CodeLamGi @kieunn Follow
519 39 22

Published Jul 31st, 2017 11:29 PM
1.7K 5 0

TABLE OF CONTENTS

Have no Table of contents

SUGGESTED ORGANIZATIONS

Mino
2 2 3

Viblo
1 11 23

Emulate Infotech Pvt. Ltd
0 1 1

+1

Java Generic

- Generic programming là cách tạo ra đoạn mã có reusable (tính tái sử dụng cao), nó rất hữu ích cho những người viết software libraries (thư viện phần mềm) làm sao để generic programming (lập trình có tính tổng quát) vì nó cho phép người dùng

VIBLO Posts Questions Discussions Search Viblo Sign In/Sign up

+1

- Sau đây là cách tạo ra Generic Class và Generic Method (đại khái là Lớp chung chung và Method chung chung có thể sử dụng cho các đối tượng khác nhau). **1. Cách tạo và sử dụng Generic Class**

- Các bạn đã từng dùng ArrayList<T> đây là Collection có thể là tập hợp của đối tượng bất kỳ ví dụ: ArrayList<String> hoặc ArrayList<Student> hoặc ArrayList<Object> , ...v...v... Như các bạn đã thấy cách tạo ra lớp Generic Class ArrayList<T> có thể sử dụng cho các đối tượng khác nhau và ở đây mình sẽ nói cho các bạn cách tạo ra và sử dụng nó giống như trên.

- Để dễ hiểu lấy 1 ví dụ minh họa:

- Tôi tạo ra một lớp để chứa các chuỗi String nó được sắp xếp lần lượt. Có 3 phương thức enqueue thêm vào 1 String vào, dequeue xóa bỏ String và isEmpty kiểm tra

có String nào trong danh sách ko.

```
class QueueOfStrings {
   private LinkedList<String> items = new LinkedList<String>();
   public void enqueue(String item) {
      items.addLast(item);
   }
   public String dequeue() {
   public boolean isEmpty() {
      return (items.size() == 0);
   }
}
```


Dùng như này à

```
QueueOfStrings qOfString = new QueueOfStrings();
qOfString.enqueue(new String());
```

- Bằng cách tạo class như này tôi chỉ có thể chứa được String, giờ tôi muốn chứa kiểu Double hoặc kiểu đối tượng khác thì sao, lại viết mới 1 class cụ thể cho dạng đó à. Không, tôi sẽ tạo ra class tương tự như trên nhưng kiểu dữ liệu sẽ để dạng động (nghĩa là nó có thể chứa bất cứ thứ gì)

```
class Queue<T> {
   private LinkedList<T> items = new LinkedList<T>();
   public void enqueue(T item) {
      items.addLast(item);
   }
   public T dequeue() {
      return items.removeFirst();
   }
   public boolean isEmpty() {
```


- T là (type parameters) đại diện cho đối tượng bạn truyền vào để lưu trữ, T là ký tự bất kỳ, bạn có thể thay thế bằng 1 từ khác ví dụ như class Queue<ItemType> {...}

- Cách sử dụng đa năng hơn nhiều rồi, tôi có thể tạo ra Queue chứa String hoặc Int hoặc .... đối tượng khác nhau.

```
Queue<String> qOfString = new Queue<>();
qOfString.enqueue(new String("1"));
```

```
qOfString.dequeue(); //Remove object first
qOfString.isEmpty(); //Check list is empty?

Queue<Double> qOfDouble = new Queue<>();
qOfDouble.enqueue(12.123);
qOfDouble.dequeue();
qOfDouble.isEmpty();
```

- Có thể tạo ra 2 hoặc nhiều type parameters như sau:

```
class Pair<T,S> {
   public T first;
   public S second;
   public Pair( T a, S b ) {  // Constructor.
```



```
}

Pair<String,Color> colorName = new Pair<String,Color>("Red", Color.RED);
Pair<Double,Double> coordinates = new Pair<Double,Double>(17.3,42.8);
```


2. Cách tạo ra Generic Method

- Cũng giống như **Collection.Sort()** có thể sắp xếp được bất cứ Object đối tượng nào khác thì ta cũng tạo ra phương thức có thể dùng cho các đối tượng khác nhau.

- Ví dụ phương thức **countOccurrences** đếm số lần xuất hiện chuỗi String stringPatern có trong mảng String[] listString như sau:

```
/**
* Trả về số stringPatern có trong danh sách listString
*/
public static int countOccurrences(String[] listString, String stringPatern) {
   int count = 0;
   if (stringPatern== null) {
      for ( String s: listString)
         if (s== null)
            count++;
   }
   else {
```



```
            count++;
   }
   return count;
}
```

- Với cách viết method như trên ta chỉ có thể truyền tham số mảng String và 1 chuỗi String để tìm kiếm chuỗi String trong mảng String[] đó. Nếu áp dụng phương thức trên để tìm kiếm kiểu số nguyên Int trong mảng Int[] là không thể.

- Giờ cách viết tổng quát method trên có thể áp dụng cho mọi đối tượng như sau:

```
public static <T> int countOccurrences(T[] list, T itemToCount) {
   int count = 0;
   if (itemToCount == null) {
      for ( T listItem : list )
         if (listItem == null)
            count++;
   }
   else {
      for ( T listItem : list )
         if (itemToCount.equals(listItem))
            count++;
   }
   return count;
}
```



```
//palette is a variable of type Color[] and color is a variable of type Color
int ct = countOccurrences( palette, color );
//numbers is a variable of type Integer[]
int ct = countOccurrences( numbers, 17 );
```


T (type parameter) có thể thay bằng tên khác T phải đặt trước giá trị trả về của phương thức

- Một ví dụ khác:

```
public static <T> int countOccurrences(Collection<T> collection, T itemToCount) {
   int count = 0;
   if (itemToCount == null) {
      for ( T item : collection )
         if (item == null)
            count++;
   }
   else {
      for ( T item : collection )
         if (itemToCount.equals(item))
            count++;
   }
   return count;
}
```

LinkedList of JButtons, ...v..v..

+1

3.Ký tự đại diện <?>

- Xét ví dụ sau:

```
private boolean checkEquals(RestricExample<T> e) {
 if(number.doubleValue() == e.number.doubleValue()) {
   return true;
 }
 return false;
}

 private boolean checkEquals2(RestricExample<?> e) {
  if(number.doubleValue() == e.number.doubleValue()) {
   return true;
  }
  return false;
 }
```

- Phương thức checkEquals chỉ chấp nhận tham số e có kiểu dữ liệu giống với biến number. Phương thức checkEquals2 chấp nhận tham số e có kiểu dữ liệu khác với biến number.

VIBLO Posts Questions Discussions Search Viblo Sign In/Sign up

+1

- Ví dụ:

```
public void processElement(List<? extends A> elements){
 ...
}
```

Trong đó:

- Sử dụng phương thức processElement (Chấp nhận bất kỳ đối tượng nào miễn là đối tượng này phải kế thừa từ lớp A hoặc đối tượng của A => ClassA, ClassB và ClassC)

```
List<A> listA = new ArrayList<A>();
processElement(listA);
```



```
List<C> listC = new ArrayList<C>();
processElement(listC);
```


- Ký tự đại diện <? super type> chấp nhận bất kỳ đối tượng nào miễn là đối tượng này là cha của type hoặc đối tượng của type

Ví dụ:

```
public void processElement(List<? super A> elements){
 ...
}
List<A> listA = new ArrayList<A>();
processElement(listA);

List<Object> listO = new ArrayList<Object>();
processElement(listO);
```

Generic trong java – Giới hạn kiểu dữ liệu với ký tự đại diện <? extends type>

- Có một số trường hợp chúng ta muốn hạn chế kiểu dữ liệu của các tham số. Chẳng hạn tạo phương thức chỉ chấp nhận tham số với kiểu dữ liệu là số nguyên hoặc số thực.


```
private T number;

public RestricExample(T number) {
 this.number = number;
}

public double reciprocal() {
  return 1/number.doubleValue();
}

public static void main(String[] args) {
  RestricExample<Integer> n1 = new RestricExample<Integer>(5);
  System.out.println("Reciprocal: " + n1.reciprocal());
```

```
    RestricExample<Double> n2 = new RestricExample<Double>(7.5);
    System.out.println("Reciprocal: " + n2.reciprocal());

    //error
    //RestricExample<String> s1 = new RestricExample<String>("Hello");
  }
}
```

Related

Tran Anh Vu
789 4 0 3

Java
Do Khanh Toan
629 4 0 2

extends
Ngoc Nguyen
178 1 0 0

Ngo Dinh Ngoc
1036 1 1 1

More from CodeLamGi

Nói tạm biệt với NullPointerException trong...
CodeLamGi
1170 1 2 -1

Giới thiệu Realm – Giải pháp thay thế cho SQLite
CodeLamGi
453 0 0 0

Kotlin – Phiên bản nâng cấp của Java
CodeLamGi
469 1 0 2

Fragment và cơ chế BackStack và sử dụng...
CodeLamGi
2706 1 2 4

Comments

Login to comment

minh_dai @minh_dai — Apr 27th, 2018 9:06 AM

Cho em hỏi là : public static <T> int countOccurrences(Collection<T> collection, T itemToCount) . Thì <T> là có phải là kiểu trả về không ạ, em thấy có int rồi thì tại sao còn cần <T> ạ, hay nó có nghĩ là mình có thể trả về một kiểu khác ngoài kiểu int đã định sẵn ??? em cảm ơn

0 | Reply Share

VIBLO Posts Questions Discussions Search Viblo Sign In/Sign up

0 | Reply Share

minh_dai @minh_dai — Jul 13th, 2018 6:16 PM

tham số kiểu T ? thế nếu tham số truyền vào là Collection<G> collection, T itemToCount . Thì sẽ chuyển thành <T,G> hả anh ??

0 | Reply Share

@minh_dai hiểu sai vấn đề rồi !! Ví dụ minh họa : public static <G,T> int countOccurrences(Collection<G> collection, T itemToCount)

0 | Reply Share

Bang Dinh @bangdinh95 Dec 27th, 2018 3:13 PM

public static <G,T,H,K,L,M> int countOccurrences(Collection<G> collection, T itemToCount, H h, K k, L l, M m){ [.....] return 0; }

0 | Reply Share