

BÀI 5

LẬP TRÌNH KẾ THỪA TRONG JAVA

TÌNH HUỐNG DẪN NHẬP

Bài toán: Xây dựng hệ thống quản lý nhân sự và sinh viên của trung tâm tin học PT.

Giám đốc Trung tâm tin học PT muốn xây dựng hệ thống phần mềm để tiện quản lý sinh viên và nhân sự trong Trung tâm. Có 2 yếu tố liên quan đến nhân sự cần quản lý:

- Nhân Viên (có tên, ngày sinh, giới tính, ngày vào làm, điểm thi).
- Sinh Viên (có tên, ngày sinh, giới tính, ngày nhập học, tiền lương).

Vì sinh Viên và nhân Viên thuộc vào vị trí quản lý khác nhau mà lại có những thông số quản lý giống nhau nhưng không thể gộp chung để quản lý được. Do đó nếu có sự thay đổi về thông số trong cách quản lý (giả sử thay đổi về cách quản lý ngày sinh) phải sửa lại thông tin cho cả nhân viên và sinh viên.


Nên một yêu cầu khác là làm sao để hệ thống có thể nhóm những thông tin giống nhau của Sinh Viên và Nhân Viên để tiện quản lý.



Vậy theo Anh Chị làm thế nào để hệ thống có thể tránh sự lặp lại trong cách quản lý thông tin của nhân viên và sinh viên?

MỤC TIÊU

Trình bày một số khái niệm cơ bản về tính kế thừa trong Java (Supper Class, Sub-Class, Overriding).



Xây dựng chương trình Java đơn giản sử dụng tính kế thừa.



NỘI DUNG

1

Tính kế thừa trong lập trình hướng đối tượng.

2

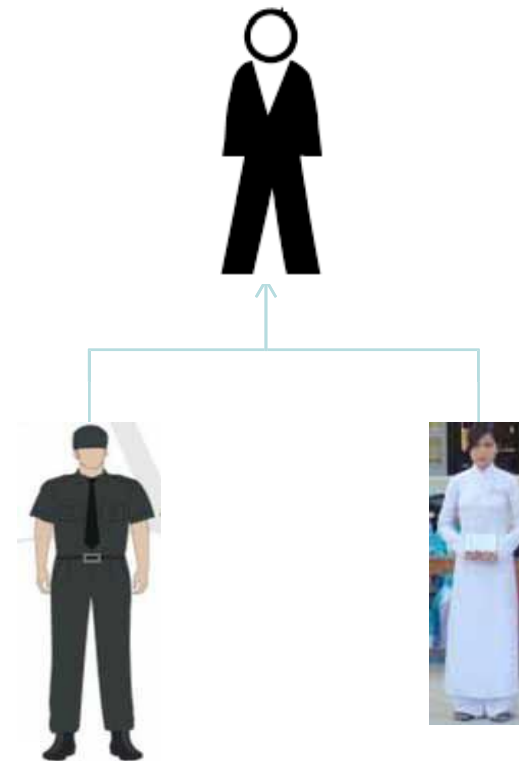
Lập trình kế thừa trong Java.

3

Vấn đề Upcasting – Downcasting.

1. TÍNH KẾ THỪA

- Kế thừa là là khái niệm dùng để chỉ quá trình tạo ra các lớp mới dựa trên những đặc điểm và phương thức của một lớp đã có.
- Kế thừa giúp tránh việc phải xây dựng lại các đặc tính đã có của một lớp, tạo ra khả năng mở rộng và nâng cấp một cách đơn giản, hiệu quả.



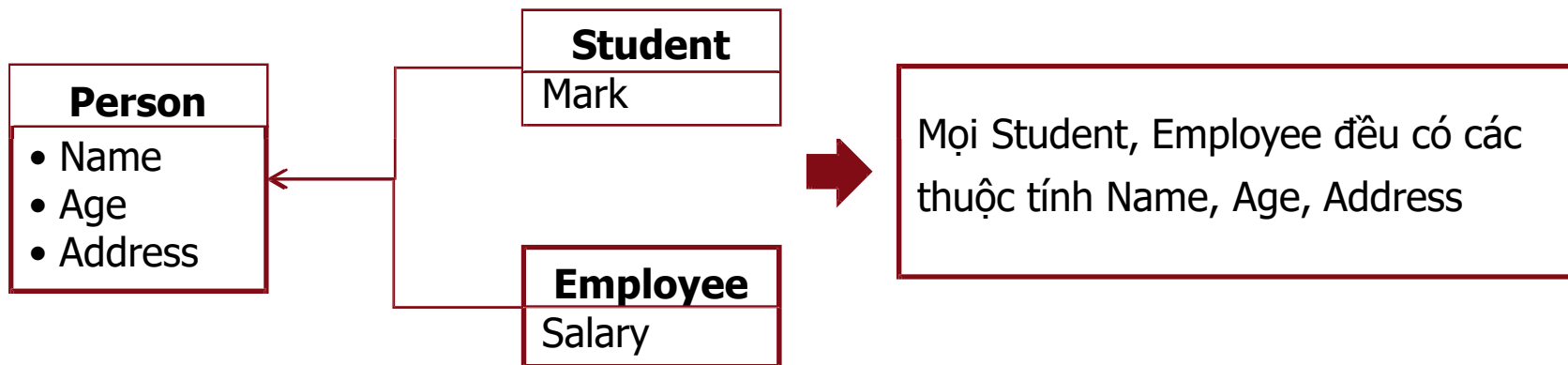
Đối tượng sinh viên và nhân viên kế thừa từ đối tượng con người.

2. LẬP TRÌNH KẾ THỪA TRONG JAVA

1. Supper Class và Sub-Class.
2. Xây dựng kế thừa với ngôn ngữ Java.
3. Overriding, Overloading.
4. Từ khóa super.

2.1. SUPER CLASS VÀ SUB-CLASS

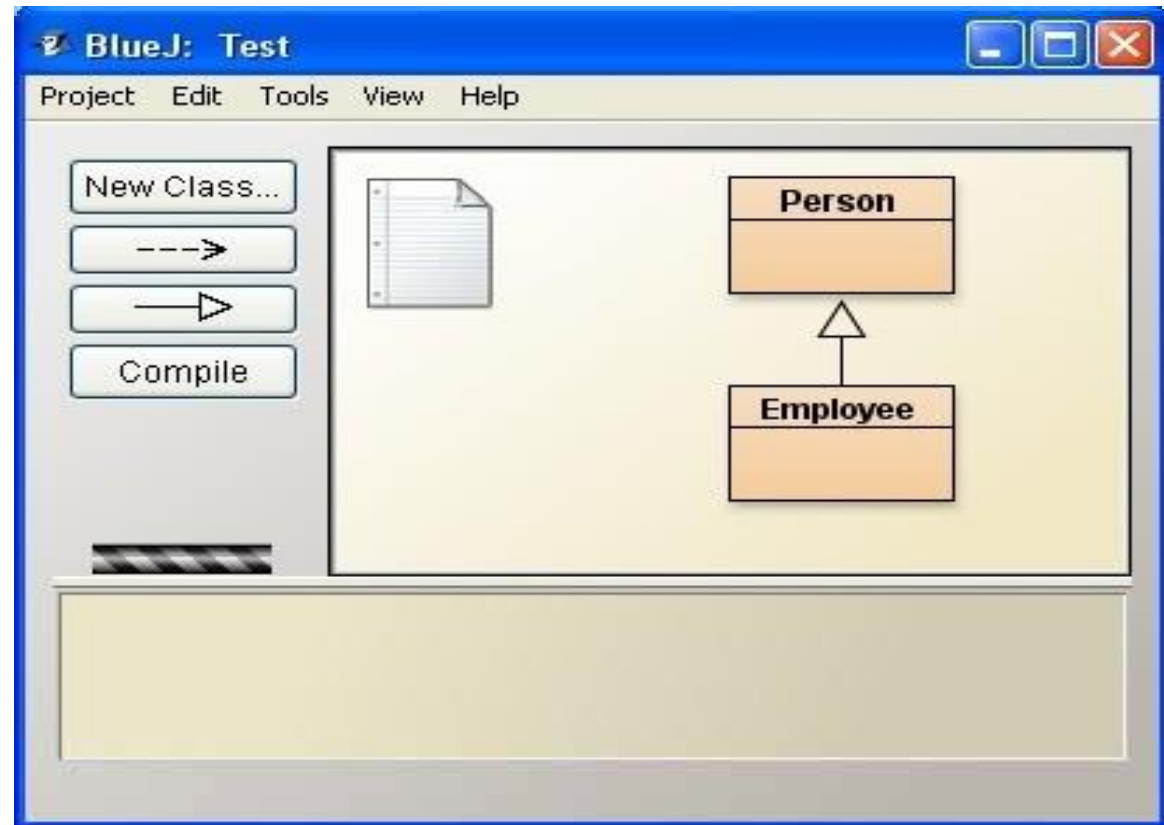
- Super Class là thuật ngữ dùng để chỉ các lớp cho phép các lớp khác kế thừa. Có thể gọi là lớp Cơ sở hay lớp Cha. (lớp **Person**).
- Sub-class là thuật ngữ dùng để chỉ các lớp kế thừa. Có thể gọi là lớp Dẫn xuất hay lớp Con. (lớp **Student** và **Employee**).
- Lớp kế thừa sẽ được **thừa hưởng các đặc điểm từ lớp cơ sở** bao gồm các **phương thức** và các **trường nếu** được phép.
- Khi một lớp A kế thừa từ một lớp B, ta có mối quan hệ **A là B**:
 - Trong bài toán tình huống ta phân tích lớp Student và Employee có chung lớp Person. Vậy Student kế thừa từ lớp Person → Student là Person.
 - Như vậy mọi đặc điểm mô tả trong lớp Person đều có thể xuất hiện trong lớp Student.



2.2. XÂY DỰNG KẾ THỪA TRONG JAVA

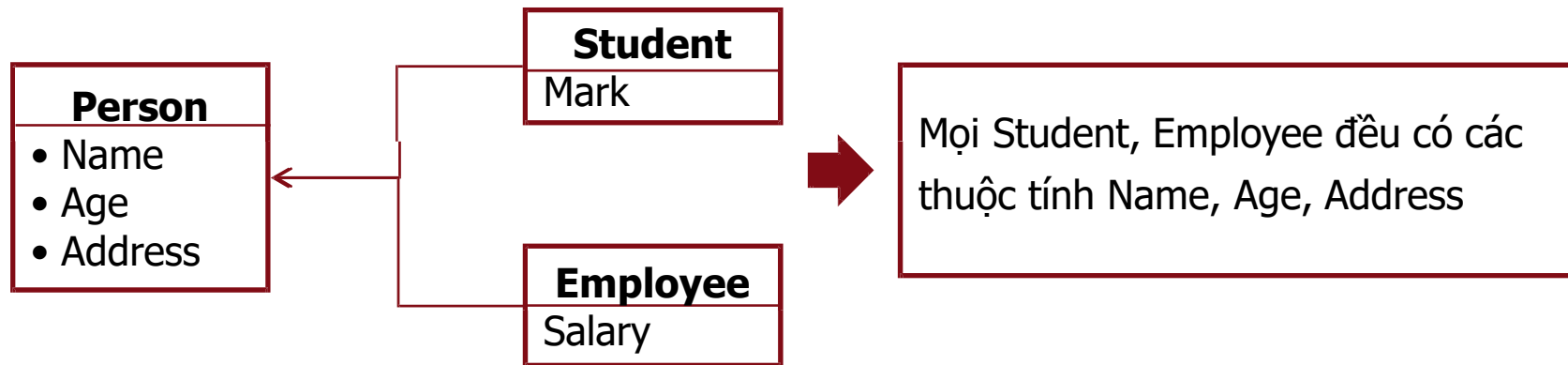
- Trong Java, mặc định tất cả các lớp đều được kế thừa từ một lớp cơ sở là lớp `java.lang.Object`.
- Java cho phép một lớp chỉ được kế thừa từ một lớp cơ sở.
- Hàm tạo là thành phần không được kế thừa.
- Trong bài toán tình huống xây dựng lớp Person, lớp Employee kế thừa lớp Person:

```
public class Employee extends
Person
{
}
}
```



TRUY XUẤT CÁC THÀNH PHẦN

Mặc dù lớp con chứa tất cả các thành phần của lớp cha, nhưng nó không thể truy xuất các thành phần được khai báo là **private** trong lớp cha.

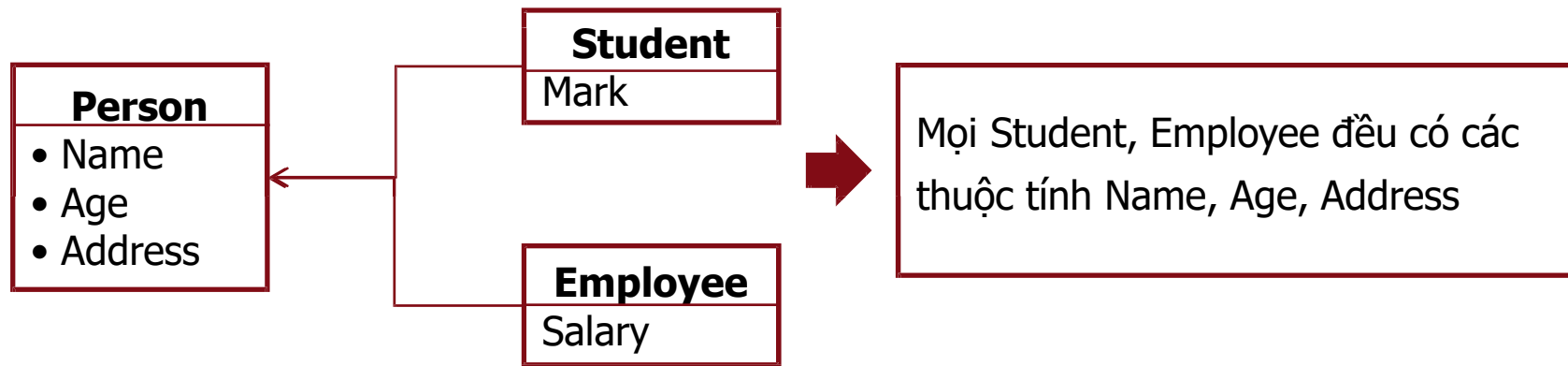


Giả sử các thuộc tính **name, age, address** được khai báo là **private** trong lớp Person. Thì trong lớp Student, không thể truy xuất trực tiếp vào các thuộc tính này. Để truy xuất được, ta xây dựng cặp hàm **set/get** (như trong session1).

```
Student st01 = new Student();
st01.name = "Nguyen Van A"; //đoạn code này sẽ sinh lỗi
```

BIẾN CỦA LỚP CHA CÓ THỂ THAM CHIẾU ĐẾN MỘT LỚP CON

Một biến tham chiếu của lớp cha có thể gán để tham chiếu đến một lớp con bất kỳ dẫn xuất từ lớp cha.



```
Person peter = new Employee();  
peter.mark = 20.5; //lỗi
```

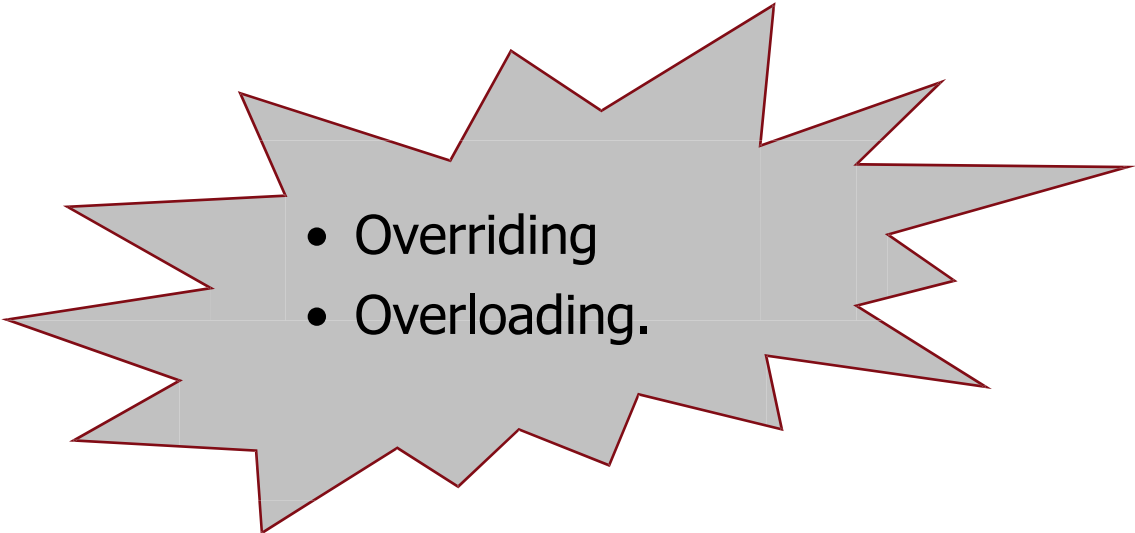
peter là một tham chiếu đến đối tượng Employee, do đó chỉ có quyền truy xuất những thành phần được định nghĩa bởi lớp cha (Person).

CÂU HỎI TƯƠNG TÁC



Sự khác nhau giữa lớp cha và lớp con?

2.3. OVERRIDING, OVERLOADING

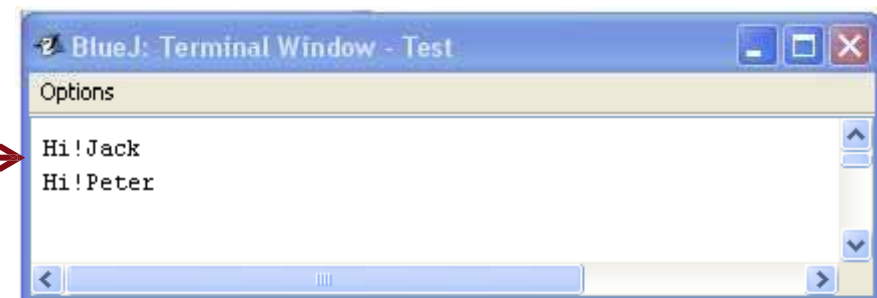
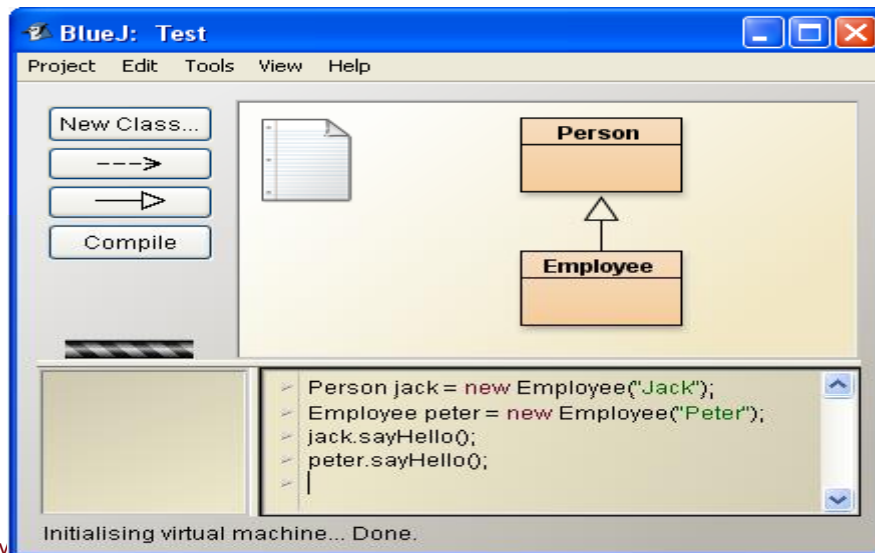
- 
- Overriding
 - Overloading.

OVERRIDING

```
public class Person {  
    protected String name;  
    /**  
     * Constructor for objects of  
     class Person  
     */  
    public Person(String name) {  
        // To do:  
        this.name=name;  
    }  
    public void sayHello() {  
        System.out.println("Hello!" + name);  
    }  
}
```

```
public class Employee extends Person  
{  
    /**  
     * Constructor for objects of  
     class Employee  
     */  
    public Employee(String name) {  
        // To do:  
        super(name);  
    }  
    //overriding  
    public void sayHello() {  
        System.out.println("Hi!" + name);  
    }  
}
```

Cài đặt lại phương thức
sayHello() của lớp Person



Kết quả chạy trên BlueJ

OVERLOADING

- Overloading là cơ chế cho phép trong một lớp có thể có nhiều hơn một phương thức có cùng tên, kiểu dữ liệu trả về nhưng khác nhau về số lượng tham số truyền vào;
- Constructor cũng là phương thức, do đó constructor cũng được overloaded.

```
public class Person {  
    protected String name;  
    public Person(){ //no body }  
    /**  
     * Constructor for objects of class  
    Person  
     */  
    public Person(String name) {  
        // To do:  
        this.name=name;  
    }  
  
    public void sayHello() {  
        System.out.println("Hello!" + name);  
    }  
}
```

Overloading constructor:
Person p1 = new Person();
Person p2 = new
Person("person");

CÂU HỎI TƯƠNG TÁC



Hãy phân biệt hai quá trình Overloading và Overriding?

2.4. TỪ KHOÁ SUPER

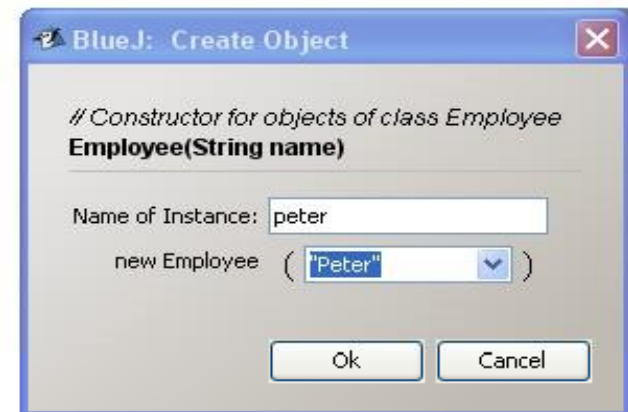
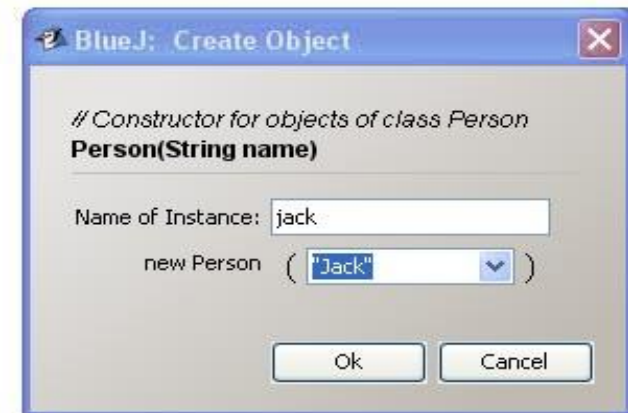
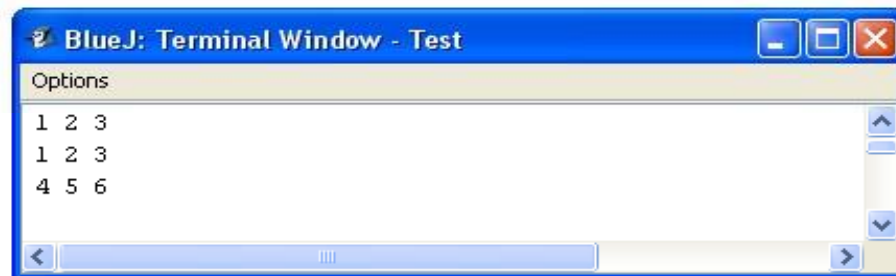
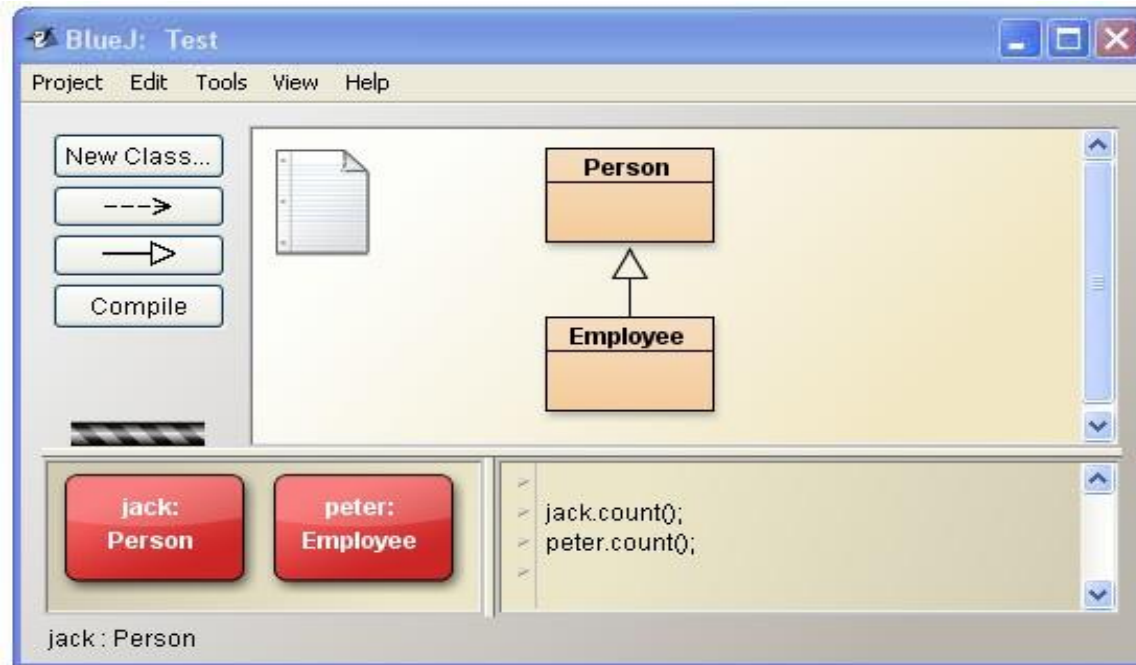
- Sử dụng từ khoá super giúp lập trình viên có thể tái sử dụng các hàm tạo cũng như các phương thức đã được cài đặt ở lớp cơ sở từ lớp con.
- Sử dụng từ khóa super để gọi lại hàm tạo của lớp cha trong hàm tạo của lớp con.

```
public class Person {  
    protected String name;  
    /**  
     * Constructor for objects of  
    class Person  
     */  
    public Person(String name) {  
        this.name=name;  
    }  
    public void count() {  
        System.out.println("1 2 3");  
    }  
}
```

```
public class Employee extends Person  
{  
    /**  
     * Constructor Employee  
     */  
    public Employee(String name) {  
        super(name);  
    }  
    public void count() {  
        super.count();  
        System.out.println("4 5 6");  
    }  
}
```


2.4. TỪ KHOÁ SUPER (tiếp theo)

Kết quả chạy trên BlueJ



CÂU HỎI TRẮC NGHIỆM

Câu hỏi 1 trên 4 ▾

Điểm: 10

Một lớp kế thừa từ một lớp khác được gọi là super-class:

- ☐ A. Đúng
- ☐ B. Sai

PROPERTIES

On passing, 'Finish' button:

On failing, 'Finish' button:

Allow user to leave quiz:

User may view slides after quiz:

User may attempt quiz:

Goes to Next Slide

Goes to Next Slide

At any time

At any time

Unlimited times

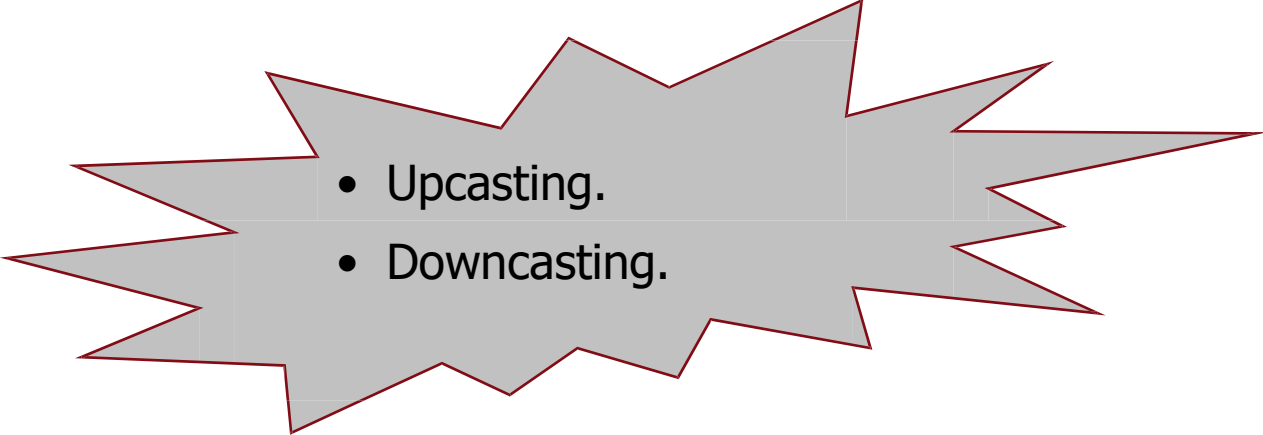


Properties...



Edit in Quizmaker

3. UPCASTING, DOWNCASTING

- 
- Upcasting.
 - Downcasting.

UPCASTING

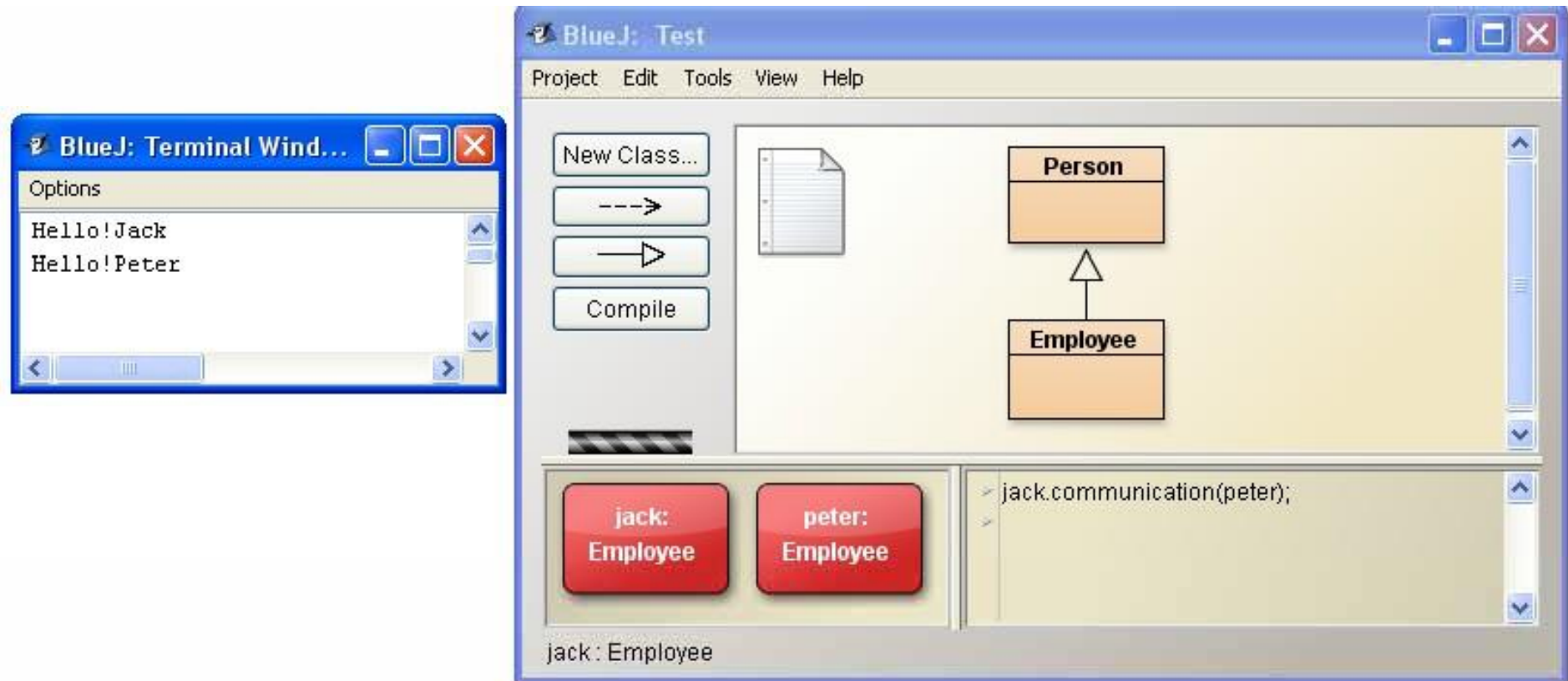
- Upcasting là một biểu hiện đặc trưng của lập trình kế thừa, khi một đối tượng thuộc lớp con được coi như là một thể hiện của lớp cha (up-casting).
- Ví dụ:



```
public class Person {
    protected String name;
    public Person(String name) {
        this.name=name;
    }
    public void sayHello() {
        System.out.println("Hello!" + name);
    }
    public void communication(Person someone)
    {
        this.sayHello();
        someone.sayHello();
    }
}
```

```
public class Employee extends
Person {
    /**
     * Constructor for objects of
     class Employee
     */
    public Employee(String name)
    {
        // To do:
        super(name);
    }
}
```

UPCASTING (tiếp theo)



DOWNCASTING

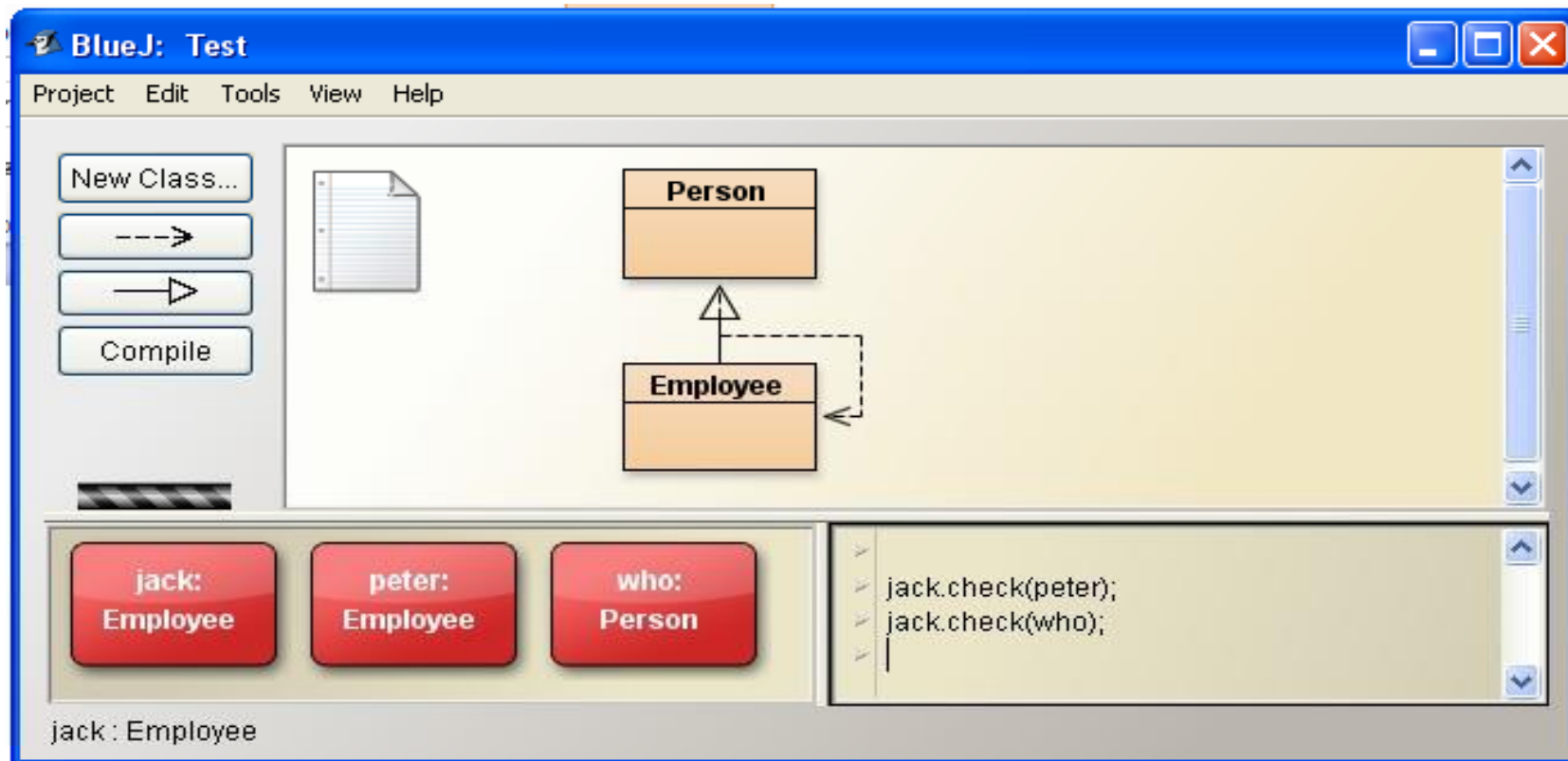
- Downcasting là cơ chế ép kiểu ngược lại với upcasting.
- Downcasting được sử dụng khi một đối tượng ban đầu được khai báo như là một thể hiện của lớp cha nhưng trong tình huống cụ thể cần phải hành xử như là một thể hiện của lớp con.
- Downcasting thông thường được sử dụng kèm với từ khoá instanceof giúp quá trình ép kiểu an toàn hơn.

```
public class Person {
    protected String name;
    /**
     * Constructor for objects of class
     Person
     */
    public Person() { }
    public Person(String name) {
        // To do:
        this.name=name;
    }
    public void check(Person someone) {
        if(someone instanceof Employee) {
            ((Employee)
            someone).printSalary();
        }
    }
}
```

```
public class Employee extends Person {
    protected double salary=100;
    /**
     * Constructor for objects of class
     Employee
     */
    public Employee(String name) {
        // To do:
        super(name);
    }
    public double getSalary() {
        return salary;
    }
    public void printSalary() {
        System.out.println(salary+"$");
    }
}
```

DOWNCASTING (tiếp theo)

Ví dụ tham khảo



CASE STUDY

```
public class Animal {
    private String name;

    public Animal() {
    }

    public Animal(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void phatAm(){
        System.out.println("Phat Am");
    }
}
```

```
public class Cat extends Animal {
    public Cat(String _name) {
        super(_name);
    }

    @Override
    public void phatAm() {
        System.out.println("Meo meo");
    }
}
```

```
public class Dog extends Animal{

    public Dog(String _name) {
        super(_name);
    }
    @Override
    public void phatAm() {
        System.out.println( "Gau gau");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Animal dog = new Dog("Dog keu: ");
        System.out.print(dog.getName());
        dog.phatAm();
        Animal cat = new Cat("Cat keu: ");
        System.out.print(cat.getName());
        cat.phatAm();

    }
}
```


BÀI TẬP

- Viết một lớp các đối tượng Person với những đặc tính sau: Có thuộc tính tên, Có thuộc tính tuổi, Có thuộc tính giới tính.
- Từ lớp Person phát triển các lớp Employee, Manager, Student và Programmer với những đặc điểm được mô tả lần lượt như sau:
 - Lớp Employee: Kế thừa những đặc điểm của Person (Employee là Person)
 - o Có thuộc tính số tiền lương cơ bản.
 - o Có thuộc tính hệ số cơ bản.
 - o Có tính năng tính lương theo công thức.
Ví dụ: Tiền lương = Hệ số \times lương cơ bản.
 - Lớp Manager: Manager cũng là Employee do vậy mang những đặc điểm của Employee.
 - o Có thêm thuộc tính lương trách nhiệm.
 - o Tính năng tính lương sẽ thay đổi lại: Tiền lương=(tiền lương tính như Employee) + lương trách nhiệm.
 - Lớp Student: Phát triển từ lớp Person. Thêm thuộc tính mã sinh viên.
 - Lớp Programmer: Phát triển từ lớp Student. Thêm thuộc tính số năm kinh nghiệm.
- Các ngôn ngữ lập trình thành thạo.
 - Có khả năng viết chương trình HelloWorld với ngôn ngữ lựa chọn.

BÀI TẬP (tiếp theo)

Kịch bản 1:

Sử dụng các lớp vừa định nghĩa xây dựng chương trình java với kịch bản sau:

- Tạo một danh sách các Employee.
- Nhập các thông số cần thiết cho employee và hiển thị ra bảng lương của các employee.

Kịch bản 2:

Sử dụng các lớp vừa định nghĩa xây dựng chương trình java với kịch bản sau:

- Khai báo và khởi tạo một đối tượng Programmer.
- Hiển thị các thông tin của Programmer.
- In ra mã nguồn chương trình HelloWorld của ngôn ngữ được chọn bằng khả năng của đối tượng Programmer.



Bài tập tham khảo 2

CÂU HỎI TRẮC NGHIỆM

Câu hỏi 1 trên 3 ▾

Điểm: 10

```
class A {}  
class B{}  
class C extend A, B{}  
Khai báo trên có đúng không?
```

- ☐ A. Đúng
- ☐ B. Sai

PROPERTIES

On passing, 'Finish' button:

On failing, 'Finish' button:

Allow user to leave quiz:

User may view slides after quiz:

User may attempt quiz:

Goes to Next Slide

Goes to Next Slide

At any time

At any time

Unlimited times



Properties...



Edit in Quizmaker

TÓM LƯỢC CUỐI BÀI

Sau khi học xong bài này chúng ta đã nắm được các kiến thức sau:

- Nắm được khái niệm về tính kế thừa trong lập trình hướng đối tượng;
- Nắm được cách lập trình kế thừa trong Java;
- Hiểu về vấn đề Upcasting – Downcasting.

CÂU HỎI TRẮC NGHIỆM

Câu hỏi 1 trên 7

Điểm: 10

Cho một phần của mã trong lớp Widget:

```
1: class Widget extends Thingee {  
2:     private int widgetCount = 0;  
3:     int addWidget(){  
4:         widgetCount++;  
5:         return widgetCount;  
6:     }  
7:     String wName;  
8:     public Widget(int mx, String t){  
9:         wName = "I am Widget #" + addWidget();  
10:    }  
11: }
```

Ý nghĩa của từ private trong dòng 2 là gì?

- ☐ A. Vì widgetCount là private, nên tất cả phương thức trong lớp Widget và các lớp khác đều có thể truy xuất tới.
- ☐ B. Vì widgetCount là private, nên chỉ phương thức addWidget() mới có thể truy xuất nó.

PROPERTIES

On passing, 'Finish' button:

On failing, 'Finish' button:

Allow user to leave quiz:

User may view slides after quiz:

User may attempt quiz:

Goes to Next Slide

Goes to Next Slide

At any time

At any time

Unlimited times



Properties...



Edit in Quizmaker

Search

Một lớp tại một thời điểm có thể kế thừa được nhiều lớp cùng một lúc hay không?

Một lớp tại một thời điểm có thể kế thừa được nhiều lớp cùng một lúc hay không?

Gợi ý:

Một lớp tại một thời điểm chỉ có thể được kế thừa một lớp, không được phép kế thừa nhiều lớp cùng một lúc.

Để thực hiện việc đa kế thừa phải làm thế nào?

Lớp được khai báo với từ khóa là final có ý nghĩa gì?

Phương thức được khai báo với từ khóa final có ý nghĩa gì?

Phân biệt Overloading và Overriding?

PROPERTIES

Allow user to leave interaction:

Show 'Next Slide' Button:

Completion Button Label:

Anytime

Don't show

Next Slide



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Select a term:

Base-class

Derived-class

Downcasting

Overloading

Overriding

Sub-class

Supper class

Từ khóa instanceof

Từ khóa super

Upcasting

Base-class

Lớp Cơ sở - lớp cha.

PROPERTIES

Allow user to leave interaction:

Show 'Next Slide' Button:

Completion Button Label:

Anytime

Don't show

Next Slide

