



MENU



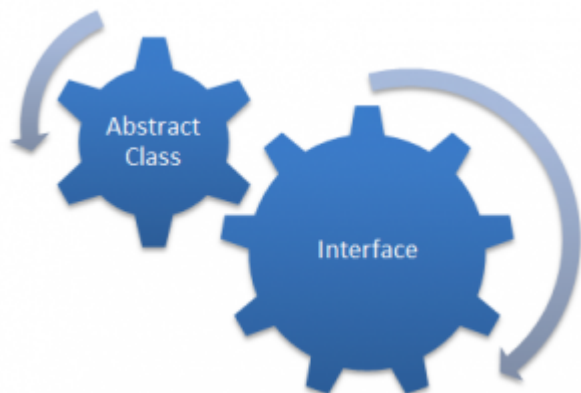
## Abstract class và Interface trong Java

Xem thêm các chuyên mục:

[Java cơ bản](#)

- 1- Giới thiệu
- 2- Class trừu tượng (Abstract Class)
- 3- Ví dụ với lớp trừu tượng
- 4- Tổng quan về Interface
- 5- Cấu trúc của một Interface
- 6- Class thực hiện Interface

260  
Shares



### 1- Giới thiệu

Trong tài liệu hướng dẫn này tôi sẽ hướng dẫn về Interface và class trừu tượng (Abstract Class). Đồng thời phân tích sự giống và khác nhau giữa chúng.

### 2- Class trừu tượng (Abstract Class)

Abstract class (Class trừu tượng). Hãy xem ví dụ về một class như thế:

Top  
 Middle page  
 Bottom

## Những việc làm hấp dẫn



### Java Developer/Application Management Support (J2EE, JavaScript)

QR Retail Automation 📍 Ho Chi Minh 💰 \$500 - \$1,500

J2EE

Java

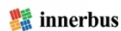
JavaScript



### [All Levels] 10 Java Developers

ECORAU Technology 📍 Ha Noi 💰 \$400 - \$1,500

Java



### Full-Stack Developer (Java, Web, Spring, VueJS)

VNIB TECH Co. Ltd 📍 Ho Chi Minh 💰 Negotiable

Java

Full-Stack

Web

Spring

VueJS

```

1 // Đây là một lớp trừu tượng.
2 // Nó bắt buộc phải khai báo là abstract
3 // vì trong nó có một phương thức trừu tượng
4 public abstract class ClassA {
5
6     // Đây là một phương thức trừu tượng.
7     // Nó không có thân (body).
8     // Access modifier của phương thức này là public.
9     public abstract void doSomething();
10
11     // Access modifier của phương thức này là protected.
12     protected abstract String doNothing();
13     // Phương thức này không khai báo access modifier.
14     // Access modifier của nó là mặc định.
15     abstract void todo();
16 }
17
18 // Đây là một lớp trừu tượng.
19 // Nó được khai báo là abstract,
20 // mặc dù nó không có phương thức trừu tượng nào.
21 public abstract class ClassB {
22
23 }
```

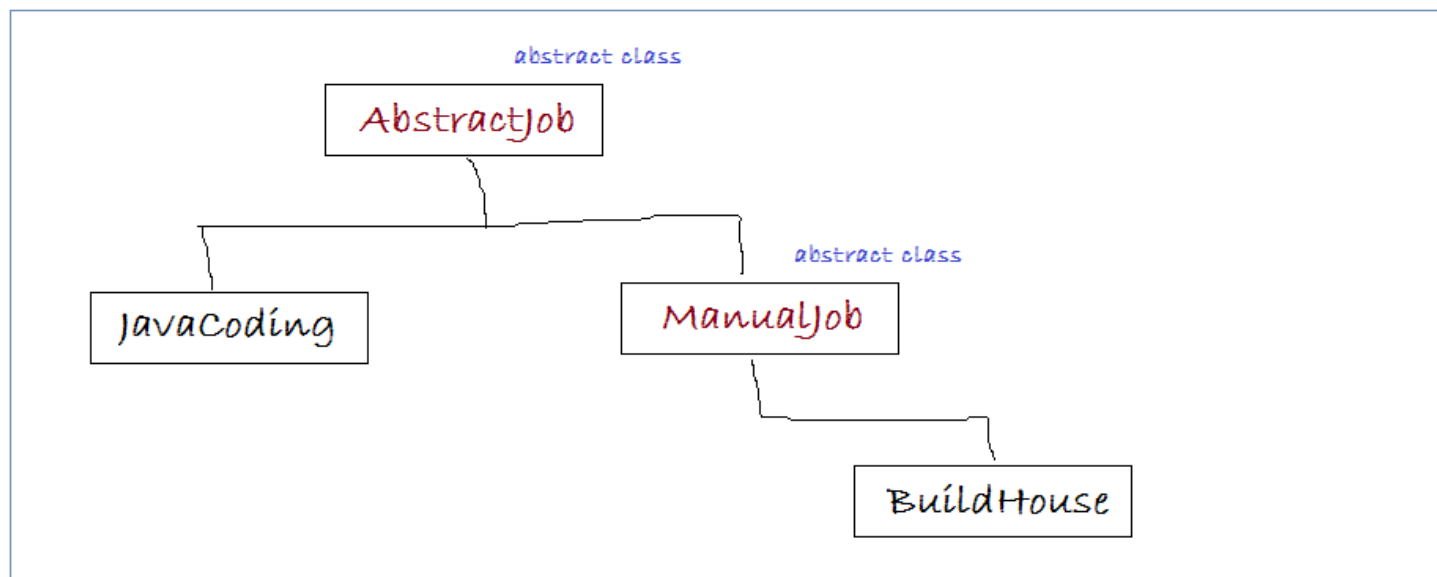
Đặc điểm của một class trừu tượng là:

1. Nó được khai báo **abstract**.
2. Nó có thể khai báo 0, 1 hoặc nhiều method trừu tượng bên trong.
3. Không thể khởi tạo 1 đối tượng trực tiếp từ một class trừu tượng.



### 3- Ví dụ với lớp trừu tượng

Hãy xem hình minh họa:



#### AbstractJob.java

```
1 package org.o7planning.tutorial.abs;
2
3 // Một lớp trừu tượng (Mô phỏng một công việc)
4 // Nó khai báo 2 phương thức trừu tượng.
5 public abstract class AbstractJob {
6
7     public AbstractJob() {
8
9     }
10
11 // Đây là một phương thức trừu tượng.
12 // Method này trả về tên của công việc.
13 public abstract String getJobName();
14
15 // Đây là một phương thức trừu tượng.
16 public abstract void doJob();
17
18 }
```

?

#### JavaCoding.java

```
1 package org.o7planning.tutorial.abs;
2
3 // Lớp này thực hiện hết các phương thức trừu tượng của lớp cha.
4 public class JavaCoding extends AbstractJob {
5
6     public JavaCoding() {
7
8     }
9
10 // Thực hiện phương thức trừu tượng khai báo tại lớp cha.
11 @Override
12 public void doJob() {
```

?

```
12     System.out.println("Coding Java...");
13 }
14
15 // Thực hiện phương thức trừu tượng khai báo tại lớp cha.
16 // Phương thức này sẽ có thân (body).
17 // Và trả về tên của công việc.
18 @Override
19 public String getJobName() {
20     return "Coding Java";
21 }
22
23 }
```

### ManualJob.java

```
1 package org.o7planning.tutorial.abs;
2
3 // ManualJob - (Mô phỏng một công việc phổ thông)
4 // Lớp cha (AbstractJob) có 2 phương thức trừu tượng.
5 // Lớp này mới chỉ thực hiện 1 phương thức trừu tượng của lớp cha.
6 // Vì vậy nó bắt buộc phải khai báo là abstract.
7 public abstract class ManualJob extends AbstractJob {
8
9     public ManualJob() {
10
11     }
12
13 // Thực hiện phương thức trừu tượng của lớp cha.
14 @Override
15 public String getJobName() {
16     return "Manual Job";
17 }
18
19 }
```

### BuildHouse.java

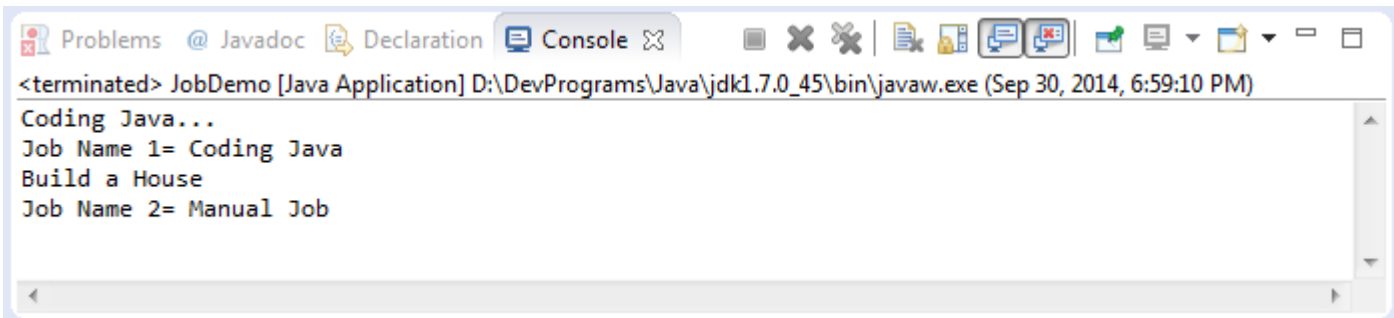
```
1 package org.o7planning.tutorial.abs;
2
3 // Class này thừa kế từ class trừu tượng ManualJob
4 // BuildHouse không được khai báo là trừu tượng.
5 // Vì vậy nó cần thực hiện tất cả các phương thức trừu tượng còn lại.
6 public class BuildHouse extends ManualJob {
7
8     public BuildHouse() {
9
10    }
11
12 // Thực hiện phương thức trừu tượng của lớp cha.
13 @Override
14 public void doJob() {
15     System.out.println("Build a House");
16 }
17
18 }
```

## Ví dụ demo

## JobDemo.java

```
1 package org.o7planning.tutorial.abs;
2
3 public class JobDemo {
4
5     public static void main(String[] args) {
6
7         // Khởi tạo một đối tượng AbstractJob
7         // từ Constructor của lớp JavaCoding.
9         AbstractJob job1 = new JavaCoding();
10
11        // Gọi phương thức doJob()
12        job1.doJob();
13
14        // Phương thức getJobName là trừu tượng trong lớp AbstractJob.
15        // Nhưng nó đã được thực hiện tại một lớp con nào đó.
16        // Vì vậy bạn có thể gọi nó.
17        String jobName = job1.getJobName();
18
19        System.out.println("Job Name 1= " + jobName);
20
21        // Khởi tạo một đối tượng AbstractJob
22        // từ Constructor của lớp BuildHouse.
23        AbstractJob job2 = new BuildHouse();
24
25        job2.doJob();
26
27        String jobName2 = job2.getJobName();
28
29        System.out.println("Job Name 2= " + jobName2);
30
31    }
32 }
```

Kết quả chạy ví dụ:



## 4- Tổng quan về Interface

Chúng ta biết rằng một class chỉ có thể mở rộng từ một class cha.

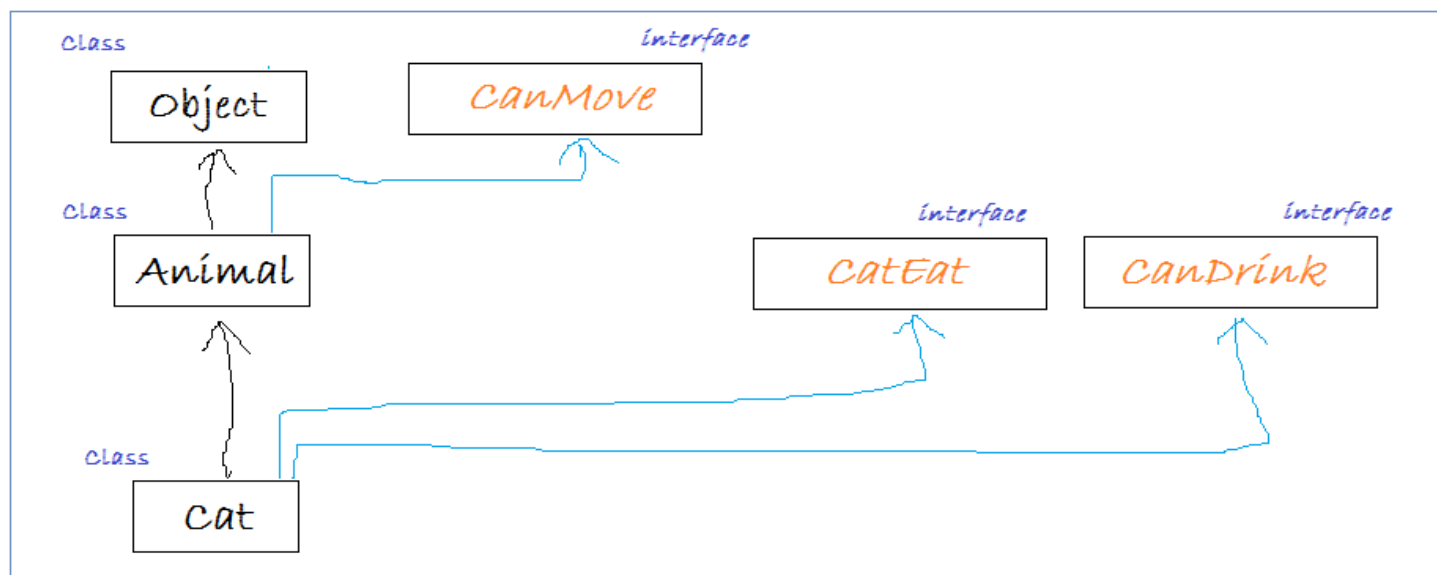
```
1 // Lớp B là con của lớp A, hoặc nói cách khác là B mở rộng từ A
2 // Java chỉ cho phép một lớp mở rộng từ duy nhất một lớp khác.
```

```

3  public class B extends A {
4      // ....
5  }
6
7  // Trong trường hợp không chỉ rõ lớp B mở rộng từ một lớp cụ thể nào.
8  // Mặc định, hiểu rằng B mở rộng từ lớp Object.
9  public class B {
10
11  }
12
13  // Cách khai báo này, và cách phía trên là tương đương nhau.
14  public class B extends Object {
15
16  }

```

Nhưng một **class** có thể mở rộng từ nhiều **Interface**



```

1  // Một lớp chỉ có thể mở rộng từ 1 lớp cha.
2  // Nhưng có thể thực hiện (mở rộng) từ nhiều Interface.
3  public class Cat extends Animal implements CanEat, CanDrink {
4
5      // ....
6  }

```

## Các đặc điểm của interface

1. Interface luôn luôn có modifier là: **public interface**, cho dù bạn có khai báo rõ hay không.
2. Nếu có các trường (field) thì chúng đều là: **public static final**, cho dù bạn có khai báo rõ hay không.
3. Các method của nó đều là method trừu tượng, nghĩa là không có thân hàm, và đều có modifier là: **public abstract**, cho dù bạn có khai báo hay không.
4. Interface không có Constructor (cấu tử).

## 5- Cấu trúc của một Interface

### NoAccessModifierInterface.java

```
1 package org.o7planning.tutorial.itf;
2
3 // Đây là một interface không chỉ định rõ 'access modifier'.
4 // Access modifier của nó là mặc định.
5 // Chỉ các lớp cùng package mới có thể thi hành interface này.
6 interface NoAccessModifierInterface {
7
8 }
```

?

### CanMove.java

```
1 package org.o7planning.tutorial.itf;
2
3 // Interface này định nghĩa một tiêu chuẩn
4 // về những thứ có khả năng di chuyển.
5 public interface CanMove {
6
7     // Các phương thức trong Interface đều là trừu tượng và public.
8     public abstract void run();
9
10    // Cho dù bạn không viết rõ 'public abstract' thì java luôn hiểu là vậy.
11    void back();
12
13    // Tốc độ.
14    public int getVelocity();
15
16 }
```

?

### CanDrink.java

```
1 package org.o7planning.tutorial.itf;
2
3 // Interface này định nghĩa ra một tiêu chuẩn
4 // về những thứ có khả năng biết uống.
5 public interface CanDrink {
6
7     // Các trường (field) trong Interface đều là 'public static final'.
8     // Cho dù bạn có khai báo rõ hay không java luôn hiểu ngầm vậy.
9     public static final String PEPSI = "PEPSI";
10     final String NUMBER_ONE = "NUMBER ONE";
11     String SEVENUP = "SEVEN UP";
12
13     public void drink();
14
15 }
```

?

### CanEat.java

```
1 package org.o7planning.tutorial.itf;
2
3 // Interface này định nghĩa ra một tiêu chuẩn
```

 Top ?  
 Middle page  
 Bottom

```
4 // về những thứ có khả năng biết ăn.  
5 public interface CanEat {  
6  
7     public void eat();  
8 }
```

## 6- Class thực hiện Interface

### Animal.java

```
1 package org.o7planning.tutorial.cls;  
2  
3 import org.o7planning.tutorial.itf.CanMove;  
4  
5 // Animal (Động vật).  
6 // Lớp này mở rộng từ lớp Object (Mặc dù không ghi rõ).  
7 // Và được khai báo thực hiện hiện (hoặc gọi là thừa kế) interface CanMove.  
8 // Interface CanMove có 3 phương thức trừu tượng.  
9 // Lớp này chỉ thực hiện 1 phương thức.  
10 // Vì vậy nó bắt buộc phải khai báo là abstract.  
11 // Các phương thức trừu tượng còn lại sẽ được các lớp con thực hiện.  
12 public abstract class Animal implements CanMove {  
13  
14     // Thực hiện phương thức run() của interface CanMove.  
15     @Override  
16     public void run() {  
17         System.out.println("Animal run...");  
18     }  
19  
20 }
```

### Cat.java

```
1 package org.o7planning.tutorial.cls;  
2  
3 import org.o7planning.tutorial.itf.CanDrink;  
4 import org.o7planning.tutorial.itf.CanEat;  
5  
6 // Lớp Cat mở rộng từ lớp Animal và thi hành 2 interface CanEat, CanDrink.  
7 // Cat là lớp thông thường (Nó không được khai báo là trừu tượng).  
8 // Vì vậy nó phải thực hiện mọi phương thức trừu tượng của các Interface.  
9 public class Cat extends Animal implements CanEat, CanDrink {  
10  
11     private String name;  
12  
13     public Cat(String name) {  
14         this.name = name;  
15     }  
16  
17     public String getName() {  
18         return this.name;  
19     }  
20  
21     // Thực hiện phương thức của interface CanMove.  
22     @Override  
23     public void back() {
```



```
24     System.out.println(name + " cat back ...");
25 }
26
27 // Thực hiện phương thức của interface CanMove.
28 @Override
29 public int getVelocity() {
30     return 110;
31 }
32
33 // Thực hiện phương thức của interface CanEat.
34 @Override
35 public void eat() {
36     System.out.println(name + " cat eat ...");
37 }
38
39 // Thực hiện phương thức của interface CanDrink.
40 @Override
41 public void drink() {
42     System.out.println(name + " cat drink ...");
43 }
44
45 }
```

### Mouse.java

```
1 package org.o7planning.tutorial.cls;
2
3 import org.o7planning.tutorial.itf.CanDrink;
4 import org.o7planning.tutorial.itf.CanEat;
5
6 public class Mouse extends Animal implements CanEat, CanDrink {
7
8     @Override
9     public void back() {
10         System.out.println("Mouse back ...");
11     }
12
13     @Override
14     public int getVelocity() {
15         return 85;
16     }
17
18     @Override
19     public void drink() {
20         System.out.println("Mouse drink ...");
21     }
22
23     @Override
24     public void eat() {
25         System.out.println("Mouse eat ...");
26     }
27
28 }
```

### AnimalDemo.java

```
1 package org.o7planning.tutorial.cls;
2
```

```
3  import org.o7planning.tutorial.itf.CanEat;
4
5  public class AnimalDemo {
6
7      public static void main(String[] args) {
8
9          // Thừa kế trường tĩnh từ interface CanDrink.
10         System.out.println("Drink " + Cat.SEVENUP);
11
12         // Khởi tạo một đối tượng CanEat.
13         // Một đối tượng khai báo là CanEat.
14         // Nhưng thực tế là Cat.
15         CanEat canEat1 = new Cat("Tom");
16
17         // Một đối tượng khai báo là CanEat.
18         // Nhưng thực tế là Mouse.
19         CanEat canEat2 = new Mouse();
20
21         // Tính đa hình (Polymorphism) thể hiện rõ tại đây.
22         // Java luôn biết một đối tượng là kiểu gì
23         // ==> Tom cat eat ...
24         canEat1.eat();
25         // ==> Mouse eat ...
26         canEat2.eat();
27
28         boolean isCat = canEat1 instanceof Cat;
29
30         System.out.println("catEat1 is Cat? " + isCat);
31
32         // Ép kiểu (Cast).
33         if (canEat2 instanceof Mouse) {
34             Mouse mouse = (Mouse) canEat2;
35
36             // Gọi phương thức drink (Thừa kế từ CanDrink).
37             mouse.drink();
38         }
39     }
40 }
41 }
```

Kết quả chạy ví dụ:

```
<terminated> AnimalDemo [Java Application] D:\DevPrograms\Java\jdk1.7.0_45\bin\javaw.exe (Sep 30, 2014, 9:45:00 PM)
Drink SEVEN UP
Tom cat eat ...
Mouse eat ...
catEat1 is Cat? true
Mouse drink ...
```

Xem thêm các chuyên mục:

Top  
 Middle page  
 Bottom

## Java cơ bản

---

- Bắt đầu với Java cần những gì?
- Hướng dẫn cài đặt và cấu hình Java
- Hướng dẫn cài đặt và cấu hình Java trên Ubuntu
- Hướng dẫn cài đặt và cấu hình Eclipse
- Hướng dẫn cài đặt và cấu hình Eclipse trên Ubuntu
- Học nhanh Java cho người mới bắt đầu
- Lịch sử của bit và byte trong khoa học máy tính
- Các kiểu dữ liệu trong Java
- Các toán tử Bitwise
- Câu lệnh rẽ nhánh (if else) trong Java
- Câu lệnh rẽ nhánh switch trong Java
- Vòng lặp trong Java
- Mảng (Array) trong Java
- JDK Javadoc định dạng CHM
- Thừa kế và đa hình trong Java
- Abstract class và Interface trong Java
- Access modifier trong Java
- Hướng dẫn sử dụng Java Enum
- Hướng dẫn sử dụng Java Annotation
- So sánh và sắp xếp trong Java
- Hướng dẫn sử dụng Java String, StringBuffer và StringBuilder
- Hướng dẫn xử lý ngoại lệ trong Java - Java Exception Handling
- Hướng dẫn sử dụng Java Generics
- Hướng dẫn sử dụng nền tảng tập hợp trong Java (Java Collection Framework)
- Thao tác với tập tin và thư mục trong Java
- Hướng dẫn sử dụng luồng vào ra nhị phân trong Java
- Hướng dẫn sử dụng luồng vào ra ký tự trong Java
- Hướng dẫn sử dụng Date, Time trong Java
- Cú pháp và các tính năng mới trong Java 5
- Cú pháp và các tính năng mới trong Java 8
- Hướng dẫn sử dụng biểu thức chính quy trong Java
- Hướng dẫn lập trình đa luồng trong Java - Java Multithreading
- Thư viện điều khiển các loại cơ sở dữ liệu khác nhau trong Java

- Hướng dẫn sử dụng Java JDBC kết nối cơ sở dữ liệu
- Lấy các giá trị của các cột tự động tăng khi Insert một bản ghi sử dụng JDBC
- Hướng dẫn nén và giải nén trong Java
- Hướng dẫn sử dụng Java Reflection
- Hướng dẫn gọi phương thức từ xa với Java RMI
- Hướng dẫn lập trình Java Socket
- Các nền tảng nào bạn nên chọn để lập trình ứng dụng Java Desktop?
- Hướng dẫn sử dụng Java Commons IO
- Hướng dẫn sử dụng Java Commons Email
- Hướng dẫn sử dụng Java Commons Logging
- Ví dụ mã hóa và giải mã trong Java sử dụng Apache Base64 (Base64 encode/decode)

## Tài liệu mới nhất

---

- Hướng dẫn sử dụng HTML Video
- Hướng dẫn sử dụng Javascript URL Encoding
- Hướng dẫn sử dụng HTML Entity
- Hướng dẫn sử dụng HTML URL Encoding
- Hướng dẫn sử dụng HTML Quotation (Các trích dẫn)
- Hướng dẫn sử dụng HTML Heading
- Hướng dẫn sử dụng HTML IFrame
- Hướng dẫn sử dụng HTML Hyperlink
- Hướng dẫn sử dụng HTML List
- Hướng dẫn sử dụng HTML Image



o7planning.org

Liên kết bạn bè: [freetuts.net](https://freetuts.net) | [laptrinhvb.net](https://laptrinhvb.net) | [canva.com](https://canva.com)