Quan Hệ Kế Thừa (Inheritance)

Nội dung

- Giới thiệu
- Cú pháp
- Ý nghĩa
- Phân loại
- Một số từ khóa

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

Giới thiệu: **Kế Thừa**

- Thừa kế là một loại quan hệ giữa 2 đối tượng: Cha - Con

  Ví dụ: người con được quyền thừa kế các loại tài sản do cha mẹ để lại

- Trong lập trình hướng đối tượng

"***Object-Oriented programming allows you to define new classes from existing classes. This is called inheritance***"

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

Giới thiệu: **Kế Thừa**

- Kế thừa trong lập trình hướng đối tượng (OOP)
  - Lớp Con được gọi là SubClass
  - Lớp Cha được gọi là SuperClass
  - Lớp Con được xem là một loại (**is-a-kind-of**) lớp của lớp Cha, hay nói cách khác giữa SubClass và SuperClass có mối quan hệ "**is-a**"
- Ví dụ: HinhTron là con của lớp HinhHoc

  SinhVien là con của lớp Nguoi

  HocSinh là con của lớp SinhVien **??**

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

Giới thiệu: **Kế Thừa**

- Đặc điểm của kế thừa trong OOP
  - Lớp Con được phép thừa kế lại những **thuộc tính** và **phương thức** của lớp Cha
  - Lớp Con được phép:
    - ✓Thêm vào các **thuộc tính** và **phương thức** riêng của nó
    - ✓Hiệu chỉnh các **phương thức** được thừa kế từ lớp Cha theo mục đích riêng của lớp Con

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

Vì sao phải có sự thừa kế ?

- Tồn tại nhiều loại đối tượng có các thuộc tính và hành vi tương tự hoặc liên quan đến nhau

  Ví dụ: Person, Student, Manager, Teacher,…

```
public class Person {
     private String name;
     private int age;
     private String sex;

     public Person(String name, int age, String sex) {
          this.name = name;
          this.age = age;
          this.sex = sex;
     }

     public void getInfo() {
          System.out.println("Name: " + this.name);
          System.out.println("Age: " + this.age);
          System.out.println("Sex: " + this.sex);
     }
}
```

```
public class Teacher {
     private String name;
     private int age;
     private String sex;

     public Teacher (String name, int age, String sex) {
          this.name = name;
          this.age = age;
          this.sex = sex;
     }

     public void getInfo() {
          System.out.println("Name: " + this.name);
          System.out.println("Age: " + this.age);
          System.out.println("Sex: " + this.sex);
     }
}
```


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

Tính tái sử dụng: **Reuse**

- Tồn tại nhiều loại đối tượng có các thuộc tính và hành vi tương tự hoặc liên quan đến nhau

  Ví dụ: Person, Student, Manager, Teacher,…

```
public class Person {
     private String name;
     private int age;
     private String sex;

     public Person(String name, int age, String sex) {
          this.name = name;
          this.age = age;
          this.sex = sex;
     }

     public void getInfo() {
          System.out.println("Name: " + this.name);
          System.out.println("Age: " + this.age);
          System.out.println("Sex: " + this.sex);
     }
}
```

```
public class Manager {
     private String name;
     private int age;
     private String sex;

     public Manager (String name, int age, String sex) {
          this.name = name;
          this.age = age;
          this.sex = sex;
     }

     public void getInfo() {
          System.out.println("Name: " + this.name);
          System.out.println("Age: " + this.age);
          System.out.println("Sex: " + this.sex);
     }
}
```


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

Tính tái sử dụng: **Reuse**

- Tồn tại nhiều loại đối tượng có các thuộc tính và hành vi tương tự hoặc liên quan đến nhau

  Ví dụ: Person, Student, Manager, Teacher,…

```
public class Person {
     private String name;
     private int age;
     private String sex;

     public Person(String name, int age, String sex) {
          this.name = name;
          this.age = age;
          this.sex = sex;
     }

     public void getInfo() {
          System.out.println("Name: " + this.name);
          System.out.println("Age: " + this.age);
          System.out.println("Sex: " + this.sex);
     }
}
```

```
public class Student {
     private String name;
     private int age;
     private String sex;

     public Student(String name, int age, String sex) {
          this.name = name;
          this.age = age;
          this.sex = sex;
     }

     public void getInfo() {
          System.out.println("Name: " + this.name);
          System.out.println("Age: " + this.age);
          System.out.println("Sex: " + this.sex);
     }
}
```


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

Tính tái sử dụng: **Reuse**

- Xuất hiện nhu cầu tái sử dụng lại các mã nguồn, chương trình hay thư viện đã viết
  - Tái sử dụng thông qua hình thức copy
  - Tái sử dụng thông qua quan hệ "has-a" (Composition)
  - Tái sử dụng thông qua quan hệ "is-a" (Inheritance)

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

Tính tái sử dụng: **Reuse**

- Việc copy lại mã nguồn
  - Tốn công, dễ nhầm lẫn, tốn kém dung lượng lưu trữ
  - Khó sửa lỗi do tồn tại nhiều phiên bản
  - Mang tính rập khuôn

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

Tính tái sử dụng: **Reuse**

- Sử dụng quan hệ "has-a" (Composition)
  - Sử dụng lớp cũ như là thành phần của lớp mới
  - Sử dụng lại các cài đặt với giao diện mới
    - Phải viết lại giao diện
    - Cứng nhắc, Thiếu linh hoạt, đôi khi không đáp ứng được yêu cầu truy xuất

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

Tính tái sử dụng: **Reuse**

- Ví dụ về quan hệ "has-a" (Composition)

```
class Person {
    private String name;
    private Date birthday;
    public String getName() { return name; }
    ...
}
class Employee {
    private Person me;
    private double salary;
    public String getName() { return me.getName(); }
    ...
}
```


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

Tính tái sử dụng: **Reuse**

- Ví dụ về quan hệ "has-a" (Composition)

```
class Manager {
    private Employee me;
    private Employee assistant;
    public void setAssistant(Employee e) {...}
    ...
}
...
Manager junior = new Manager();
Manager senior = new Manager();
senior.setAssistant(junior);    //--- Lỗi vì junior không phải là Employee
```


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

Tính tái sử dụng: **Reuse**

- Sử dụng quan hệ "is-a" (Inheritance)
  - Tránh được những khuyết điểm của quan hệ "has-a"
  - Tránh được sự lãng phí tài nguyên
  - Việc thừa kế còn cho phép tạo ra những "tài sản" riêng hoặc điều chỉnh lại những "tài sản" được thừa kế cho phù hợp với đối tượng và mục đích sử dụng ➔ tính linh hoạt cao

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

Ý Nghĩa: **Kế Thừa**

- Ý nghĩa của việc thừa kế
  - Tạo ra hiệu quả trong lập trình
  - Giúp ta có khả năng tái sử dụng code (không phải đi định nghĩa lại các lớp đã được xây dựng)
  - Tiết kiệm thời gian và công sức
  - Thuận tiện cho việc bảo trì
  - Thuận tiện cho việc nâng cấp và phát triển chương trình

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

Cú pháp: **Kế Thừa**

- Để tạo ra một lớp Con kế thừa từ lớp Cha trong Java, ta có cú pháp như sau:

```
[modifier] class <SubClass> extends <SuperClass> {
        // Thành phần dữ liệu
        …
        // Thành phần xử lý
        …
}
```


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

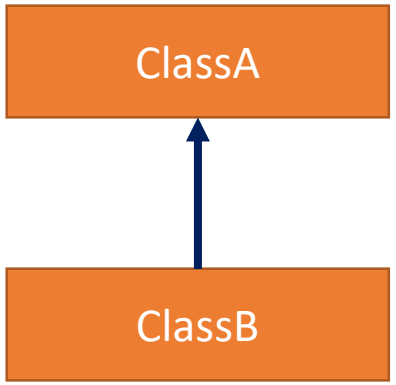
Cú pháp: **Kế Thừa**

- Ví dụ

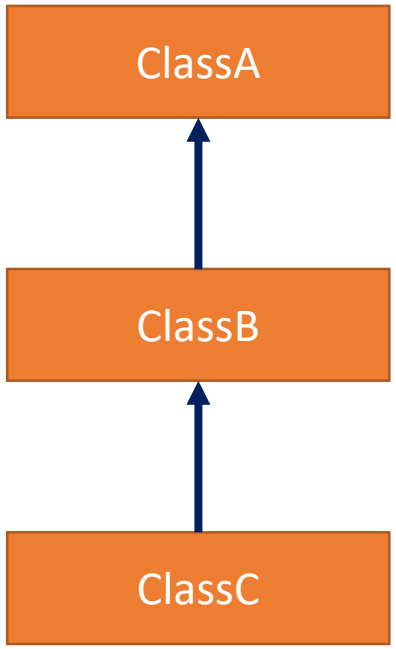
```
class HinhTron extends HinhHoc{
…
}
class XeMay extends PhuongTienGiaoThong{
…
}
class GiangVien extends NhanVien{
…
}
```


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

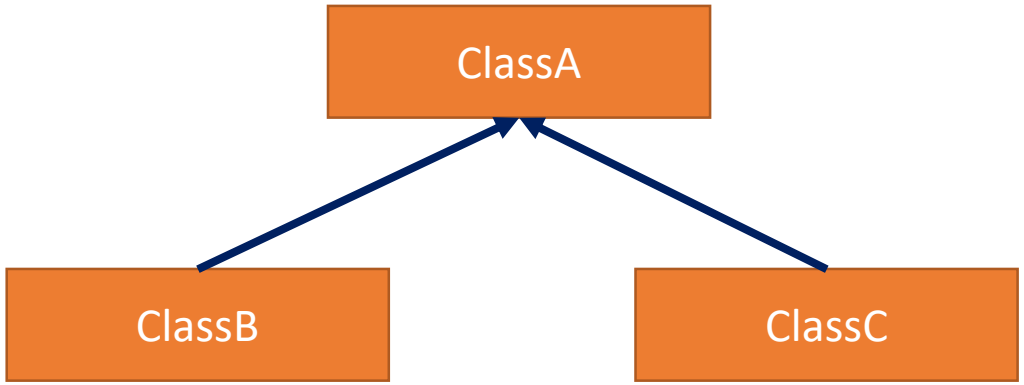
Phân loại **Thừa kế**



Single Inheritance
(Thừa kế đơn tầng/
đơn thừa kế)



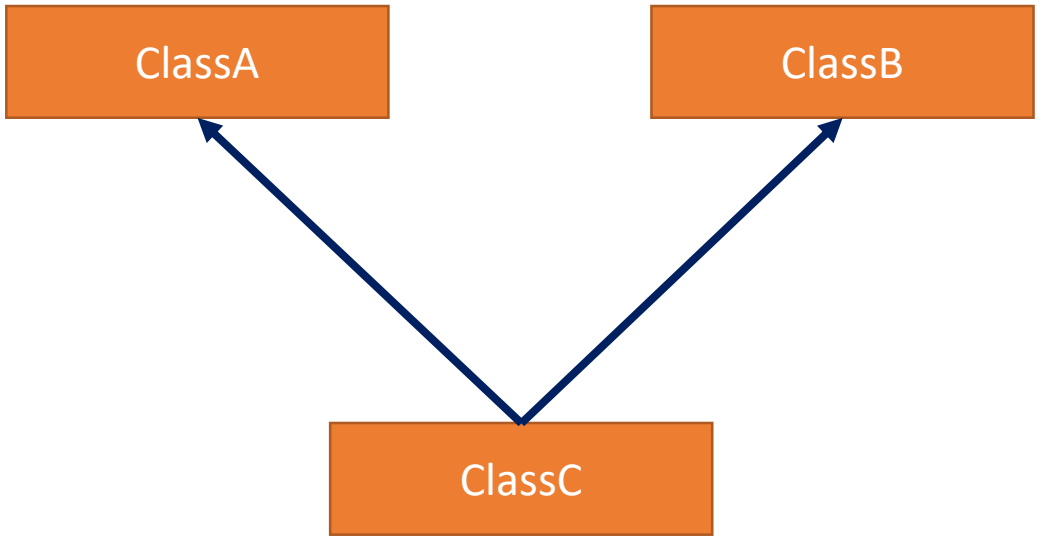
Multilevel Inheritance
(Thừa kế đa tầng)



Hierarchical Inheritance
(Thừa kế có thứ bậc)

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

Phân loại **Thừa kế**



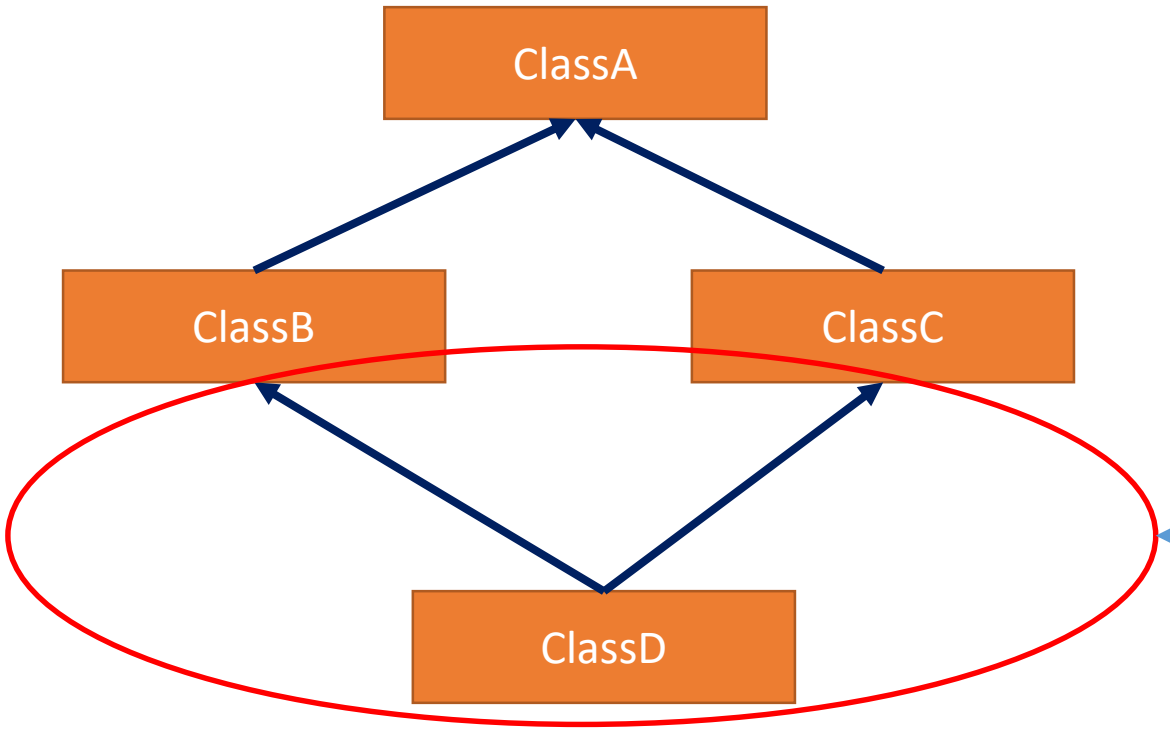
Multiple Inheritance
(Đa thừa kế)

Không tồn tại trong Java

```
public class Test {
     class A{
          void outputMessage() {
                    System.out.println("Hello");
          }
     }

     class B{
          void outputMessage() {
                    System.out.println("Welcome");
          }
     }

     class C extends A, B {
     // giả sử tồn tại kiểu thừa kế này
     }

     public void main(String[] args) {
          C classC = new C();

          // Phương thức outputMessage() nào sẽ được trả về ?
          classC.outputMessage();
     }
}
```


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

Phân loại **Thừa kế**



Vì multiple Inheritance không được hỗ trợ trong Java

Hybrid Inheritance
(Lai thừa kế)

Không tồn tại trong Java

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

Một số đặc điểm cần **lưu ý**

- Một lớp Con không phải là một tập hợp con (subset) của lớp Cha mà lớp Con là một mở rộng (**extends**) của lớp Cha

  Vì: nó có thể chứa nhiều thuộc tính và phương thức hơn lớp Cha

- Các thuộc tính **private** của lớp Cha không được thừa kế cho lớp Con

- Trong Java, một lớp Con chỉ được thừa kế từ duy nhất một lớp Cha mà thôi (*single inheritance*), nếu muốn lớp Con có thể thừa kế được nhiều thuộc tính và phương thức từ các lớp Cha khác nhau, chúng ta phải sử dụng qua khái niệm **Interface**

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

Một số đặc điểm cần **lưu ý**

- Lớp con được "thừa hưởng" các tài sản từ lớp cha (*field*, *method*), tuy nhiên việc "thừa hưởng" này phải tuân theo một số đặc điểm như sau:
  - Các thành phần **public** và **protected** của lớp Cha trở thành **public** và **protected** của lớp Con
  - Các thành phần **{default}**của lớp Cha sẽ trở thành **{default}** của lớp Con nếu cả Cha và Con đều thuộc cùng 1 package
  - Các thành phần **private** của lớp Cha sẽ không bao giờ được thừa kế
  - Phương thức **constructor** không cho phép thừa kế

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

Package & Access Control

	private	public	protected	default
Cùng lớp	**YES**	**YES**	**YES**	**YES**
Cùng package Lớp con	**NO**	**YES**	**YES**	**YES**
Cùng package Không phải lớp con	**NO**	**YES**	**YES**	**YES**
Khác package Lớp con	**NO**	**YES**	**YES**	**NO**
Khác package Không phải lớp con	**NO**	**YES**	**NO**	**NO**

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

Ví dụ

SubClass

SuperClass

```
class HinhHoc {
    private String color;
    private boolean filled; // tô màu
    private Date dateCreated; // ngày tạo
    public HinhHoc(boolean filled) {
        dateCreated = new java.util.Date();
        this.color = "brown";
        this.filled = filled;
    }
   // Thay đổi màu của đối tượng
   public void setColor(String color) {
        this.color = color;
   }
   …
}
```

```
class HinhTron extends HinhHoc{
    private double radius;

    public HinhTron() {
        setColor("blue");
        this.radius = 1.0;
    }

    public HinhTron(String color, double radius) {
        this.color = color;
        this.radius = radius;
    }
    …
}
```


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

Ví dụ

SubClass

SuperClass

```
class HinhHoc {
    public String color;
    private boolean filled; // tô màu
    private Date dateCreated; // ngày tạo
    public HinhHoc(boolean filled) {
        dateCreated = new java.util.Date();
        this.color = "brown";
        this.filled = filled;
    }
   // Thay đổi màu của đối tượng
   public void setColor(String color) {
        this.color = color;
   }
   …
}
```

```
class HinhTron extends HinhHoc{
    private double radius;

    public HinhTron() {
        setColor("blue");
        this.radius = 1.0;
    }

    public HinhTron(String color, double radius) {
        this.color = color;
        this.radius = radius,
    }
    …
}
```


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

Từ khóa **super**

- Là từ khóa đại diện cho lớp Cha
- Giống như từ khóa **this**, tuy nhiên **super** dùng để truy xuất đến phương thức (method) và phương thức khởi tạo (constructor) của lớp Cha
- Khi thực hiện phương thức khởi tạo ở một lớp Con thì phương thức khởi tạo của lớp Cha sẽ luôn được gọi để thực hiện trước (khởi tạo Cha rồi mới đến khởi tạo Con)

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

Từ khóa **super**

```
public ClassName() {
  // some statements
}
```

Equivalent

```
public ClassName() {
  super();
  // some statements
}
```

```
public ClassName(double d) {
  // some statements
}
```

Equivalent

```
public ClassName(double d) {
  super();
  // some statements
}
```


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

Ví dụ

```
class HinhHoc {
    private String color;
    private boolean filled; // tô màu
    private Date dateCreated; // ngày tạo
    public HinhHoc(boolean filled) {
        dateCreated = new java.util.Date();
        this.color = "brown";
        this.filled = filled;
    }
   // Thay đổi màu của đối tượng
   public void setColor(String color) {
        this.color = color;
   }
   …
}
```

```
class HinhTron extends HinhHoc{
    private double radius;

    public HinhTron() {
        super(true);
        this.radius = 1.0;
    }

    public HinhTron(double radius) {
        this.radius = radius;
    }
    …
}
```


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