mẫu thực tế

Phân tích, thực hiện code

Instance - Thể hiện của lớp đối tượng

```
public  class  NhanVien
{
    //Thuộc tính
     String maNV;
     String tenNV;
     String cmnd;
     String email;
     String sdt;
     Boolean gioiTinh;
    //Phương thức
     public void tinhLuong();
     publiic void tongGioLam();
     public void tongNgayNghi();
}
```

Tên Lớp — Từ khóa

```
NhanVien nhanVien1 = new NhanVien();
```

Tên đối tượng — Hàm khởi tạo

```
NhanVien nhanVien2 = new NhanVien();
```

Đối tượng **nhanVien1** hay ***instance*** của lớp NhanVien

Đối tượng **nhanVien2** hay ***instance*** của lớp NhanVien

INSTANCE - THỂ HIỆN CỦA LỚP ĐỐI TƯỢNG

1. Muốn xài được lớp đối tượng → Phải tạo các đối tượng cụ thể. Một trường hợp cụ thể của lớp đối tượng đối tượng → Thể hiện của lớp đối tượng (Instance of Class)
2. Sử dụng từ khóa *new* để tạo ra thể hiện của lớp đối tượng. Lúc này máy tính sẽ cấp phát một vùng nhớ cho đối tượng

MỘT SỐ GHI NHỚ

- Đối tượng : Object
- Lớp đối tượng : Class
- Thuộc tính/Dữ liệu/Biến thành viên : Attributes/ Data members
- Phương thức (Method)
- Thể hiện của Lớp đối tượng : (Instance)
- Naming convention :

✓ Lớp (Class) bắt đầu bằng chữ hoa và là một danh từ

  - NhanVien, SinhVien, Students, …

✓ Trường dữ liệu/thuộc tính (Fields /Attributes) bắt đầu bằng chữ thường

  - tuSo, mauSo, hoVaTen,…

✓Phương thức (Method) bắt đầu bằng chữ thường

  - tinhLuong(), cong(), chuyenLop()

PHƯƠNG THỨC KHỞI TẠO (CONSTRUCTOR)

```
public class DemoSinhVien
    public static void main ( String [ ] args )
    {
        SinhVien sv1 = new SinhVien();
        SinhVien sv2 = new SinhVien();
        sv1.hoTen = “CyberLearn”;
        sv1.email = “abc@gmail.com";

        sv2.hoTen = “CyberSoft”;
        sv2.email = “cyber@gmail.com”;

        System.out.println ( “Tên: “ + sv1.hoTen + ”- Email: “ +
sv1.email );
        System.out.println ( “Tên: “ + sv2.hoTen + ”- Email: “ +
sv2.email );
}
```

- Constructor là một Method đặc biệt được sử dụng để khởi tạo đối tượng (Object)
- Constructor được gọi tại thời điểm object được tạo
- Phải cùng tên với tên class
- Không có kiểu trả về
- Có thể có tham số hoặc không có tham số

DẪN XUẤT - TÍNH ĐÓNG GÓI

- Bước 4: Thay đổi private cho thuộc tính của lớp SinhVien

```
public class SinhVien

{

    private String hoTen;
   //Phương thức khởi tạo mặc định không tham số
    public String email;


   public SinhVien(){

   }

   //Phương thức khởi tạo sử dụng 2 tham số để

   // truyền dữ liệu cho thuộc tính

   public SinhVien( String hoTen, String email){

        this.hoTen = hoTen;

       this.email = email;

   }


}
```

DẪN XUẤT (ACCESS MODIFIER)

- private : chỉ được xài trong lớp đó, ra ngoài lớp không được xài
- public : Xài nơi nào cũng được
- protected: Các lớp con cháu dòng họ được xài (Kế thừa sẽ thấy)
- Lưu ý: Các biến thành viên của lớp đối tượng thường để private

→ che dấu thông tin của lớp đó.

TÍNH ĐÓNG GÓI (ENCAPSULATION)

- Muốn cho bên ngoài thấy hoặc che dấu đi
- Tạo ra cơ chế để ngăn ngừa việc gọi phương thức của lớp này tác động hay truy xuất dữ liệu của đối tượng thuộc về lớp khác.
- Không quan tâm bên trong code gì, chỉ cho bên ngoài thấy cách xài, ví dụ các phương thức (tính lương, tính tiền,….)
- Dễ dàng thay đổi code bên trong lớp đó mà không ảnh hướng lớp khác
- Dễ dàng nâng cấp và bảo trì vì không đụng các lớp khác.
- Ngăn ngừa việc gán các giá trị không hợp lệ vào thành phần dữ liệu (Attribute) của mỗi đối tượng.

DẪN XUẤT - TÍNH ĐÓNG GÓI

```
public class SinhVien {
        // thuộc tính / biến thành viên / data members
        private String hoTen;
        private String email;

        // Accessor methods (phương thức truy xuất các thuộc tính/biến thành viên)
        public String getHoTen(){
                return hoTen;
        }

        public String getEmail(){
                return email;
        }

        // phương thức thiết lập giá trị các thuộc tính/biến thành viên)
```

MỘT SỐ GHI NHỚ

* **get** : cung cấp cho bên ngoài xài, muốn cho xài thì dùng get

* **set** : đưa giá trị từ bên ngoài vào cho thuộc tính thông qua tham số

* Không phải lúc nào cũng có cả **set** và **get** —> Demo thêm thuộc tính điểm trung bình

LET'S CODE NOW

- Demo Sửa lại code cho lớp SinhVien với các phương thức get/set để chạy
- Demo tính đóng gói cho phương thức *tinhDiemTB* và gọi bên ngoài hàm chính. Nếu không có sự ngăn chặn thuộc tính get thì có thể set được điểm trung bình —> vi phạm nguyên tắc hướng đối tượng (Sửa điểm Hà Giang)

MỘT SỐ PHƯƠNG THỨC THƯỜNG XÀI TRONG ARRAYLIST & LINKEDLIST

- ArrayList<SinhVien> listSV = new ArrayList<SinhVien>()
- listSV.add(sv); // Thêm mới đối tượng vào danh sách
- listSV.remove(sv); // Xóa đối tượng khỏi list
- listSV.get(i); // Lấy đối tượng tại vị trí i

- Duyệt for

```
for (int i = 0; i < listSV.size(); i++) {
      listSV.get(i); // Lấy sinh viên thứ i
}
```

→
* Nếu listSV bị null thì sẽ lỗi
* Phức tạp khi lấy đối tượng xài
* **Lợi**: tìm chỉ số trong danh sách

- Duyệt for each

```
for (SinhVien sv : listSV) {
    // Lấy đối tượng sv và xài
}
```

→
* Nếu listSV bị null thì bỏ qua ko chạy
* Thao tác dễ trên đối tượng xài
* **Nhược**: Nếu cần chỉ số phải thêm biến phụ