UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

# NGÔN NGỮ LẬP TRÌNH JAVA

## Chương 2
## HƯỚNG ĐỐI TƯỢNG TRONG JAVA (P1)

GVGD: ThS. Lê Thanh Trọng

# NỘI DUNG

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

1. Các khái niệm cơ bản
2. Khai báo lớp
3. Lớp nội
4. Tính đóng gói

# NỘI DUNG

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

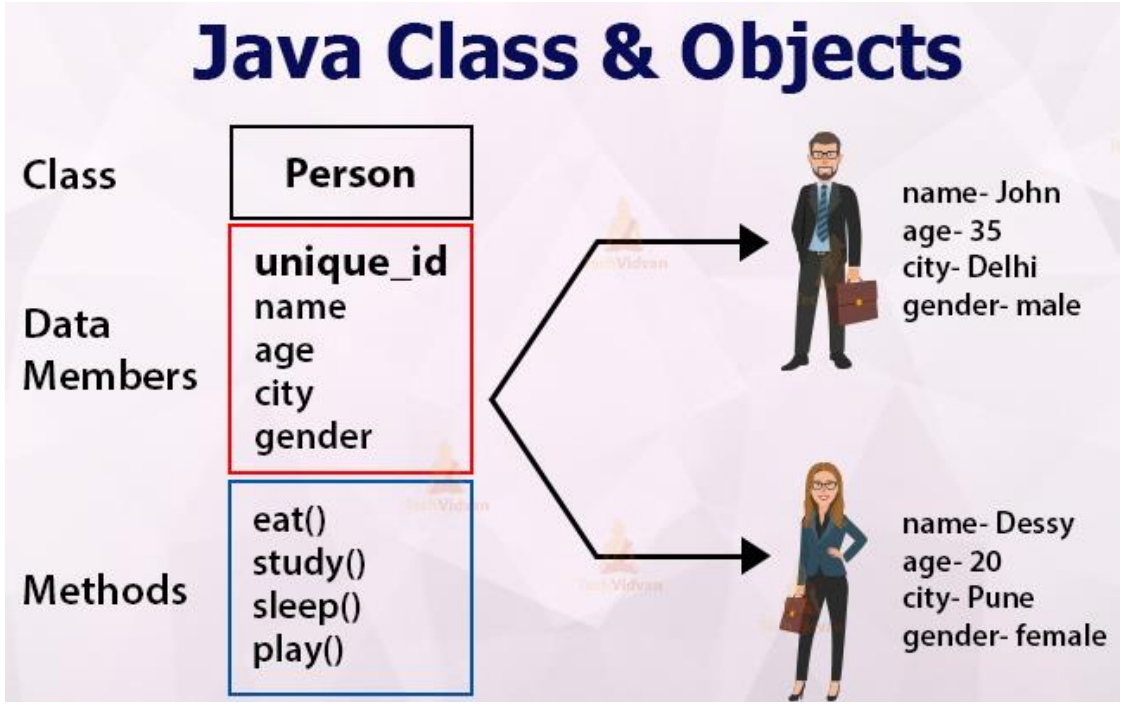
1. **Các khái niệm cơ bản**
2. Khai báo lớp
3. Lớp nội
4. Tính đóng gói

# Lớp

- Khuôn mẫu (template) để sinh ra đối tượng
- Sự trừu tượng hóa của tập các đối tượng có các thuộc tính, hành vi tương tự nhau, và được gom chung lại thành 1 lớp
  - Lớp người
  - Lớp cá
  - Lớp xe

# Lớp

- Thành phần chính
  - **Thuộc tính:** các thành phần dữ liệu đặc trưng
  - **Phương thức:** các hành vi/khả năng/chắc năng đặc trưng
  - **Constructors**
  - **Blocks**
  - **Nested class/interface**



# Lớp

```
public class Test{
    int Size;
    class SubClass{}
    public void myMethod()
    {
        System.out.println("Method");
    }
    {
        System.out.println(" Instance Block");
    }
     Test()
    {
        System.out.println("Constructor ");
    }
    static {
        System.out.println("static block");
    }
}
```

# Đối tượng

- Trong thế giới thực khái niệm đối tượng có thể xem như một thực thể: người, vật, bảng dữ liệu,…
- Đối tượng giúp hiểu rõ thế giới thực
- Cơ sở cho việc cài đặt trên máy tính
- mỗi đối tượng có định danh, thuộc tính, hành vi
- Ví dụ: đối tượng sinh viên

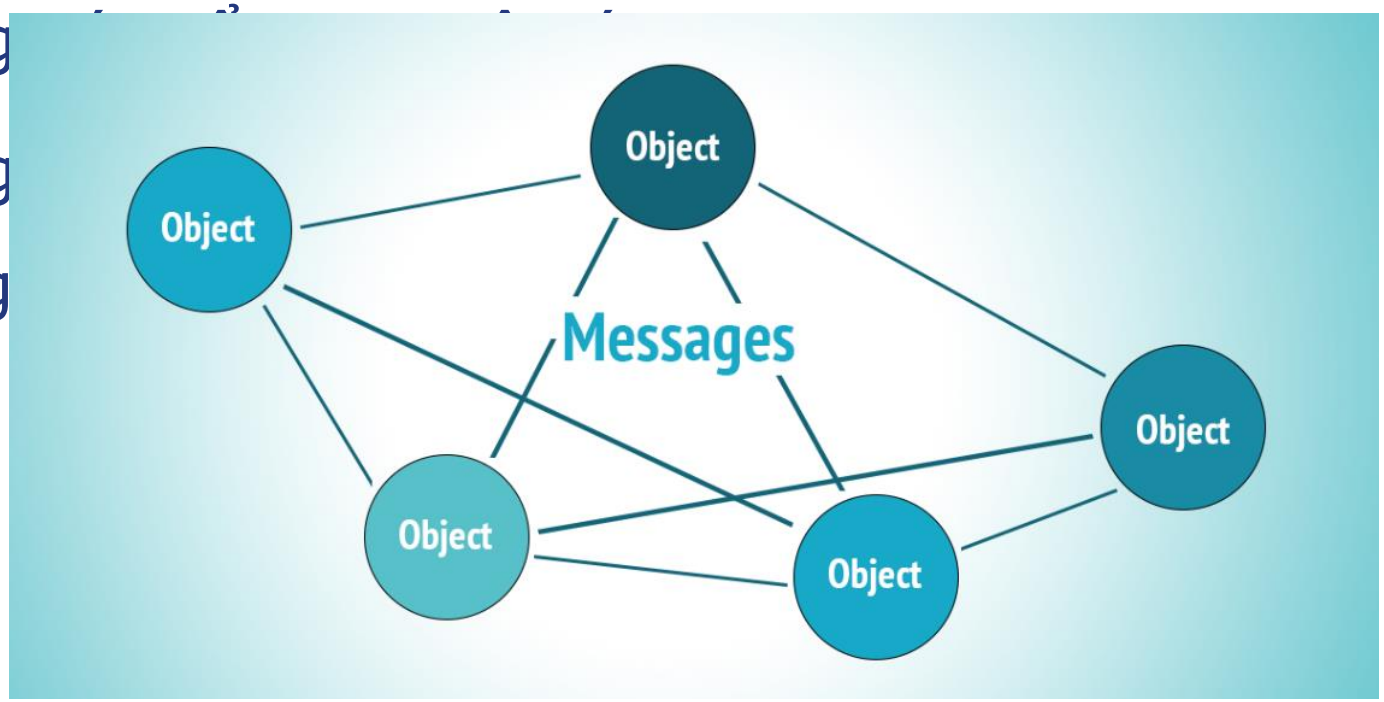
  MSSV: “TH0701001”; Tên sinh viên: “Nguyễn Văn A”
- Trong lập trình OOP, đối tượng là 1 thể hiện cụ thể của 1 lớp đó (1 biến thuộc kiểu lớp)

# Đối tượng

- Lớp **SinhVien** (HoTen, MSSV , DTB) có các đối tượng
  - "Nguyễn Văn A", mã số TH0701001, DTB 7.5
  - Sinh viên "Nguyễn Văn B", mã số TH0701002, DTB 6.8

# Hệ thống các đối tượng

- Là 1 tập hợp các đối tượng
- Mỗi đối tượng đảm trách 1 công việc
- Các đối tượng
- Các đối tượng
  ( thông qua g



# NỘI DUNG

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

1. Các khái niệm cơ bản
2. **Khai báo lớp**
3. Lớp nội
4. Tính đóng gói

# Khai báo lớp

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

*class <ClassName>*

*{*

*<khai báo các thuộc tính>*

*<các khởi tạo>*

*< khai báo các phươngthức>*

*}*

# Khai báo thuộc tính

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
class <ClassName>     {
        <Tiền tố> <kiểu dữ liệu> <tên thuộc tính>;
    }
```

- Kiểm soát truy cập đối với thuộc tính
  - **public**: có thể truy xuất từ bất kỳ 1 lớp khác
  - **protected**: có thể truy xuất được từ những lớp con
  - **default**: (không phải từ khóa) có thể truy cập từ các class trong cùng gói
  - **private**: không thể truy xuất từ 1 lớp khác
- Từ khóa khác
  - **static**: dùng chung cho mọi thể hiện của lớp
  - **final**: hằng

# Khai báo phương thức

```
class <ClassName> {
        …
        <Tiền tố> <kiểu trả về> <tên phương thức>(<các đối số>){
                …
        }
}
```

- Kiểm soát truy cập
  - **public**: có thể truy cập được từ bên ngoài lớp khai báo
  - **protected**: có thể truy cập được từ lớp cùng gói và và các lớp dẫn xuất (cùng hoặc khác gói)
  - **default**: (không phải từ khóa) có thể truy cập từ các class trong cùng gói
  - **private**: chỉ được truy cập bên trong lớp khai báo

# Khai báo phương thức

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- Từ khóa khác
  - **static**: phương thức lớp dùng chung cho tất cả các thể hiện của lớp, có thể được thực hiện kể cả khi không có đối tượng của lớp
  - **final**: không được khai báo chồng ở các lớp dẫn xuất
  - **abstract**: không có phần source code, sẽ được cài đặt trong các lớp dẫn xuất
  - **synchoronized**: dùng để ngăn những tác động của các đối tượng khác lên đối tượng đang xét trong khi đang đồng bộ hóa, dùng trong lập trình miltithreads

# Ví dụ 1

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
class SinhVien {
        // Danh sách thuộc tính
        String  MaSV, TenSV, DCLienLac;
        int     NamSinh;
        …
        // Danh sách các khởi tạo
        SinhVien(){}
        SinhVien (…)  { …}
        …
        // Danh sách các phương thức
        public void capNhatSV (…) {…}
        public void xemThongTinSV() {…}
        …
}
```

# Ví dụ 1

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
…
// Tạo đối tượng mới thuộc lớp SinhVien
SinhVien SV = new SinhVien();
…
// Gán giá trị cho thuộc tính của đối tượng
SV.MaSV = "TH0601001";
SV.TenSV = "Nguyen Van A";
SV.DCLienlac = "KP6, Linh Trung, Thu Duc";
SV.NamSinh = 1998;
…
// Gọi thực hiện phương thức
SV.xemThongTinSV();
```

# Ví dụ 2

```
class SinhVien {
        // Danh sách thuộc tính
        private String    MaSV;
        String    TenSV, DCLienlac;
        int       NamSinh;
        …
    }
    …
    SinhVien SV = new SinhVien();
    SV.MaSV = "TH0601001"; /* Lỗi truy cập thuộc tính private từ
                              bên ngoài lớp khai báo */
    SV.TenSV = "Nguyen Van A";
```

# Phương thức khởi tạo

- **Khởi tạo (constructor):** là một loại phương thức đặc biệt của lớp, dùng để khởi tạo một đối tượng.
  - Dùng để khởi tạo giá trị cho các thuộc tính của đối tượng
  - Cùng tên với lớp
  - Không có giá trị trả về
  - Tự động thi hành khi tạo ra đối tượng (new)
  - Có thể có tham số hoặc không
- **Lưu ý:**
  - Mỗi lớp sẽ có 1 constructor mặc định (nếu ta không khai báo constructor nào)
  - Ngược lại, nếu ta có khai báo 1 constructor khác thì constructor mặc định chỉ dùng được khi khai báo tường minh

# Khởi tạo

```
class SinhVien
{
        ...
        // Không có định nghĩa constructor nào
}
...
// Dùng constructor mặc định
SinhVien SV = new SinhVien();
```

# Khởi tạo

```
class SinhVien
{
    ...
    // không có constructor mặc định
    SinhVien(<các đối số>) {...}

}
...
SinhVien SV = new SinhVien();
// lỗi biên dịch
```

```
class SinhVien
{
    ...
    // khai báo constructor mặc định
    SinhVien(){}
    SinhVien(<các đối số>) {...}
}
...
SinhVien SV = new SinhVien();
```

# Overloading method

- **Overloading method:** Khai báo trong một lớp nhiều phương thức có cùng tên nhưng khác tham số (khác kiểu dữ liệu, khác số lượng tham số) gọi là khai báo chồng phương thức

```
class SinhVien {
        ...
    public void xemThongTinSV() {
        ...
    }
    public void xemThongTinSV(String psMaSV)
    {
        ...
    }
}
```

# Tham hiếu this

- Biến ẩn tồn tại trong tất cả các lớp, this được sử dụng trong khi chạy và tham khảo đến bản thân lớp chứa nó

```
class SinhVien{
        String  MaSV, TenSV, DCLienlac;
        int     NamSinh;
        …
        public void xemThongTinSV() {
                System.out.println(this.MaSV);
                System.out.println(this.TenSV);
        …
        }
}
```

# NỘI DUNG

1. Các khái niệm cơ bản
2. Khai báo lớp
3. **Lớp nội**
4. Tính đóng gói

# Phân loại

- Non-static
  - Inner
  - Local method
  - Anonymous
- Static

# Inner class

```
class A {
        class B{
        }
}
//không thể khởi tạo đối tượng B mà không khởi tạo đối tượng A
```

- Các khởi tạo đối tượng B

```
A objectA = new A();
A.B objectB = objectA.new B();
```

# Inner class

```
class ThoiGian {

    public int ngay, thang, nam;

    class Time {

        public int gio, phut, giay;

        public void showTime() {
            System.out.println("Ngày " + ngay + "/" + thang + "/" + nam);
            System.out.println("Time: " + this.gio + ": " + this.phut + ": " + this.giay);
        }
    }
}
```

# Inner class

- Lớp nội có thể truy các thành phần thuộc lớp ngoài và ngược lại (trừ khi lớp nội được khai báo là private)
- Nếu lớp nội được khai báo là lớp private thì không thể tạo ra đối tượng lớp nội từ bên ngoài

```
public static void main(String[] args) {
    ThoiGian tg = new ThoiGian();
    ThoiGian.Time time = tg.new Time();
    tg.ngay = 20;
    tg.thang = 7;
    tg.nam = 1996;
    time.gio = 20;
    time.phut = 22;
    time.giay = 01;
    time.showTime();
}
```

# Local method

```
public class LocalMethodTest {
    static void printLine()
    {
        class MethodInnerClass
        {
            void print()
            {
                System.out.println("Hello World");
            }
        }
        MethodInnerClass inner = new MethodInnerClass();
        inner.print();
    }
    public static void main(String[] args) {
        printLine();
    }
}
```

# Anonymous inner class

```
abstract class AbstractClass
{
    abstract void print();
}
public class AnonymousClassTest {
    public static void main(String[] args) {
        AbstractClass AC = new AbstractClass(){
            void print() {
                System.out.println("Test Anonymous class");
            }
        };
        AC.print();
    }
}
```

# Anonymous inner class là tham số

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
abstract class Message
{
    abstract String getMessage();
}
public class MyClass {
    void displayLine(Message ac)
    {
      System.out.println(ac.getMessage());
    }
    public static void main(String[] args) {
        MyClass mc = new MyClass();
        mc.displayLine(new Message(){
            String getMessage() {
                return "Hello";
            }
        });
    }
}
```

# Static nested class

```
public class StaticInnerTest {
    static class Inner
    {
        void print()
        {
            System.out.println("Hello");
        }
    }
    public static void main(String[] args) {
        StaticInnerTest.Inner in = new StaticInnerTest.Inner();
        in.print();
    }
}
```

# NỘI DUNG

1. Các khái niệm cơ bản
2. Khai báo lớp
3. Lớp nội
4. **Tính đóng gói**

# Tính đóng gói

- Nhóm những gì có liên quan với nhau vào thành một và có thể sử dụng một cái tên để gọi
- Che dấu một phần hoặc tất cả thông tin, chi tiết cài đặt bên trong với bên ngoài
- Ví dụ:
  - Các phương thức đóng gói các câu lệnh
  - Đối tượng đóng gói dữ liệu và các hành vi/phương thức liên quan
- Đối tượng = Dữ liệu + Hành vi/Phương thức

# Tính đóng gói

- Đóng gói còn là khai báo các lớp thuộc cùng gói trong java

***package <tên gói>;*** // khai báo trước khi khai báo lớp

```
class <Lớp 1> {
        …
}
class <Lớp 2> {
        …
}
…
```

# Tóm tắt bài học

- Chương trình với Java được tổ chức và xây dựng dựa trên nền tảng là các lớp và đối tượng
- Các tính chất quan trọng: trừu tượng, đóng gói và cheo giấu thông tin, kế thừa và đa hình
- Các từ khóa kiểm soát truy cập: public, protected, default (ko có từ chỉ định), private. Cần thiết lập phạm vi các thành phần tích hợp
- 2 thành phần chính của lớp: thuộc tính (dữ liệu đặc trưng) và phương thức (hành vi/hoạt động đặc trưng)
- Lớp Object là lớp cha ngầm định của mọi lớp, do vậy tất cả các lớp trong Java đều có chung 1 gốc trong cây kế thừa
- Một số thao liên quan đến phương thức: overload vs override
- Một lớp có thể chứa khai báo của một lớp khác (lớp lồng)

# Bài tập

1. Xây dựng lớp Point2D biểu diễn điểm trong mặt phẳng 2 chiều và các phương thức khởi tạo, lấy và thiết lập giá trị các thuộc tính, di chuyển, tính khoảng cách giữa 2 điểm, nhập, xuất. Trong hàm main cho phép nhập vào 2 điểm và một vector (để di chuyển) và xuất ra kết quả của việc di chuyển các điểm và khoảng cách giữa 2 điểm.
2. Viết chương trình tạo lớp Time biểu diễn thời gian gồm: giờ, phút, giây và các phương thức: khởi tạo, lấy và thiết lập giá trị các thuộc tính, tăng giây lên 1 đơn vị, so sánh 2 đối tượng thời gian. Trong hàm main, nhập vào 2 thời gian, xuất ra thời gian lớn hơn và gọi hàm tăng thời gian (thứ nhất) mỗi giây.

# Bài tập

3. Khai báo lớp PhanSo có các thuộc tính: tử số, mẫu số (số nguyên) và các phương thức: constructor (0, 1, 2 tham số), nhập, xuất, rút gọn, cộng, trừ, nhân, chia, so sánh (với một phân số khác). Sau đó viết chương trình cho phép nhập vào dãy n phân số, xuất ra:
   - Các phân số vừa nhập
   - Tổng các phân số
   - Xuất danh sách phân số theo thứ tự tăng dần