
Generic

Generic

Generics → Tham số hóa kiểu dữ liệu

```
ArrayList<Integer> arr = new ArrayList<Integer>();  
arr.add(5);  
arr.add(35);  
arr.add("Java"); //error  
arr.add(true);   //error
```

Generic

Một số quy ước đặt tên kiểu tham số generic

Ký tự	Ý nghĩa
E	Element (Phần tử)
K	Key (Khóa)
V	Value (Giá trị)
T	Type (Kiểu dữ liệu)
N	Number (Số)

Lớp generic

Một lớp có thể tham chiếu bất kỳ kiểu đối tượng nào được gọi là lớp generic

```
public class MyGeneric<T> {  
    public T obj;  
    public T getObj() {  
        return obj;  
    }  
    public void add(T obj) {  
        this.obj = obj;  
    }  
}
```

Lớp generic



Lớp generic

Sử dụng kiểu
Integer


```
public static void main(String[] args) {  
    //Use Integer  
    MyGeneric<Integer> myGeneric1 = new MyGeneric<Integer>();  
    myGeneric1.add(3);  
    System.out.println(myGeneric1.getObj());  
  
    //Use String  
    MyGeneric<String> myGeneric2 = new MyGeneric<String>();  
    myGeneric2.add("java");  
    System.out.println(myGeneric2.getObj());  
}
```

Sử dụng kiểu
String

Phương thức generic

Một phương thức trong class hoặc interface đều có thể sử dụng generic

Phương thức generic



```
public static <E> void printArray(E[] elements) {  
    for (E element : elements) {  
        System.out.print(element + " ");  
    }  
    System.out.println();  
}
```

```
public static void main(String[] args) {  
    Integer[] intArray = { 10, 20, 30, 40, 50 };  
    Character[] charArray = { 'J', 'A', 'V', 'A' };  
  
    System.out.print("Mang so nguyen: ");  
    printArray(intArray);  
  
    System.out.print("Mang ky tu: ");  
    printArray(charArray);  
}
```

Gọi tới phương thức
generic

Mảng generic

Có thể khai báo một mảng generic nhưng không thể khởi tạo mảng generic vì kiểu generic không tồn tại tại thời điểm chạy. Generic chỉ có tác dụng với trình biên dịch để kiểm soát code.

Khai báo

```
T[] arr; // Ok
```

```
T[] arr2 = new T[5]; // Error
```

Khởi tạo

Thừa kế lớp generic

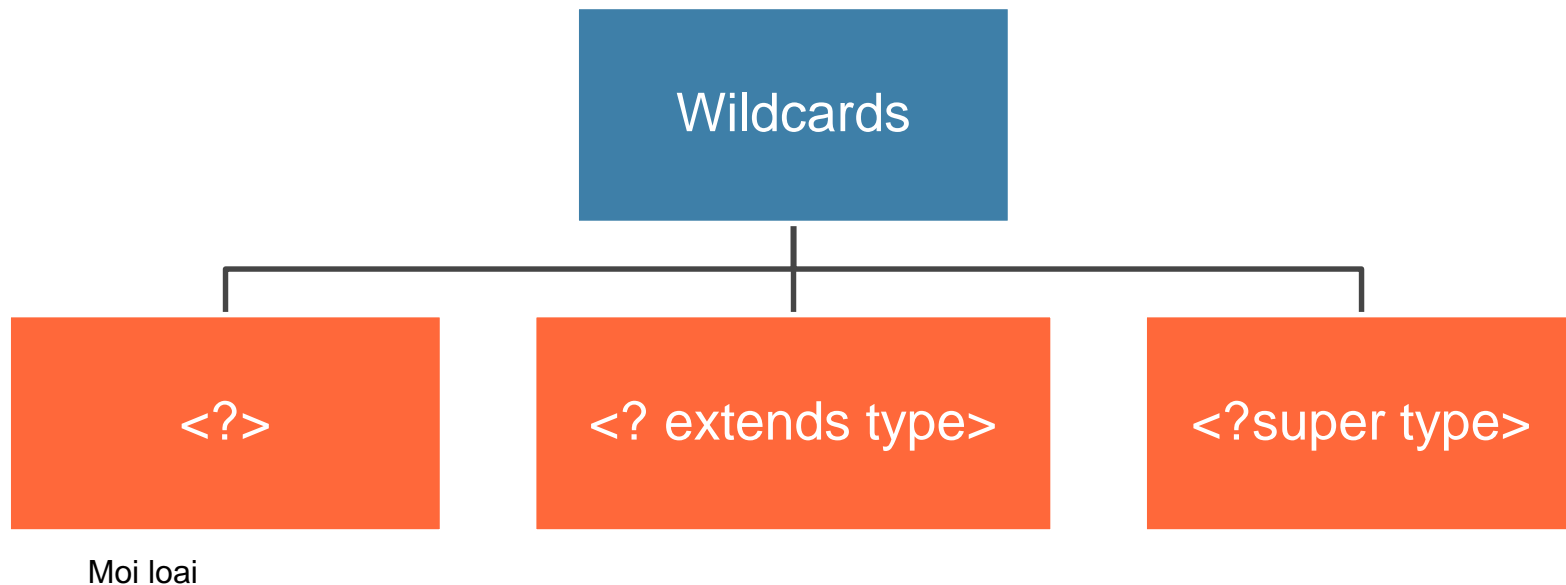
Một class mở rộng từ một class generics, nó có thể chỉ định rõ kiểu cho tham số Generics, giữ nguyên các tham số Generics hoặc thêm các tham số Generics

```
public abstract class AbstractParam<T> {  
    protected T value;  
    protected abstract void printValue();  
}
```

```
public class Email extends AbstractParam<String>{  
    @Override  
    protected void printValue() {  
        System.out.println("My email is:"+value);  
    }  
}
```

```
public static void main(String[] args) {  
    Email email = new Email();  
    email.value = "ngoc@techmaster.vn";  
    email.printValue();// My email is:ngoc@techmaster.vn  
    email.value = 10; // Lỗi  
}
```

Các ký tự đại diện generic



Ưu điểm của generics

01

Kiểu dữ liệu an toàn

02

Kiểm tra dữ liệu chặt chẽ ở Compile-time mà không phải là Runtime-error

03

Hạn chế việc ép kiểu (cast) thủ công mà không an toàn.

04

Giúp chúng ta viết các thuật toán được sử dụng nhiều, dễ dàng thay đổi, an toàn dữ liệu và dễ đọc hơn

Hạn chế của generics

- Không thể gọi Generics bằng kiểu dữ liệu nguyên thủy (Primitive type: int, long, double, ...), thay vào đó sử dụng các kiểu dữ liệu Object (wrapper class thay thế: Integer, Long, Double, ...).
- Không thể tạo instances của kiểu dữ liệu Generics, thay vào đó sử dụng reflection từ class.
- Không thể sử dụng static cho Generics.
- Không thể ép kiểu hoặc sử dụng instanceof.
- Không thể tạo mảng với parameterized types.
- Không thể tạo, catch, throw đối tượng của parameterized types (Generic Throwable)
- Không thể overload các hàm trong một lớp

Collections

Collection là gì?

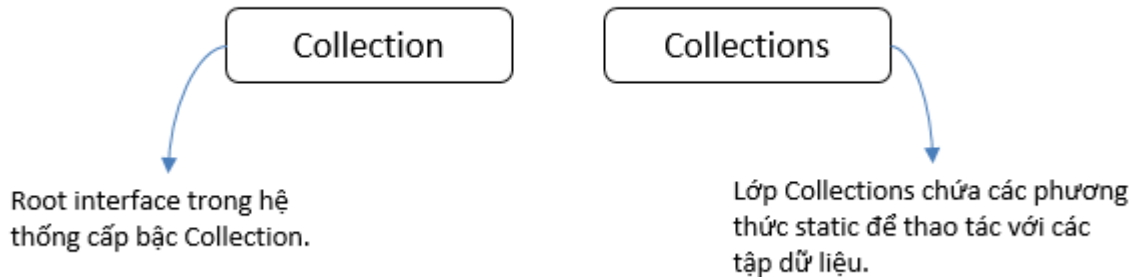


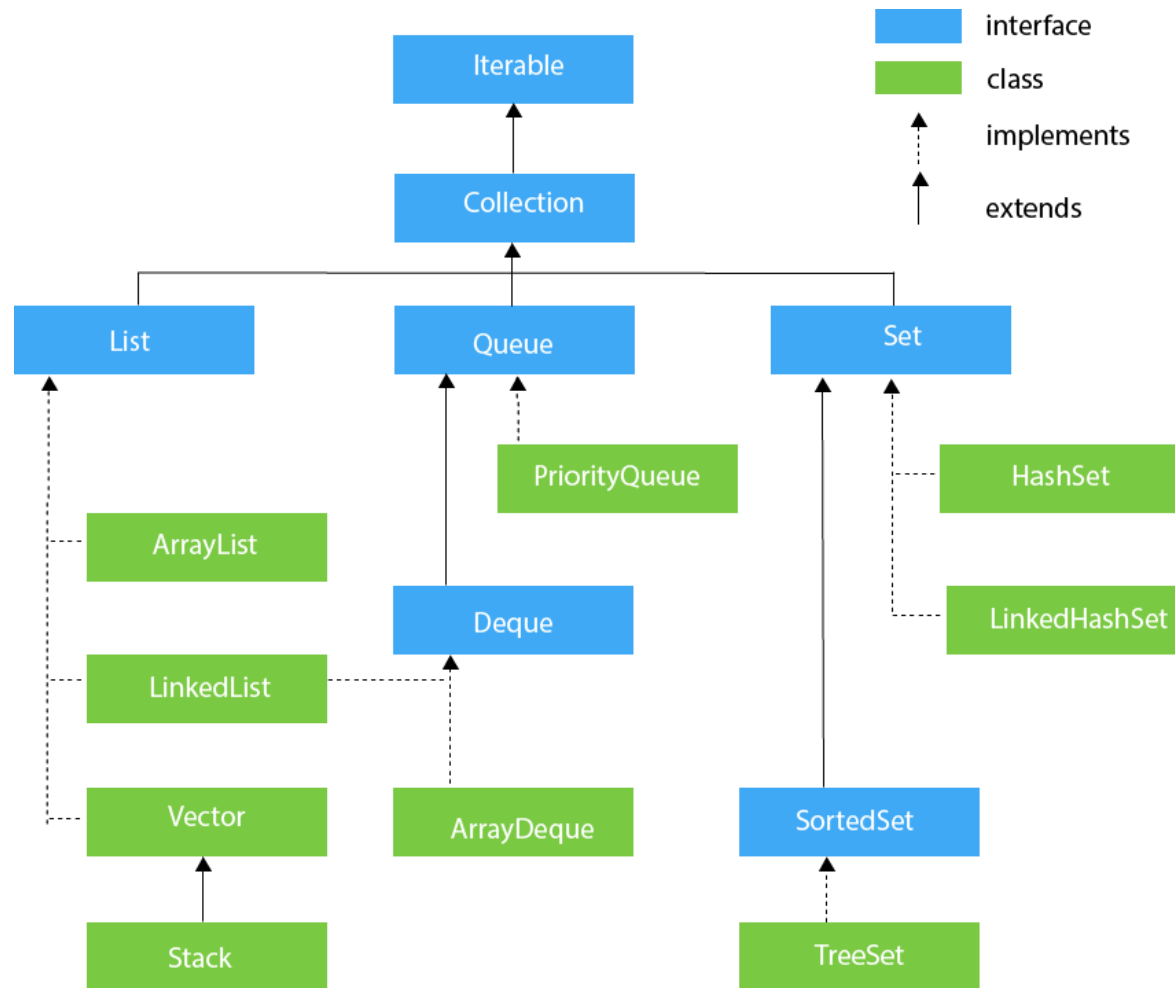
Collection là một framework cung cấp một kiến trúc để lưu trữ và thao tác với nhóm các đối tượng

Java collections có thể đạt được tất cả các thao tác mà bạn thực hiện trên dữ liệu như tìm kiếm, sắp xếp, chèn, xóa

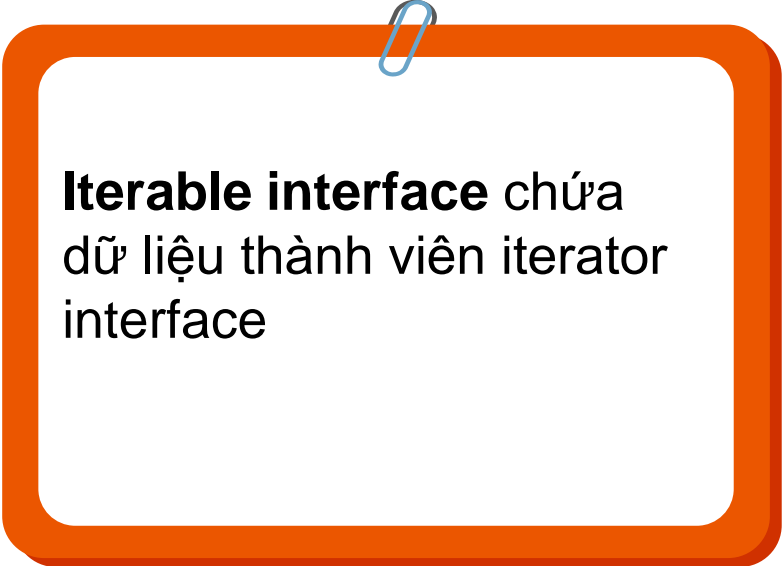
Collection là gì?

Java collection cung cấp nhiều interface (Set, List, Queue, Deque vv) và các lớp (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet,...)

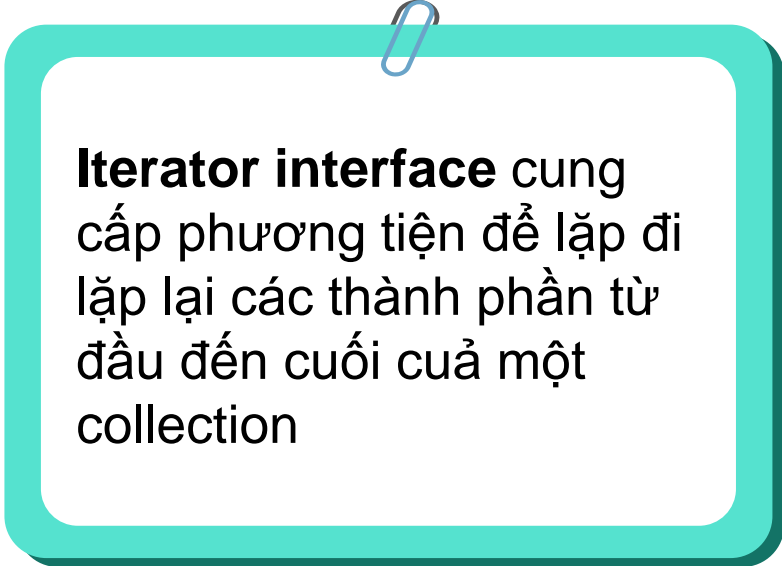




Iterable interface và iterator interface

An orange rounded rectangular box with a blue paperclip icon at the top center.

Iterable interface chứa dữ liệu thành viên iterator interface

A teal rounded rectangular box with a blue paperclip icon at the top center.

Iterator interface cung cấp phương tiện để lặp đi lặp lại các thành phần từ đầu đến cuối của một collection

Các phương thức của iterator interface

Phương thức	Mô tả
public boolean hasNext()	Trả về giá trị true nếu iterator còn phần tử kế tiếp đang duyệt
public object next()	Trả về phần tử hiện tại và di chuyển con trỏ tới phần tử tiếp theo
public void remove()	Loại bỏ phần tử cuối được trả về bởi iterator

Collection interface

Collection interface được thực hiện bởi tất cả các lớp trong Collection Framework. Nói cách khác, Collection interface là nền tảng mà Collection Framework phụ thuộc vào nó.

List Interface

List Interface là giao diện con của Collection Interface. Nó ngăn cách cấu trúc dữ liệu kiểu danh sách trong đó chúng ta có thể lưu trữ tập hợp các đối tượng có thứ tự.

List Interface được thực hiện bởi các lớp ArrayList, LinkedList, Vector và Stack

Để khởi tạo List interface chúng ta sử dụng:

```
List <Kiểu dữ liệu> <Tên>= new ArrayList();  
List <Kiểu dữ liệu> <Tên> = new LinkedList();  
List <Kiểu dữ liệu> <Tên> = new Vector();  
List <Kiểu dữ liệu> <Tên> = new Stack();
```

Set Interface

Set là kiểu dữ liệu mà bên trong nó mỗi phần tử chỉ xuất hiện duy nhất một lần và Set interface cung cấp các phương thức để thao tác với set

Set interface được kế thừa từ Collection Interface nên nó được cung cấp đầy đủ các phương thức của Collection Interface



Set Interface

Một số class thực thi Set Interface thường gặp:

- **TreeSet**: là 1 class thực thi giao diện Set Interface, trong đó các phần tử trong set đã được sắp xếp.
- **HashSet**: là 1 class implement Set Interface, mà các phần tử được lưu trữ dưới dạng bảng băm (hash table).
- **EnumSet**: là 1 class dạng set như 2 class ở trên, tuy nhiên khác với 2 class trên là các phần tử trong set là các enum chứ không phải object.

Queue Interface

Queue(Hàng đợi) là kiểu dữ liệu nổi tiếng với kiểu vào ra FIFO, tuy nhiên với Queue Interface thì queue không chỉ còn dừng lại ở mức đơn giản như vậy mà nó cung cấp cho bạn các phương thức để xây dựng các queue phức tạp hơn nhiều như priority queue, deque. Queue Interface cũng kế thừa và mang đầy đủ các phương thức từ Collection Interface.

- **LinkedList**: chính là LinkedList mình đã nói ở phần List
- **PriorityQueue**: là 1 dạng queue mà trong đó các phần tử trong queue sẽ được sắp xếp.
- **ArrayDeque**: là 1 dạng deque (queue 2 chiều) được implement dựa trên mảng

Map Interface

Map (đồ thị/ánh xạ) là kiểu dữ liệu cho phép ta quản lý dữ liệu theo dạng cặp key-value, trong đó key là duy nhất và tương ứng với 1 key là một giá trị value. Không giống như các interface ở trên, Map Interface không kế thừa từ Collection Interface mà đây là 1 interface độc lập với các phương thức của riêng mình.

Class về Map

TreeMap

EnumMap

Map

HashMap

WeakHashMap

