Lớp và Đối Tượng (T.T)
(Class & Object)

Nội dung

- Trừu tượng hóa dữ liệu
- Lớp và thành phần của lớp
- Một số khái niệm
- Một số kĩ thuật

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Trừu tượng hóa dữ liệu **Data Abstraction**

- Trừu tượng hóa (Abstraction)

  "*A concept or idea not associated with any specific instance*"

- Chỉ thể hiện những thông tin cốt lõi nhất của sự vật/sự việc, lược bỏ hoặc che dấu đi những thông tin quá chi tiết/cụ thể

- Ví dụ: ta dùng kí hiệu + để mô tả phép cộng hai giá trị thuần túy trong toán học mà không cần mô tả chi tiết là hai giá trị đó phải là giá trị gì và phép toán này có thể áp dụng được ở đâu

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Trừu tượng hóa trong lập trình

- Trừu tượng hóa điều khiển (kĩ thuật lập trình cấu trúc): sử dụng các chương trình con (procedure) và các luồng điều khiển (control flow) để giải quyết bài toán
- Ví dụ: if (a == true) b = (1 + 4) * 5;
- Trừu tượng hóa dữ liệu (kĩ thuật lập trình hướng đối tượng): mô hình hóa dữ liệu thành các thuộc tính và phương thức rồi sử dụng kết hợp các thuộc tính và phương thức này để giải quyết vấn đề
- Ví dụ: circle.calculateArea() { return $(circle.radius)^2 * 3.14$; }

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

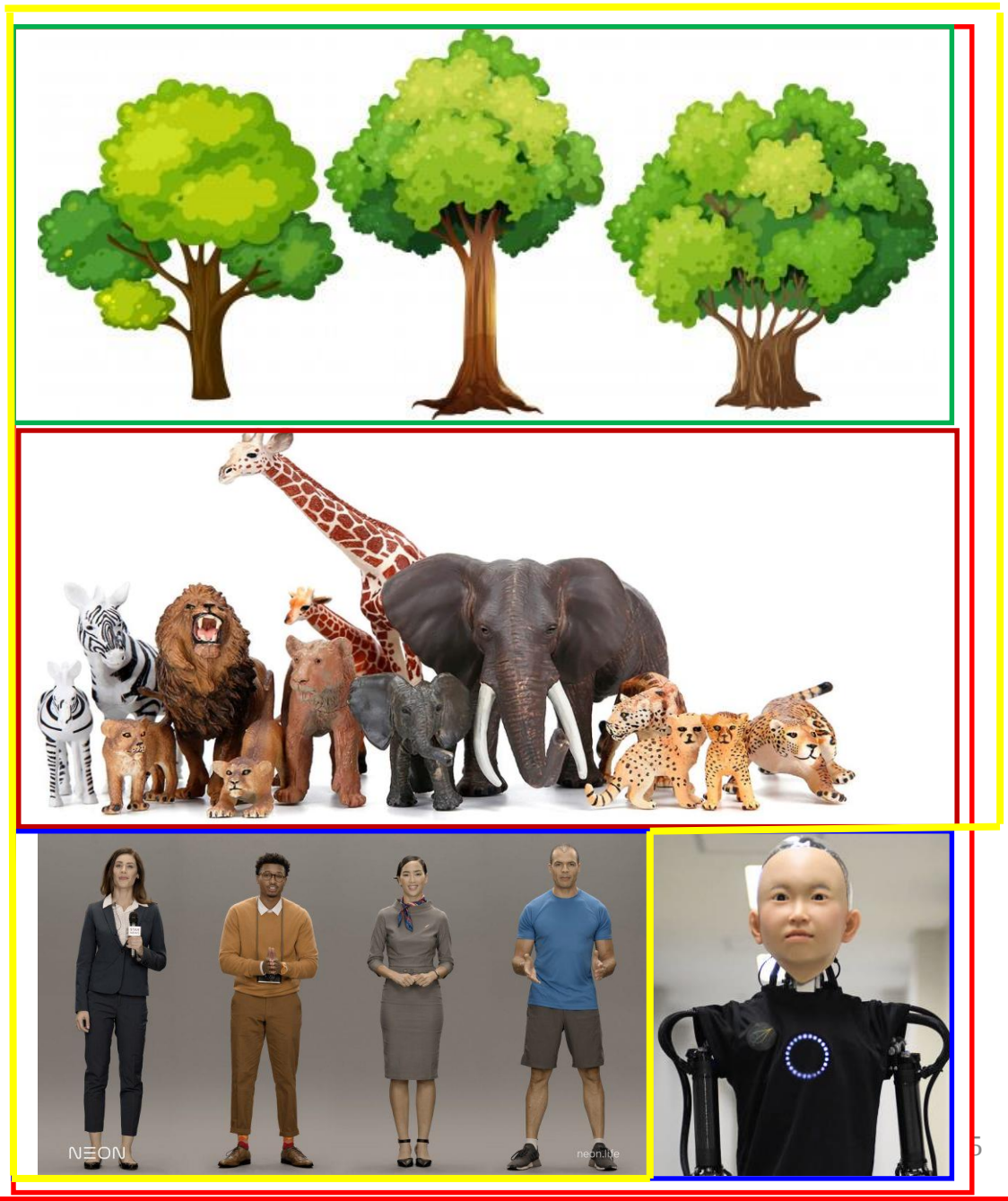
Ví dụ

- Trừu tượng hóa các đối tượng
- Dựa theo các đặc điểm chung để phân loại đối tượng thành các nhóm

Nhóm 1: sinh vật sống

Nhóm 2: thực vật – động vật – con người

Nhóm 3: tự nhiên – nhân tạo



Lớp và thành phần của lớp **Class**

- Lớp (**Class**) là cách phân loại các đối tượng dựa trên đặc điểm chung của chúng
- Lớp có thể được coi là khuôn mẫu để tạo ra các đối tượng

  Ví dụ: Người, động vật, thực vật, phương tiện,…
- Lớp chính là kết quả của quá trình ***trừu tượng hóa dữ liệu***
  - Lớp định nghĩa một ***kiểu dữ liệu*** mới, trừu tượng hóa tập các đối tượng cùng chung thuộc tính
  - Một đối tượng được gọi là một ***thể hiện*** của lớp

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Lớp và thành phần của lớp **Class**

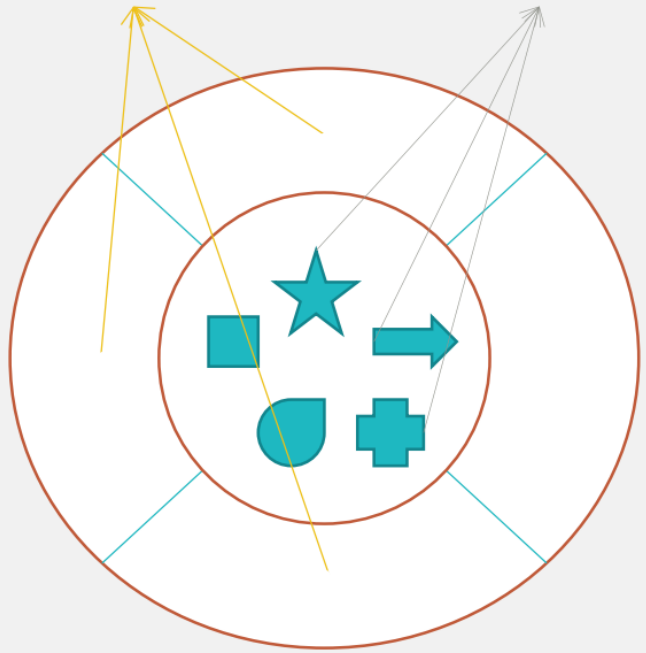
- Lớp đóng gói các ***phương thức*** và ***thuộc tính*** chung của các đối tượng cùng loại

Phương thức: các hành vi mà đối tượng có thể thực hiện được

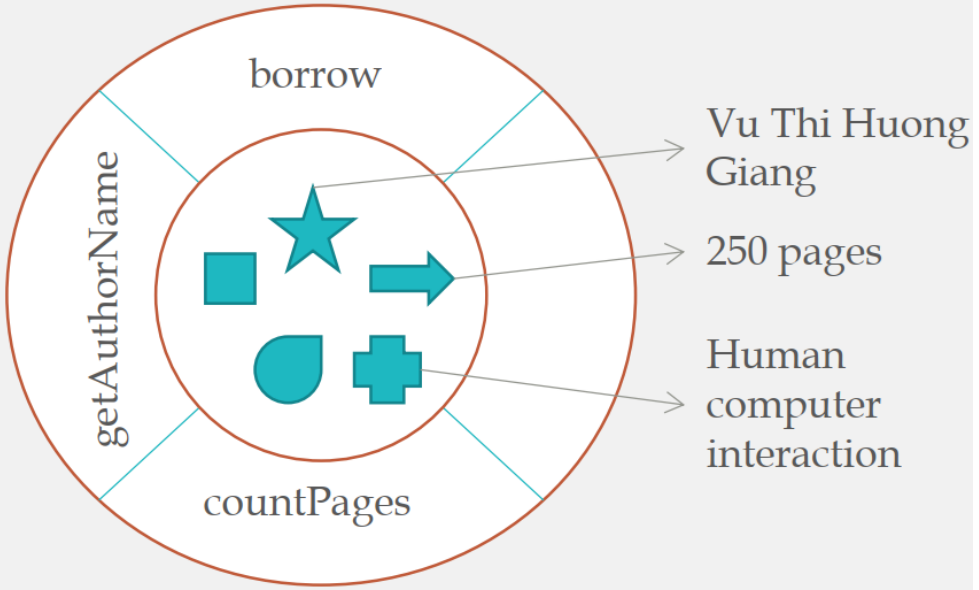
Thuộc tính: các thông tin liên quan đến đối tượng

Thể hiện: một đối tượng cụ thể

Thuộc tính thể hiện: những giá trị gán cho các thuộc tính của một đối tượng cụ thể



Class



Object MyBook

Thuộc tính **Field**

- Một thuộc tính của lớp là một ***trạng thái*** chung mà tất cả mọi đối tượng của lớp đều có

  Ví dụ: mọi đối tượng lớp Con Người đều có các thuộc tính

  + Họ tên
  + Giới tính

- Tuy nhiên, mỗi đối tượng cụ thể lại có các thuộc tính riêng của nó

  Ví dụ: Sinh Viên có thuộc tính riêng

  + MSSV
  + Lớp

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Phương thức **Method**

- Là tập các hoạt động mà các đối tượng (thể hiện) của lớp có thể thực hiện được
- Là cách mà một đối tượng đáp ứng lại một thông điệp bên ngoài tác động lên nó
- Thông thường các phương thức sẽ hoạt động trên các thuộc tính và làm thay đổi trạng thái (giá trị) của các thuộc tính đó
- Mọi phương thức đều phải thuộc về một lớp nào đó

  Ví dụ: lớp Ô tô có các phương thức

  + Khởi động + Tăng tốc

  + Quẹo trái + Quẹo phải

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Phạm vi (tầm vực) **Modifier**

- Là khả năng nhìn thấy/truy cập/sử dụng được các thành phần (thuộc tính, phương thức) của lớp
- Có 4 loại phạm vi
  - private: chỉ nhìn thấy được trong nội bộ của lớp
  - public: có thể nhìn thấy và sử dụng được ở bất kì nơi nào
  - {default}:
    - Là public đối với các lớp thuộc cùng 1 package
    - Là private đối với các lớp nằm trong package khác
  - protected: tương tự như {default} nhưng cho phép kế thừa giữa hai lớp cha – con dù khác package

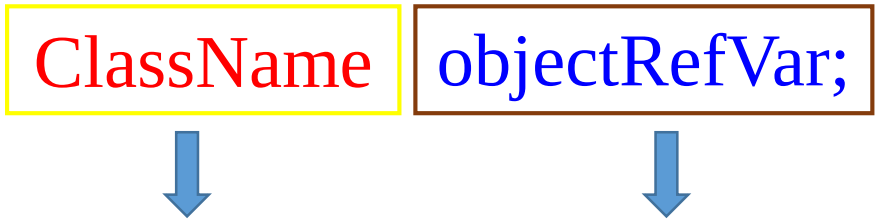
ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Một số khái niệm

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Biến tham chiếu và kiểu tham chiếu

- Trong LTHĐT, mỗi đối tượng (object) được xem là một biến tham chiếu (reference variable)



- Mỗi lớp được xem là một kiểu dữ liệu do người dùng định nghĩa, còn được gọi là kiểu tham chiếu (reference type)

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Ví dụ

- Khi ta khai báo

HinhHoc hinhTron;

- Ta có kiểu dữ liệu tham chiếu là **HinhHoc**
- Ta có biến tham chiếu là hinhTron

- Khi ta khởi tạo

HinhHoc hinhTron = new **HinhHoc**();

- Biến hinhTron sẽ tham chiếu đến một đối tượng **HinhHoc** cụ thể có vùng nhớ do hệ điều hành cấp phát

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Giá trị mặc định

- Là giá trị được khởi tạo ban đầu cho các biến, tuy nhiên giữa các biến là thuộc tính của lớp và các biến là biến cục bộ trong các hàm hay phương thức có sự khác nhau:
  - Các biến là thuộc tính (field/property) của lớp/đối tượng luôn được khởi tạo giá trị mặc định
  - Các biến cục bộ (local variable) được khai báo trong các hàm/phương thức thì không được khởi tạo giá trị mặc định

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Ví dụ

```
public class SinhVien {
    String hoTen; // giá trị mặc định là null
    int tuoi; // giá trị mặc định là 0
    boolean chinhQuy; // giá trị mặc định là false
    char gioiTinh; // giá trị mặc định là '\u0000'

    public static void main(String[] args) {
        int diemSo;
        String tenLop;
        SinhVien sinhVien = new SinhVien();
        System.out.println("Họ và tên: " + sinhVien.hoTen);
        System.out.println("Giới tính: " + sinhVien.gioiTinh);
        System.out.println("Lớp: " + tenLop);
        System.out.println("Điểm số: " + diemSo);
    }
}
```

Báo lỗi: chưa tạo giá trị mặc định cho biến

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Kiểu nguyên thủy vs kiểu tham chiếu

- Trong Java, các kiểu nguyên thủy (primitive type) là: byte, short, int, long, float, double, boolean, char. Các kiểu dữ liệu này thuần túy chỉ lưu những dữ liệu nằm trong phạm vi biểu diễn của nó
- Kiểu tham chiếu (reference type) là dạng các Class có sẵn trong Java như String, Double, Integer, Boolean,…hay các Class do người lập trình tự định nghĩa như HocSinh, ThietBi, PhuongTien, MonHoc,…

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Kiểu nguyên thủy vs kiểu tham chiếu

- Mỗi một biến (variable) khi được tạo ra đều chiếm một ô nhớ hay một vùng nhớ xác định trong bộ nhớ thi hành (RAM) của máy tính
- Tuy nhiên, biến kiểu nguyên thủy sẽ dùng bộ nhớ của mình để chứa trực tiếp giá trị mà nó biểu diễn.Ví dụ: int i = 1;

- Biến kiểu tham chiếu sẽ dùng bộ nhớ của mình để chứa tham chiếu (địa chỉ) đến ô nhớ hoặc vùng nhớ mà đối tượng được biểu diễn

Ví dụ: HinhTron h = new HinhTron(12);

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

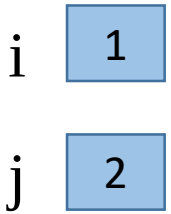
Kiểu nguyên thủy vs kiểu tham chiếu

- Khi thực hiện phép gán

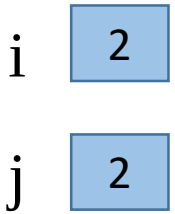
Kiểu nguyên thủy

Phép gán i = j sẽ copy giá trị của j và đưa vào i

Trước khi gán:



Sau khi gán:

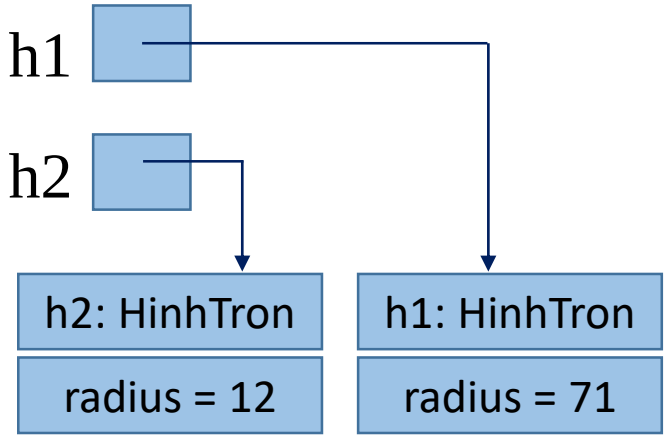


Kiểu tham chiếu

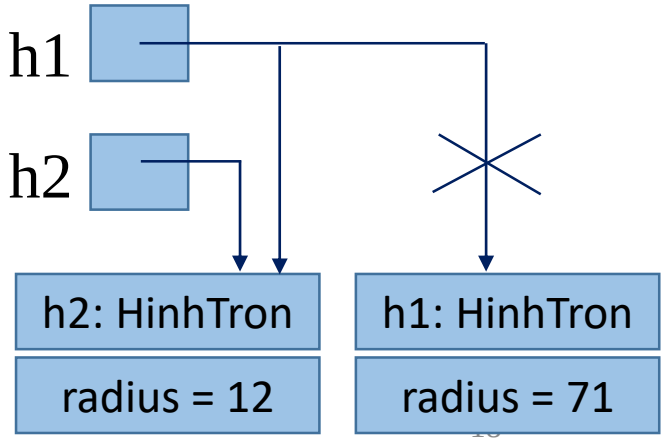
Phép gán h1 = h2 sẽ copy tham chiếu của h2 đưa vào h1

➔ biến h1 và h2 thành 1 đối tượng trùng nhau

Trước khi gán:



Sau khi gán:



Từ khóa **this**

- Đại diện cho đối tượng sẽ được biểu diễn của Class
- Sử dụng để tham chiếu đến các thành viên của Class một cách tường minh (explicit), tránh bị nhầm lẫn với các biến khác. VD: tránh bị nhầm lẫn với tham số hay biến cục bộ trùng tên của hàm

```
public class Circle{
    private double banKinh;

    // Phương thức khởi tạo
    Circle(double banKinh) {
        this.banKinh = banKinh;
    }
}
```

Trường

Tham số

```
public class Circle{
    private double banKinh;

    // Phương thức khởi tạo
    Circle() {
        double banKinh = 16.8;
        this.banKinh = banKinh;
    }
}
```

Trường

Biến cục bộ

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Gói **package**

- Trong Java, gói (package) giống như một cấu trúc thư mục giúp
  - Nhóm các Class, Interface, Sub Package có liên quan hay có cùng chung mục đích lại với nhau
  - Giúp ngăn cản xung đột khi đặt tên (các package khác nhau có thể chứa các lớp cùng tên với nhau), điều khiển truy cập và giúp thuận tiện trong việc tìm kiếm và lưu trữ
- Một package có thể chứa nhiều package con
- Còn được gọi là ***không gian tên*** hay ***tên miền*** (namespace) tương tự như các ngôn ngữ lập trình khác (C/C++, C#, VB,…)

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Gói **package**

- Java đã xây dựng sẵn một số package
  - java.lang
  - java.io
  - java.util
  - javax.swing
  - …
- Có thể tự tạo ra các gói để tổ chức các lớp
  - Cú pháp: package <tên gói>;

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Một số kĩ thuật

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Nạp chồng phương thức **Method Overloading**

- Trong một lớp ta có thể định nghĩa nhiều phương thức có cùng tên và kiểu dữ liệu trả về giống nhau, nhưng các tham số của các phương thức này phải khác nhau (*Khác kiểu, khác số lượng, thứ tự tham số*)

- Ví dụ

```
public class MyClass{
    void method(){...}
    void method(int x){...}
    void method(float x){...}
    void method(int x, double y){...}
}
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Nạp chồng hàm/phương thức

```
class MayTinh{
    int tong(int a, int b){return a + b;}
    int tong(int a, int b, int c){return a + b + c;}
}
```

```
MayTinh mt = new MayTinh();
int t1 = mt.tong(5, 7);
int t2 = mt.tong(5, 7, 9);
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Tính đóng gói **Encapsulation**

- Encapsulation: tính đóng gói là đặc tính cho phép che giấu thông tin trong lập trình hướng đối tượng

- Việc che giấu giúp bảo vệ dữ liệu thành viên, tránh những can thiệp không mong muốn từ bên ngoài Class, đồng thời tăng cường khả năng độc lập dữ liệu giữa đối tượng của lớp với môi trường bên ngoài

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Tính đóng gói **Encapsulation**

- Ví dụ: nếu không che giấu thông tin thì sao ?!

```
public class SinhVien{
        public String hoTen;
        public double diem;
}
```

```
public class MyClass{
        public static void main(String[] args){
                SinhVien sv = new SinhVien();
                sv.hoTen = "Nguyễn Văn Tèo";
                sv.diem = 20.5;
        }
}
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Tính đóng gói **Encapsulation**

- Để che giấu thông tin, nên sử dụng từ khóa private cho các field là dữ liệu thành viên của lớp

  Ví dụ: private String hoTen;

  private double diem;

- Đã che giấu rồi thì làm sao truy cập từ bên ngoài để đọc hoặc gán dữ liệu ?!. Nên bổ sung các phương thức getter, setter để cho phép truy xuất tới dữ liệu đã bị che giấu trong các trường hợp đọc, ghi dữ liệu cho đối tượng của lớp

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Tính đóng gói **Encapsulation**

- Ví dụ về phương thức getter và setter

```
public String getHoTen(){
    return  this.hoTen;
}
public void setHoTen(String s){
    this.hoTen = s;
}
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Tính đóng gói **Encapsulation**

```
public class SinhVien{
        private String hoTen;
        private double diem;
        public void setHoTen(String hoTen){
                this.hoTen = hoTen;
        }
        public String getHoTen(){
                return this.hoTen;
        }
        public void setDiem(double diem){
                if(diem < 0 || > 10){
                        System.out.println("Điểm không hợp lệ");
                }
                else{
                        this.diem = diem;
                }
        }
        public String getDiem(){
                return this.diem;
        }
}
```

```
public class MyClass{
        public static void main(String[] args){
                SinhVien sv = new SinhVien();
                sv.setHoTen("Nguyễn Văn Tèo");
                sv.setDiem(20);
        }
}
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Tóm lại

- Một số ưu điểm của **OOP**
  - Thiết kế chương trình theo hướng trừu tượng – **Abstraction**
  - Đóng gói và che giấu thông tin – **Encapsulation**
  - Có phạm vi truy xuất - **Modifier**
  - Có kĩ thuật nạp chồng phương thức/hàm - **Overloading**
  - Năng cao khả năng tái sử dụng mã nguồn – **Reusable**

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

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

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn