Java泛型类似c++的模版

public class GenericTest {

public static void main(String[] args) {

List list = new ArrayList();

list.add("qqyumidi");

list.add("corn");

list.add(100);

for (int i = 0; i < list.size(); i++) {

String name = (String) list.get(i); // 1

System.out.println("name:" + name);

}

}

}

**泛型，即“参数化类型”。一提到参数，最熟悉的就是定义方法时有形参，然后调用此方法时传递实参。那么参数化类型怎么理解呢？顾名思义，就是将类型由原来的具体的类型参数化，类似于方法中的变量参数，此时类型也定义成参数形式（可以称之为类型形参），然后在使用/调用时传入具体的类型（类型实参）。**

public class GenericTest {

public static void main(String[] args) {

/\*

List list = new ArrayList();

list.add("qqyumidi");

list.add("corn");

list.add(100);

\*/

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

list.add("qqyumidi");

list.add("corn");

//list.add(100); // 1 提示编译错误

for (int i = 0; i < list.size(); i++) {

String name = list.get(i); // 2

System.out.println("name:" + name);

}

}

}

采用泛型写法后，在//1处想加入一个Integer类型的对象时会出现编译错误，通过List<String>，直接限定了list集合中只能含有String类型的元素，从而在//2处无须进行强制类型转换，因为此时，集合能够记住元素的类型信息，编译器已经能够确认它是String类型了。

Java中的泛型这一概念提出的目的，导致其只是作用于代码编译阶段，在编译过程中，对于正确检验泛型结果后，会将泛型的相关信息擦出，也就是说，成功编译过后的class文件中是不包含任何泛型信息的。泛型信息不会进入到运行时阶段。

**泛型类型在逻辑上看以看成是多个不同的类型，实际上都是相同的基本类型。**

**List 是一个接口，它代表着有序的队列。**

**List继承了Collection，是有序的列表。**

**实现类有ArrayList、LinkedList、Vector、Stack等**

**ArrayList是基于数组实现的，是一个数组队列。可以动态的增加容量！**

**LinkedList是基于链表实现的，是一个双向循环列表。可以被当做堆栈使用！**

**Vector是基于数组实现的，是一个矢量队列，是线程安全的！**

**Stack是基于数组实现的，是栈，它继承与Vector，特性是FILO（先进后出）！**

**二.使用场景**

**在实际的应用中如果使用到队列，栈，链表，首先可以想到使用List。不同的场景下面使用不同的工具，效率才能更高！**

**1. 当集合中对插入元素数据的速度要求不高，但是要求快速访问元素数据，则使用ArrayList！**

**2. 当集合中对访问元素数据速度不做要求不高，但是对插入和删除元素数据速度要求高的情况，则使用LinkedList！**

**3.当集合中有多线程对集合元素进行操作时候，则使用Vector！但是现在BVector现在一般不再使用，如需在多线程下使用，可以用CopyOnWriteArrayList，在java.util.concurrent包下。**

**4.当集合中有需求是希望后保存的数据先读取出来，则使用Stack！**

使用以下代码对ArrayList和LinkedList中的几种主要操作所用时间进行对比：

ArrayList<Integer> arrayList = new ArrayList<Integer>();

LinkedList<Integer> linkedList = new LinkedList<Integer>();

// ArrayList add

long startTime = System.nanoTime();

for (int i = 0; i < 100000; i++) {

    arrayList.add(i);

}

long endTime = System.nanoTime();

long duration = endTime - startTime;

System.out.println("ArrayList add:  " + duration);

// LinkedList add

startTime = System.nanoTime();

for (int i = 0; i < 100000; i++) {

    linkedList.add(i);

}

endTime = System.nanoTime();

duration = endTime - startTime;

System.out.println("LinkedList add: " + duration);

// ArrayList get

startTime = System.nanoTime();

for (int i = 0; i < 10000; i++) {

    arrayList.get(i);

}

endTime = System.nanoTime();

duration = endTime - startTime;

System.out.println("ArrayList get:  " + duration);

// LinkedList get

startTime = System.nanoTime();

for (int i = 0; i < 10000; i++) {

    linkedList.get(i);

}

endTime = System.nanoTime();

duration = endTime - startTime;

System.out.println("LinkedList get: " + duration);

// ArrayList remove

startTime = System.nanoTime();

for (int i = 9999; i >=0; i--) {

    arrayList.remove(i);

}

endTime = System.nanoTime();

duration = endTime - startTime;

System.out.println("ArrayList remove:  " + duration);

// LinkedList remove

startTime = System.nanoTime();

for (int i = 9999; i >=0; i--) {

    linkedList.remove(i);

}

endTime = System.nanoTime();

duration = endTime - startTime;

System.out.println("LinkedList remove: " + duration);

结果：

ArrayList add:  13265642

LinkedList add: 9550057

ArrayList get:  1543352

LinkedList get: 85085551

ArrayList remove:  199961301

LinkedList remove: 85768810