

Like & Share & Subscribe

Anh Nguyen Ngoc

ProIT4All



# Giới thiệu Generics



Fan page: <http://facebook.com/Proit4All>



# Khái niệm Generics

Ví dụ: Sử dụng ArrayList với các kiểu dữ liệu String

```
ArrayList<String>mylist = new  
    ArrayList<String> ();
```

```
mylist.add("Hello");  
mylist.add("Goodbye");
```

Lấy ra:

```
String str = mylist.get(0);
```

*Generic*



# Ưu điểm của generic

- ❑ Kiểm tra kiểu dữ liệu trong thời điểm biên dịch
  - ❖ Trình biên dịch Java áp dụng việc kiểm tra đoạn mã generic để phát hiện các vấn đề như vi phạm an toàn kiểu dữ liệu.
  - ❖ Việc sửa lỗi tại thời gian biên dịch dễ dàng hơn nhiều khi sửa chữa lỗi tại thời điểm chạy chương trình.





# Ưu điểm của generic

## ❑ Không cần ép kiểu dữ liệu

❖ Đoạn code sau đây không dùng generic nên phải ép kiểu:

```
List list = new ArrayList();
```

```
list.add("hello");
```

```
String s = (String) list.get(0); //phải ép kiểu
```

❖ Khi dùng generic, không cần ép kiểu:

```
List<String> list = new ArrayList<String>();
```

```
list.add("hello");
```

```
String s = list.get(0); // không ép kiểu
```



# Ưu điểm của generic

□ Cho phép người lập trình viên thực hiện các thuật toán tổng quát.

❖ Bằng cách sử dụng generics, người lập trình có thể

- *Thực hiện các thuật toán tổng quát*
- *Với các kiểu dữ liệu tùy chọn khác nhau,*
- *Nội dung đoạn code trở nên rõ ràng và dễ hiểu.*



# Phân loại ArrayList

ArrayList

ArrayList  
(Không định kiểu)

ArrayList có thể chứa các phần tử bất kể kiểu dữ liệu nào

- + Các phần tử trong ArrayList được coi như một tập các đối tượng (kiểu **Object**)
- + Khi truy xuất các phần tử, cần **ép về kiểu gốc** của phần tử để xử lý

ArrayList<**Type**>  
(Có định kiểu)

ArrayList chỉ chứa các phần tử có **kiểu đã chỉ định**.

- + Khi truy xuất các phần tử **không cần ép** về kiểu gốc của phần tử để xử lý
- + Chặt chẽ, tránh rủi ro lập trình
- + Hiệu suất xử lý nhanh hơn





Like & Share & Subscribe

Fan page: <http://facebook.com/Proit4All>

# DEMO

## Generics



Java™





# ArrayList<Type> định kiểu

```
ArrayList<String> a = new ArrayList<String>();  
a.add("apple");  
a.add("mango");  
a.add("banana");  
String s = a.get(2);
```

+ Khi truy xuất các phần tử **không cần ép** về kiểu gốc của phần tử để xử lý

a.add(String)

String  
String  
String

a.get(index)

ArrayList<String>

**Chú ý: <Type> là kiểu dữ liệu không phải kiểu nguyên thủy (phải sử dụng wrapper)**





# Non-Generics

Ví dụ: Sử dụng ArrayList với các kiểu dữ liệu khác nhau

```
ArrayList mylist = new ArrayList();  
mylist.add(10);  
mylist.add("Hello");  
mylist.add(true);  
mylist.add(15.75);
```

Lấy ra:

```
int a = (Integer) mylist.get(0);  
String str = (String) mylist.get(1);
```

Ví dụ: Sử dụng ArrayList với các kiểu dữ liệu Integer

```
ArrayList<Integer> mylist = new  
    ArrayList<Integer>();
```

```
mylist.add(10);  
mylist.add("Hi"); //error  
mylist.add(true); //error  
mylist.add(15);
```

*Generic*

Lấy ra:

```
int a = mylist.get(0);
```

thuydung

ProIT4All

Fan page: <http://facebook.com/Proit4All>



THANK YOU

<http://youtube.com/@AnhNguyenNgoc>



Java™

