

# Bài 46: Các thành phần protected

---

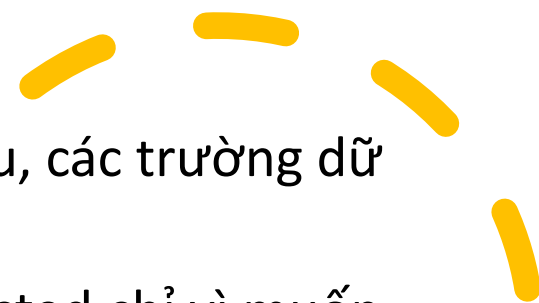
- ✓ Mục đích sử dụng
- ✓ Các trường protected
- ✓ Các phương thức protected
- ✓ Phạm vi khả dụng của các access modifier

# Mục đích sử dụng

- Sử dụng keyword protected khi muốn tạo ra một lớp với mục đích kế thừa
- Khi muốn tạo một lớp mà các tính năng của nó chỉ được sử dụng bởi các lớp con, không phải các lớp khác.



# Các trường protected

- 
- Để đảm bảo tính đóng gói dữ liệu, các trường dữ liệu của lớp luôn để private
  - Không sử dụng các trường protected chỉ vì muốn được truy cập trực tiếp vào dữ liệu của lớp cha từ lớp con
  - Khi sử dụng trường protected sẽ có khả năng rủi ro: kẻ xấu sẽ lợi dụng khả năng truy cập trực tiếp vào dữ liệu của lớp cha qua lớp con. Do đó chúng chỉ cần khai báo một lớp con của lớp có dữ liệu cần truy cập là có thể lấy được các thông tin đó từ lớp cha

# Ví dụ

- Ví dụ sau được chấp nhận trong Java nhưng không khuyến khích:

```
import java.util.Date;

public class Person {
    protected String fullName;
    private Date dateOfBirth;
    //...
}

class Student extends Person {
    private String studentId;
    //...

    public static void main(String[] args) {
        Student nam = new Student();
        nam.fullName = "Trần Văn Nam"; //ok but don't do this
    }
}
```



# Khuyến nghị

➤ Hãy làm như sau:

```
public class Person {  
    private String fullName;  
    private Date dateOfBirth;  
    //...  
  
    public String getFullName() {  
        return fullName;  
    }  
  
    public void setFullName(String fullName) {  
        this.fullName = fullName;  
    }  
  
class Student extends Person {  
    private String studentId;  
    //...  
  
    public static void main(String[] args) {  
        Student nam = new Student();  
        nam.setFullName("Trần Văn Nam"); // very good!  
    }  
}
```

# Các phương thức protected

- Các outer class, interface không thể khai báo với access modifier là protected
- Ví dụ:

```
package lesson46;  
  
import java.util.Date;  
  
protected class Person { // error!  
    private String fullName;  
    private Date dateOfBirth;  
}
```

# Các phương thức protected

➤ Các phương thức protected thường được sử dụng trong mối quan hệ kế thừa nhằm cung cấp các tính năng cho chỉ những lớp trong mối quan hệ kế thừa này sử dụng

➤ Ví dụ:

```
public class Father {  
    private String bankAccPassword; // mật khẩu tk ngân hàng  
    //...  
    protected void setBankAccPassword(String newPws) {  
        bankAccPassword = newPws;  
    }  
}  
package lesson46;  
  
public class Child extends Father {  
    // các trường dữ liệu...  
    //...  
    // phương thức cập nhật mật khẩu:  
    public void updateBankInfo(String newPassword) {  
        if(newPassword.contains("[a-zA-Z0-9/+-$"] )) { // các điều kiện...  
            setBankAccPassword(newPassword); // thiết lập mật khẩu ok  
        }  
    }  
}
```

# Các phương thức protected

- Nếu truy cập từ lớp khác không trong mối quan hệ kế thừa thì sẽ không thực hiện được:

```
import lesson46.Child;
import lesson46.Father;

public class Test {
    public static void main(String[] args) {
        Father myFather = new Father();
        Child child = new Child();

        myFather.setBankAccPassword("0123456*Avx$"); // error!
        child.updateBankInfo("0123456*Avx$"); // ok
    }
}
```



# Các phương thức protected

- Các protected constructor được sử dụng để ngăn cấm việc tạo đối tượng của lớp từ bên ngoài gói:

```
package lesson46;

import java.util.Date;

public class Person {
    private String fullName;
    private Date dateOfBirth;
    //...

    protected Person() { // protected constructor
        fullName = "No Name";
        dateOfBirth = new Date();
    }
}

package other;

import lesson46.Person;

public class Test {
    public static void main(String[] args) {
        Person person = new Person(); // error!
    }
}
```

# Phạm vi khả dụng của access modifier

- Các thành phần private: chỉ khả dụng trong phạm vi lớp chứa nó
- Các thành phần public: khả dụng với mọi lớp
- Các thành phần protected: khả dụng trong phạm vi lớp chứa nó, các lớp con, các lớp cùng gói
- Các thành phần mặc định(không có access modifier): khả dụng trong lớp chứa nó và các lớp trong cùng gói



# Minh họa cụ thể

➤ Truy cập các thành phần protected của lớp

# Nội dung tiếp theo

Ghi đề phương thức