

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

Ví dụ 1: Đối tượng trong game



Danh sách các đối tượng

Nhân vật
Quái
Trụ lửa
Trụ năng lượng
Màn chơi

Ví dụ 2: Đối tượng Nhân viên 1



Mô tả một đối tượng nhân viên

Mã nhân viên : ms001

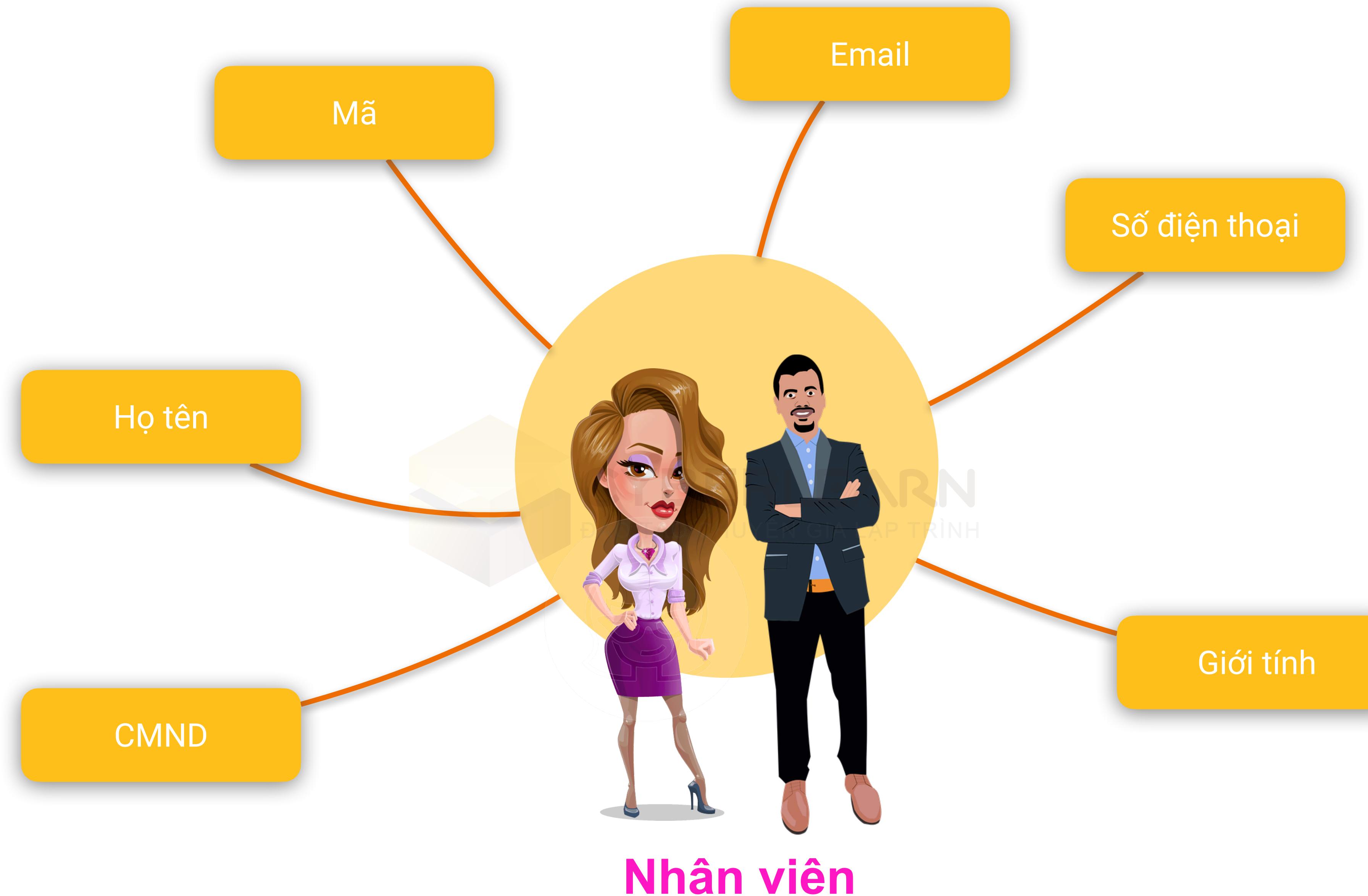
Tên nhân viên : Bưởi

CMND: 0234212

Email: buoi@gmail.com

SĐT : 0993343

Giới tính: Nữ



Đối tượng bao gồm gì?



Mã nhân viên
Tên nhân viên
CMND
Email
SĐT
Giới tính

*tinhLuong();
tongGioLam();
tongNgayNghi();*

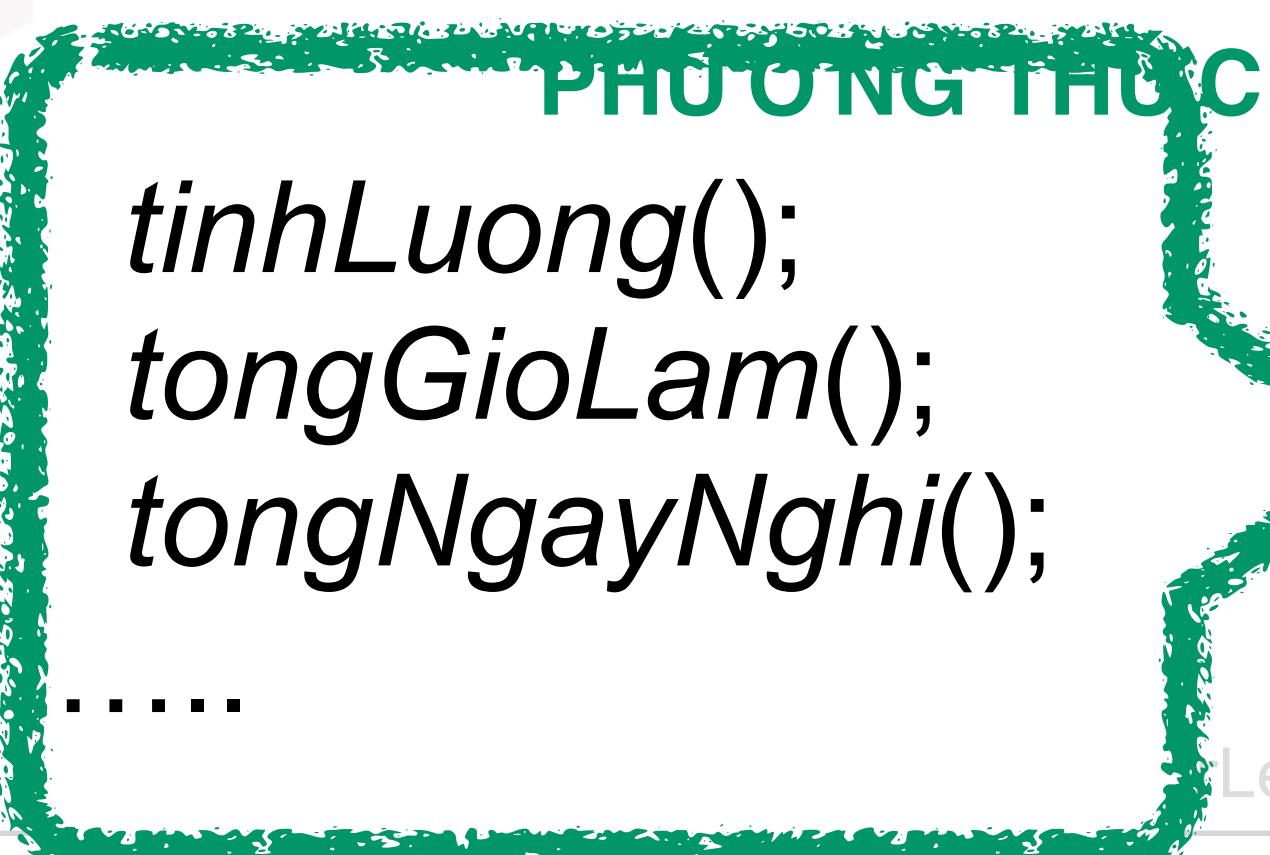
THUỘC TÍNH

PHƯƠNG THỨC

Khai báo lớp trong Java



Lớp: Nhân Viên



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();  
}
```

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

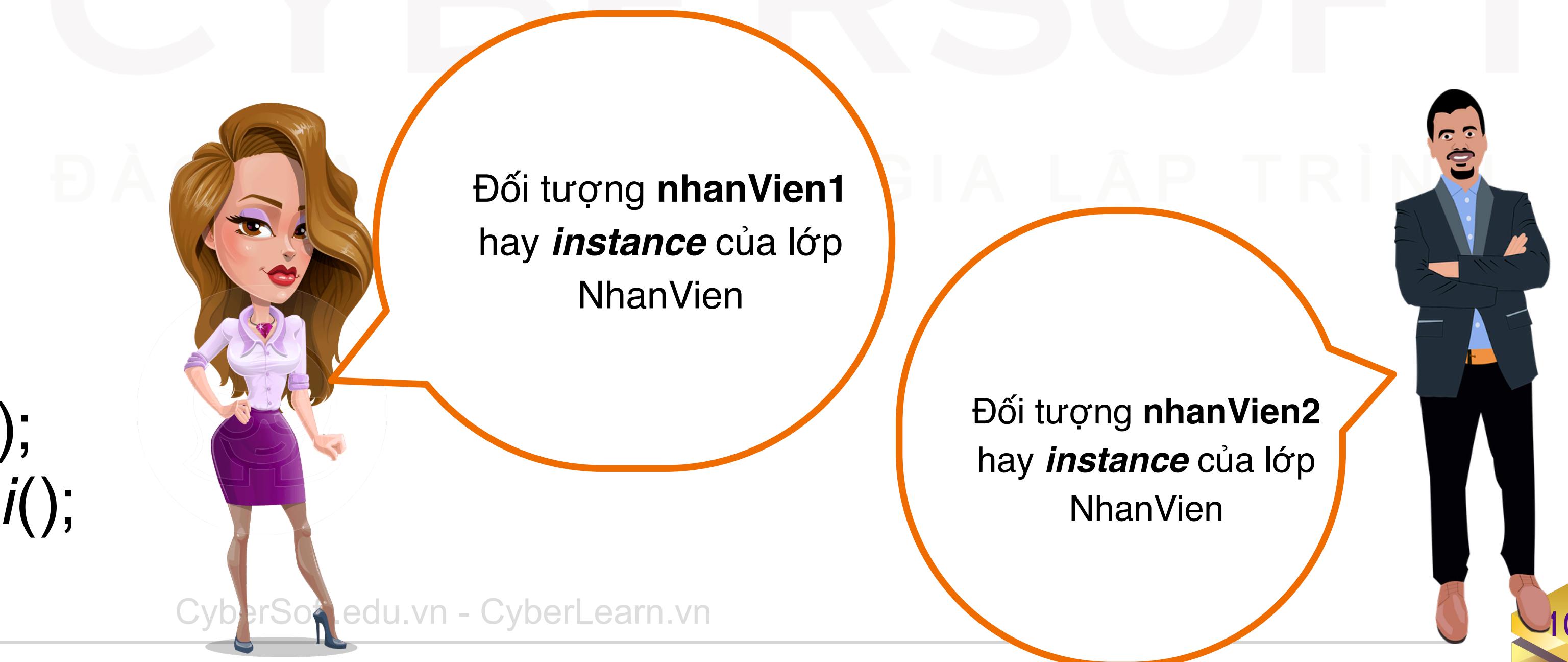
Instance - Thể hiện của lớp đối tượng

```
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();  
}
```



```
NhanVien nhanVien2 = new 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()

LET'S CODE NOW

- Bước 1: Tạo lớp đối tượng Sinh viên

```
public class SinhVien  
{  
    public String hoTen;  
    public String email;  
}
```



LET'S CODE NOW

- Bước 2: Vào hàm **main** và code theo sau:

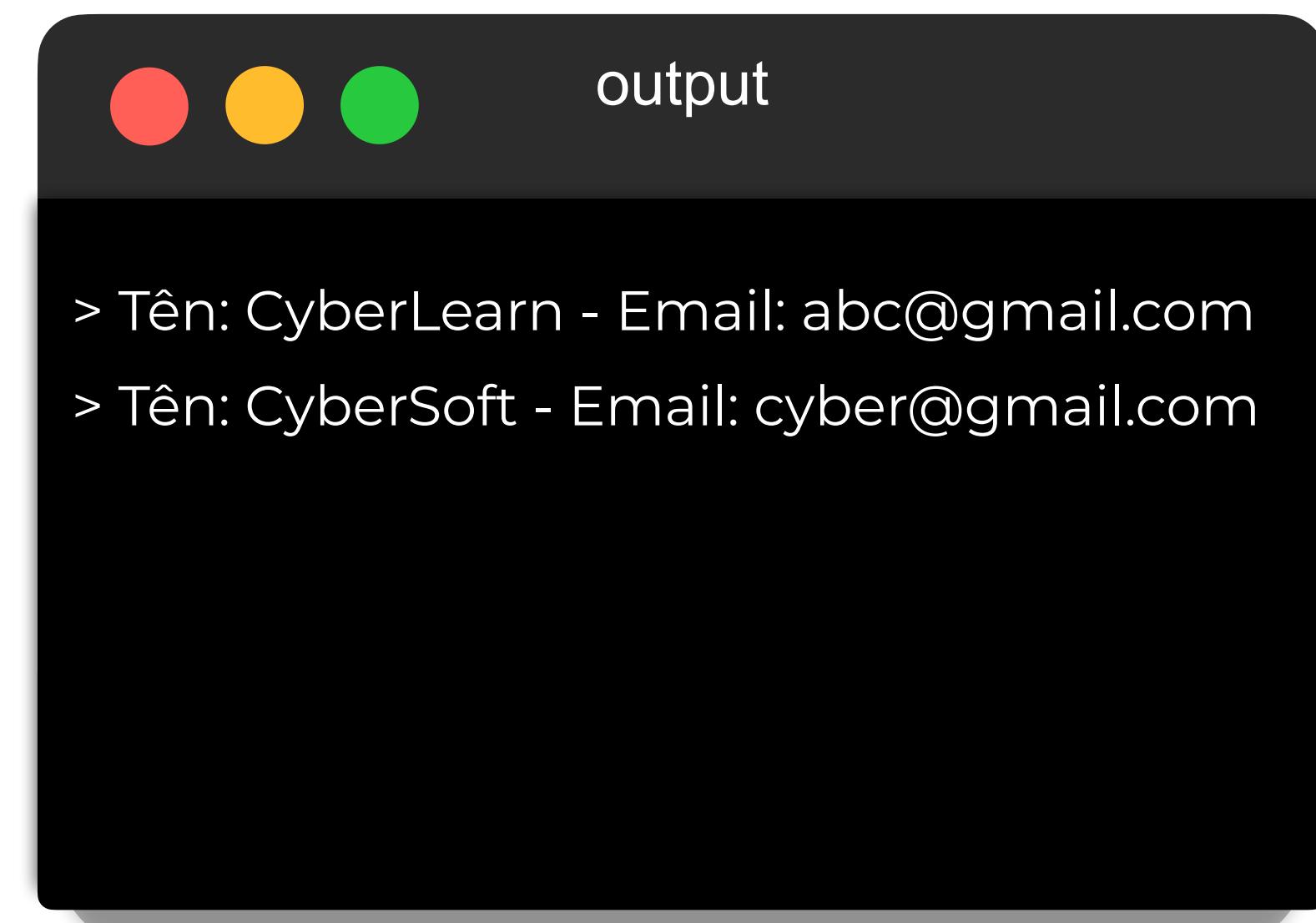
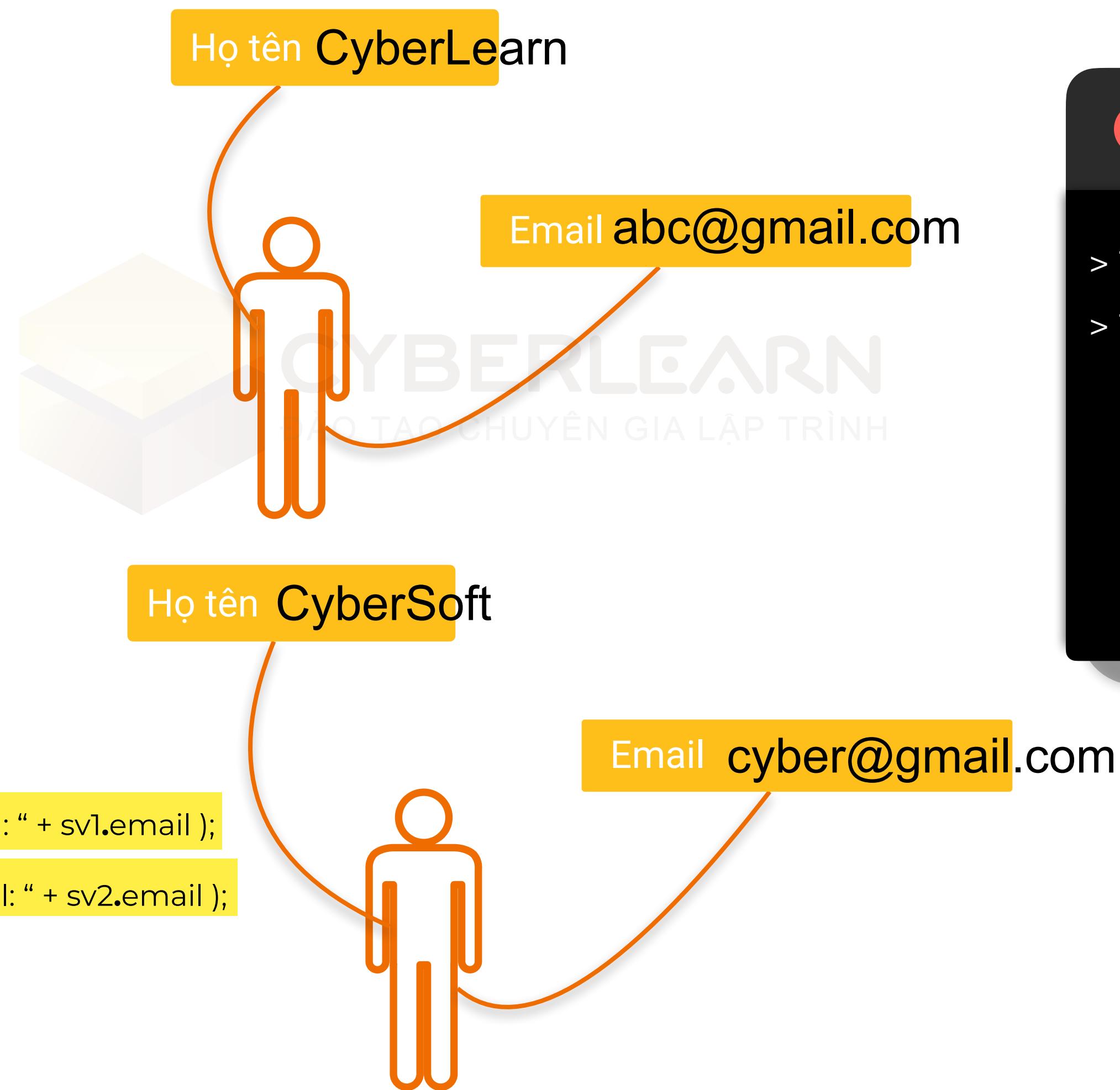
```
public class DemoSinhVien  
    public static void main ( String [ ] args )  
    {  
        SinhVien sv1 = new SinhVien();  
        SinhVien sv2 = new SinhVien();  
        sv1.hoTen = "CyberLearn";  
        sv1.email = "abc@gmail.com";  
  
        sv2.hoTen = "CyberSoft";  
        sv2.email = "cyber@gmail.com";  
  
        System.out.println ( "Tên: " + sv1.hoTen + "- Email: " + sv1.email );  
        System.out.println ( "Tên: " + sv2.hoTen + "- Email: " + sv2.email );  
    }
```



GIẢI THÍCH CODE

- Bước 2: Vào hàm **main** và code theo sau:

```
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 );  
    }  
}
```



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 );  
    }  
}
```

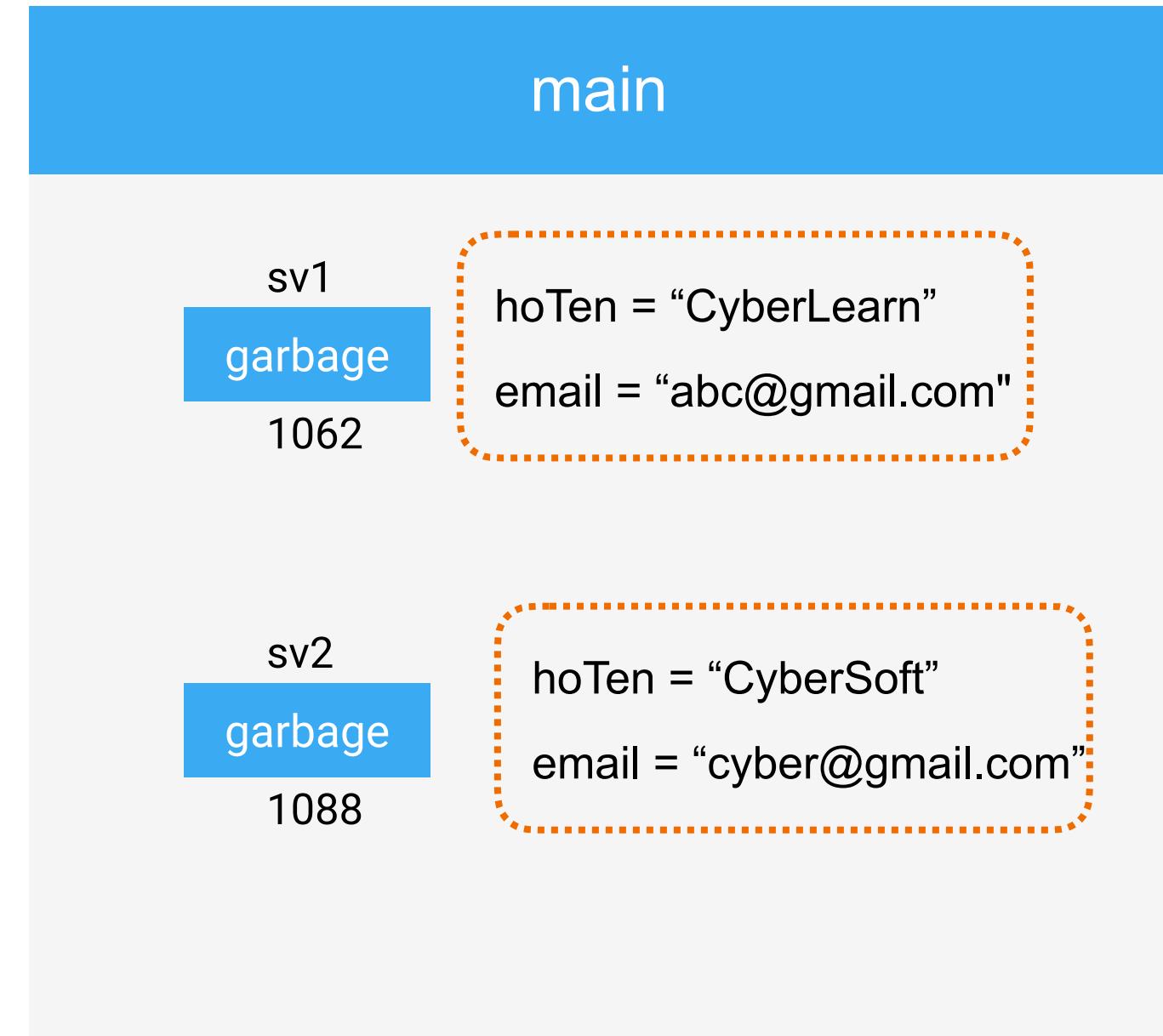
- **Constructor** là một Method đặc biệt được sử dụng để khởi tạo đối tượng (Object)
- **Constructor** được gọi tại thời điểm object được tạo
- Phải cùng tên với tên class
- Không có kiểu trả về
- Có thể có tham số hoặc không có tham số

DEMO PHƯƠNG THỨC KHỞI TẠO

- Bước 3: Mở lớp SinhVien và bổ sung các phương thức khởi tạo

```
public class SinhVien
{
    public String hoTen;
    public String email;
    //Phương thức khởi tạo mặc định không tham số
    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;
    }
}
```

```
public class DemoSinhVien
{
    public static void main ( String [ ] args )
    {
        SinhVien sv1 = new SinhVien();
        sv1.hoTen = "CyberLearn";
        sv1.email = "abc@gmail.com";
        SinhVien sv2 = new SinhVien("CyberSoft","cyber@gmail.com");
        System.out.println ( "Tên: " + sv1.hoTen + "- Email: " + sv1.email );
        System.out.println ( "Tên: " + sv2.hoTen + "- Email: " + sv2.email );
    }
}
```



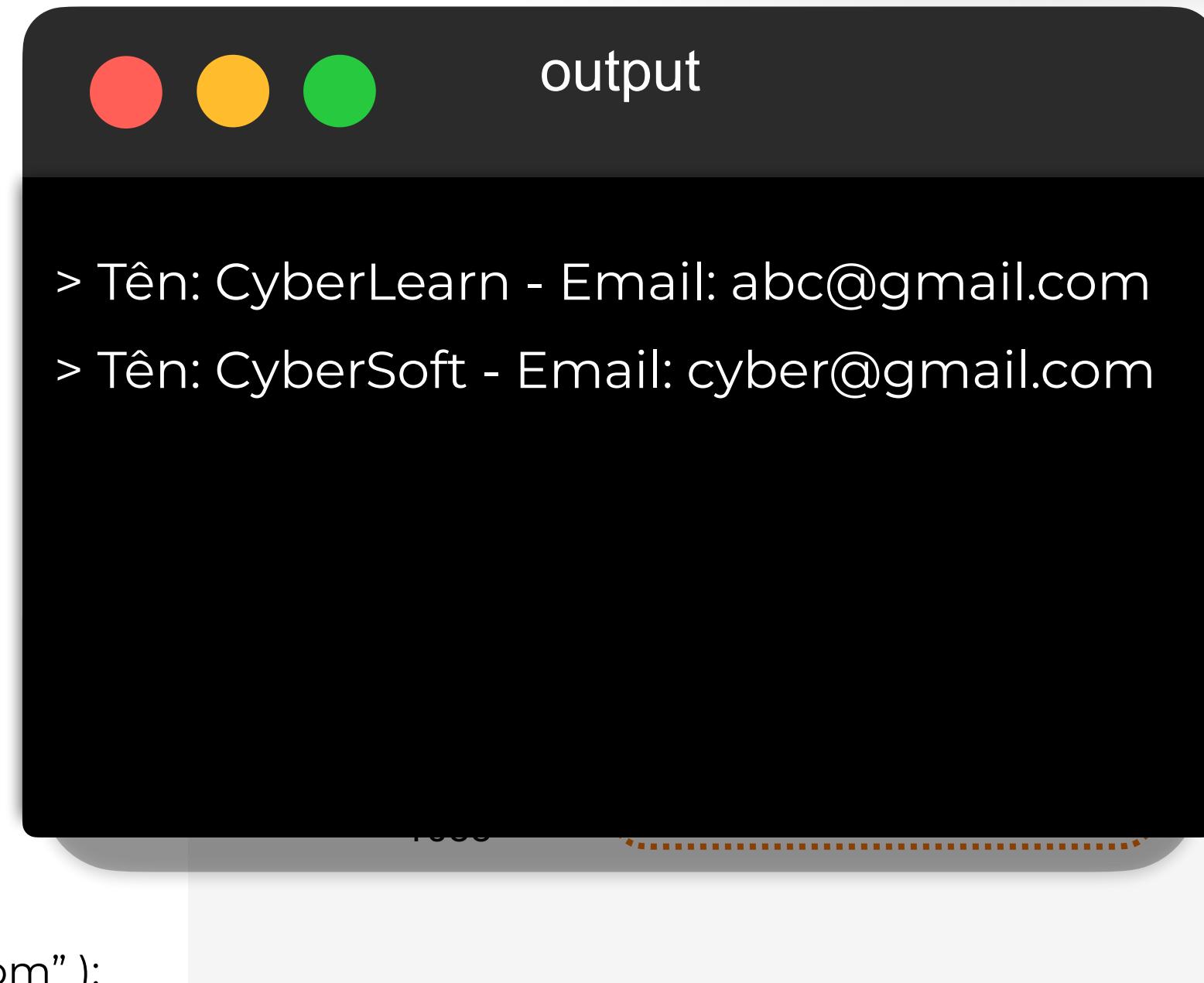
DEMO PHƯƠNG THỨC KHỞI TẠO

● Bước 3: Bổ sung các phương thức khởi tạo

```
public class SinhVien
{
    public String hoTen;
    public String email;
    //Phương thức khởi tạo mặc định không tham số
    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;
    }
}
```

```
public class DemoSinhVien
{
    public static void main ( String [ ] args )
    {
        SinhVien sv1 = new SinhVien();
        sv1.hoTen = "CyberLearn";
        sv1.email = "abc@gmail.com";
        SinhVien sv2 = new SinhVien("CyberSoft", "cyber@gmail.com" );
        System.out.println ( "Tên: " + sv1.hoTen + "- Email: " + sv1.email );
        System.out.println ( "Tên: " + sv2.hoTen + "- Email: " + sv2.email );
    }
}
```

 **this** → con trỏ của **thể hiện (instance)** của lớp hiện hành, khi nào xài ta gọi **new** và khi đó biến đối tượng sẽ chính là **this**



DẪN XUẤT - TÍNH ĐÓNG GÓI

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

```
public class SinhVien
{
    private String hoTen;
    private String email;
    //Phương thức khởi tạo mặc định không tham số
    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;
    }
}
```



CYBERLEARN
ĐÀO TẠO CHUYÊN GIA

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 hưởng lớp khác
- Dễ dàng nâng cấp và bảo trì vì không dụng 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  
    public void setHoTen ( String _hoTen){  
        this.hoTen = _hoTen;  
    }  
  
    public void setEmail ( String _email ){  
        this.email = _email;  
    }  
}
```





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

CYBERLEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

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)

DỰ ÁN: HÌNH CHỮ NHẬT

Xây dựng lớp đối tượng quản lý hình chữ nhật và cài đặt các phương thức tính chu vi và diện tích của hình chữ nhật.

- Bước 1: Xác định lớp đối tượng
- Bước 2: Liệt kê tất cả các thuộc tính của đối tượng
- Bước 3: Xây dựng các phương thức get, set cho từng thuộc tính
- Bước 4: Xây dựng các phương thức khởi tạo
- Bước 5: Xây dựng phương thức nhập, xuất
- Bước 6: Xây dựng phương thức xử lý nghiệp vụ tính chu vi (tạo thêm thuộc tính chu vi để lưu trữ)
- Bước 7: Xây dựng phương thức xử lý nghiệp vụ tính diện tích (tạo thêm thuộc tính diện tích để lưu trữ)
- Bước 8: Vào lớp xử lý chính hoặc nơi cần xử lý, tạo ra đối tượng
- Bước 9: Gọi các phương thức xử lý theo yêu cầu bài toán

DỰ ÁN: QUẢN LÝ SINH VIÊN - VERSION 1 - 1 LỚP ĐỐI TƯỢNG

Xây dựng chương trình cho phép người dùng nhập vào: Tên, Mã SV, điểm Toán, Lý Hóa. Thực hiện các nghiệp vụ dưới đây.

- Tính điểm trung bình từng sinh viên ($T + L + H)/3$)
- Xếp loại từng sinh viên theo: $>=9 \rightarrow$ Xuất Sắc, $9 < Giỏi \leq 8$, $8 < Khá \leq 7$, $< 7 < TB$ Khá ≤ 6 , $< 6 < TB \leq 5$, còn lại Yếu.

GIẢI THUẬT: QUẢN LÝ SINH VIÊN - VERSION 1 - 1 LỚP ĐỐI TƯỢNG

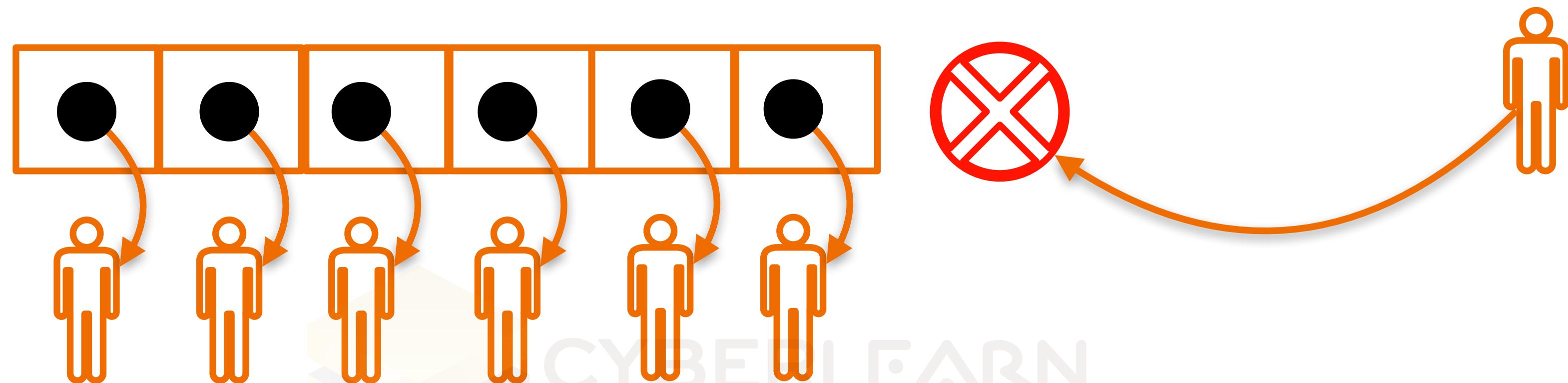
- Bước 1: Xác định lớp đối tượng
- Bước 2: Vẽ sơ đồ lớp UML
- Bước 3: Liệt kê tất cả các thuộc tính của đối tượng
- Bước 4: Liệt kê các phương thức đã thấy trước
- Bước 5: Xây dựng lớp đối tượng
- Bước 6: Xây dựng các phương thức get, set cho từng thuộc tính
- Bước 7: Xây dựng các phương thức khởi tạo
- Bước 8: Xây dựng phương thức nhập, xuất
- Bước 9: Xây dựng phương thức xử lý nghiệp vụ tính điểm trung bình (tạo thêm thuộc tính điểm trung bình để lưu trữ)
- Bước 10: Xây dựng phương thức xử lý nghiệp vụ xếp loại (tạo thêm thuộc tính xếp loại để lưu trữ)
- Bước 11: Vào lớp xử lý chính hoặc nơi cần xử lý, tạo ra đối tượng
- Bước 12: Gọi các phương thức xử lý theo yêu cầu bài toán

SỬ DỤNG ARRAYLIST & LINKEDLIST

SinhVien [] arrSV = new SinhVien [n] ;

n = 6

Thêm 1 sinh viên với ?

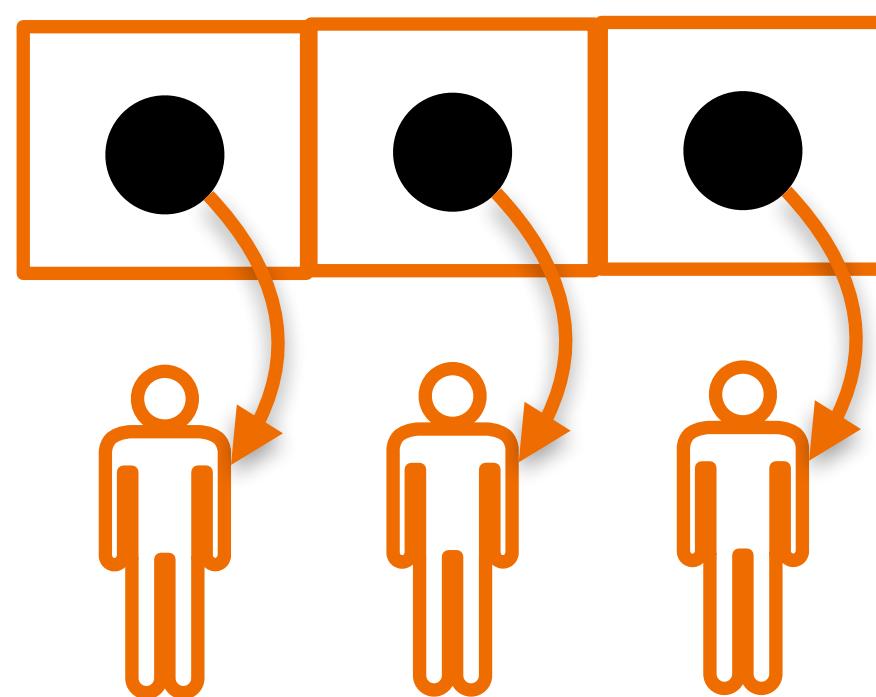


ArrayList<SinhVien> listSV = new ArrayList<SinhVien>()

listSV.add(sv);

listSV.add(sv);

listSV.add(sv)



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  
}
```

Duyệt for each

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

CYBERLEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

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

- * 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

DỰ ÁN: QUẢN LÝ SINH VIÊN - VERSION 2 - 2 LỚP ĐỐI TƯỢNG (LỚP DANH SÁCH)

Xây dựng chương trình cho phép người dùng nhập vào: Tên, Mã SV, điểm Toán, Lý Hóa. Thực hiện các nghiệp vụ dưới đây.

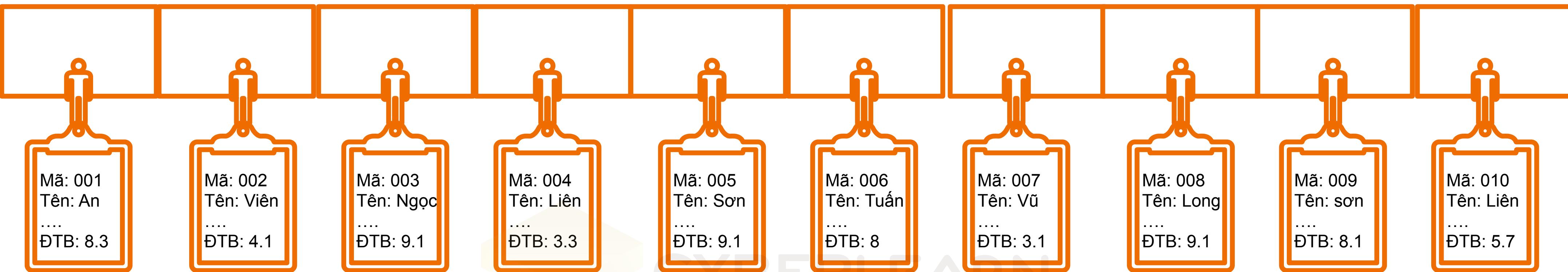
- Tính điểm trung bình từng sinh viên ($T + L + H)/3$)
- Xếp loại từng sinh viên theo: $>=9 \rightarrow$ Xuất Sắc, $9 < Giỏi \leq 8$, $8 < Khá \leq 7$, $< 7 < TB$ Khá ≤ 6 , $< 6 < TB \leq 5$, còn lại Yếu.

DỰ ÁN: QUẢN LÝ SINH VIÊN - VERSION 3 - 3 LỚP ĐỐI TƯỢNG

Xây dựng chương trình THEO HƯỚNG ĐỐI TƯỢNG cho phép người dùng nhập vào: Tên, Mã SV, điểm Toán, Lý Hóa. Cho phép nhập nhiều Sinh viên và thực hiện (Viết hàm tạo dữ liệu giả)

- Tính điểm trung bình từng sinh viên ($T + L + H)/3$)
- Xếp loại từng sinh viên theo: $>=9 \rightarrow$ Xuất Sắc, $9 < Giỏi \leq 8$, $8 < Khá \leq 7$, $7 < TB \leq 6$, $6 < TB \leq 5$, còn lại Yếu. In ra danh sách theo mẫu
- In ra SV có ĐTB cao nhất. In ra danh sách theo mẫu
- In ra tất cả sinh viên Yếu. In ra danh sách theo mẫu
- Tìm sinh viên theo tên. In ra danh sách theo mẫu
- Tìm sinh viên theo mã
- Xóa 1 sinh viên theo mã

GIẢI THUẬT: IN SINH VIÊN CÓ ĐIỂM TRUNG BÌNH CAO NHẤT



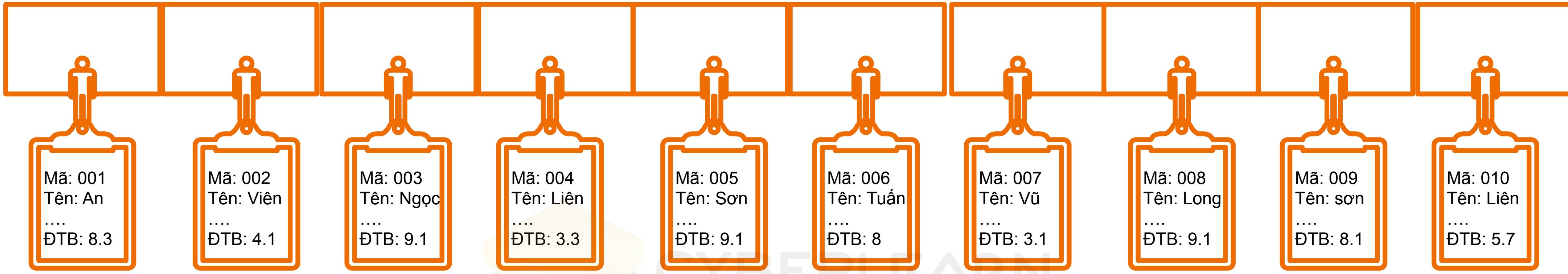
// Tìm sinh viên có điểm trung bình thực sự cao nhất trong danh sách, dùng thuật toán tìm kiếm tuyến tính

- Bước 1: Tạo biến **svMax** và gán sinh viên đầu tiên trong danh sách giả định là sinh viên có ĐTB cao nhất cho **svMax**
- Bước 2: Duyệt danh sách sinh viên
- Bước 3: So sánh ĐTB của sinh viên đang duyệt (**svCurrent**) với **svMax**, nếu **svCurrent.getDTB > svMax.getDTB**
- Bước 4: Gán lại **svMax = svCurrent**
- Bước 5: Tìm thấy được sinh viên có ĐTB lớn nhất thật sự trong danh sách

// Duyệt lại danh sách và tìm tất cả SV có ĐTB bằng với ĐTB của svMax và cho vào danh sách trả về

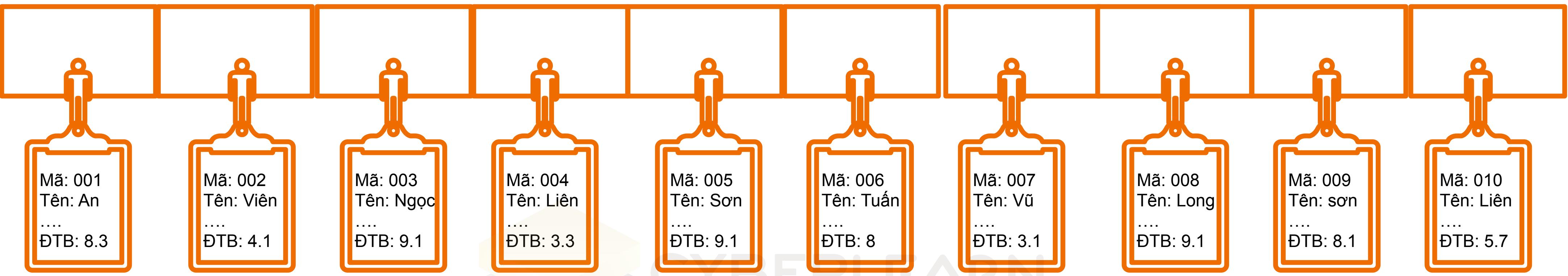
- Bước 6: Duyệt lại danh sách
- Bước 7: So sánh **svCurrent** với **svMax**, nếu **svCurrent.getDTB == svMax.getDTB**
- Bước 8: Thêm vào danh sách trả về
- Bước 9: Trả về danh sách sinh viên có ĐTB cao nhất

GIẢI THUẬT: IN DANH SÁCH SINH VIÊN YẾU



- Bước 1: Tạo biến `listYeu` để chứa danh sách sinh viên Yếu
- Bước 2: Duyệt danh sách sinh viên
- Bước 3: So sánh ĐTB của sinh viên đang duyệt `svCurrent.getDTB < 5`
- Bước 4: Nếu 3 đúng, thêm vào danh sách **`listYeu`**
- Bước 5: Trả về kết quả danh sách sinh viên Yếu

XỬ LÝ CÁC CÂU CÒN LẠI

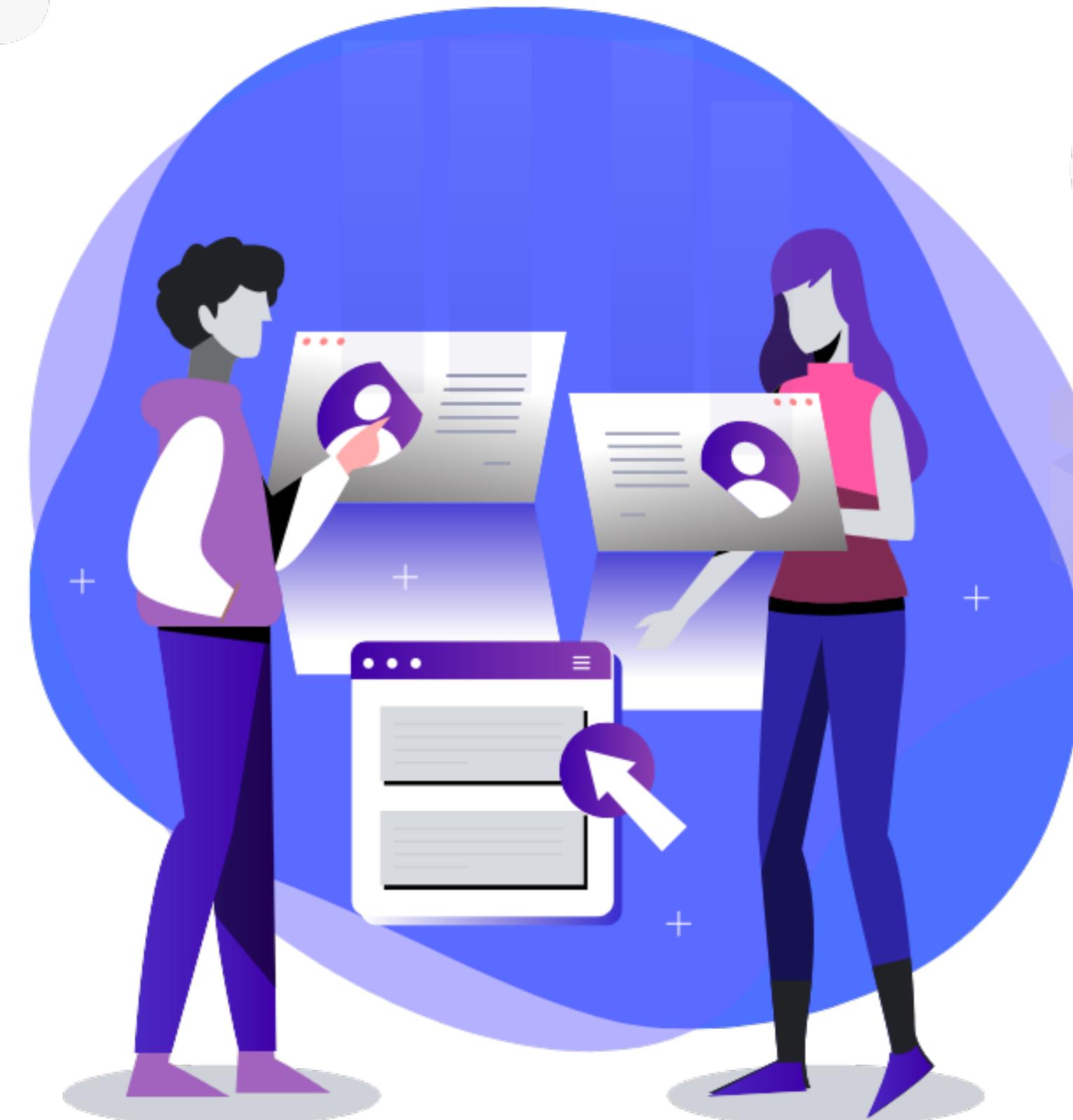


CYBERLEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

- Tìm sinh viên theo tên. In ra danh sách theo mẫu
- Tìm sinh viên theo mã
- Xóa 1 sinh viên theo mã

Một số điểm cần lưu ý

• • •



1. Phân tích nghiệp vụ → Xác định số lớp đối tượng → Vẽ sơ đồ lớp
2. Tạo các lớp đối tượng, mỗi lớp bao gồm:
 - Danh sách các thuộc tính Attributes.
 - Tập hợp các phương thức Get/Set
 - Tập hợp các phương thức khởi tạo (Cần có ít nhất 1 PT khởi tạo mặc định không tham số)
 - Các phương thức khác
 - Xử lý nhập, xuất
 - Xử lý nghiệp vụ
3. Phương thức nên được cài đặt tại lớp nào ? → Nếu nghiệp vụ cần xử lý liên quan đến một hoặc nhiều thuộc tính của lớp → Đặt trong lớp
4. Ra Lớp xử lý chính, tạo các Instance và gọi thực hiện.

Một số điểm cần lưu ý (TT)



Khi làm việc với hướng đối tượng cần nhớ:

- Muốn xài đối tượng thì đối tượng phải được khởi tạo (tạo **instance** hay **new**)
- Chú ý Null Pointer Exception vì đối tượng chưa được khởi tạo mà truy xuất đến các phương thức bên trong
- Tìm giá trị nhiều nhất cao nhất/thấp nhất thì chú ý sẽ có thể có nhiều giá trị cao nhất/thấp nhất cùng giá trị bằng nhau → Kết quả được danh sách đối tượng chứ không phải là 1 đối tượng