

原CS61B sp2018笔记 | Generics and Autoboxing

2018年07月29日 15:57:00 隐秀\_ 阅读数： 80 更多

1. Automatic Conversions

1.1 Autoboxing and Unboxing

Java中的泛型用到了 `<>`，当我们实例化一个泛型的时候，必须要指明一个确定的类型。

回忆一下，Java有8中基本类型，其他所有的类型都是引用类型。Java的一个特点就是不能把基础类型当作参数传递给泛型，比如说，`ArrayDeque<int>` 法错误，正确的写法是 `ArrayDeque<Integer>`。

对于每一种基本类型，都有一种引用类型与之对应，这些引用类型称为 **包装类(wrapper classes)**。

Primitive	Class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Doube
boolean	Boolean
char	Character

我们假定基本类型和包装类之间没有类型转换，这时如果我们要使用一个泛型的数据结构，就不得不这样做：

```
1 public class BasicArrayList {
2     public static void main(String[] args) {
3         ArrayList<Integer> L = new ArrayList<Integer>();
4         L.add(new Integer(5));
5         L.add(new Integer(6));
6
7         /* Use the Integer.valueOf method to convert to int */
8         int first = L.get(0).valueOf();
9     }
10 }
```

幸运的是，Java为基本类型和包装类之间提供了隐式的转换，所以我们可以写出这样的代码：

```
1 public class BasicArrayList {
2     public static void main(String[] args) {
3         ArrayList<Integer> L = new ArrayList<Integer>();
4         L.add(5);
5         L.add(6);
6         int first = L.get(0);
7     }
8 }
```

这种转换称为 **box** 和 **unbox**，也就是 **Auto-box**（自动装箱功能）。

对自动装箱功能有几个需要注意的地方：

- 数组没有自动装箱功能，比如，你有一个整数类型的数组 `int[] x`，它将不会自动转换成 `Integer[]` 类型。
- 自动装箱功能会减慢程序运行的速度。

## 1.2 Widening

还有一种自动转换是Java基本类型之间的转换，内存小的类型会自动转换成内存大的类型，比如说我们有这样一个函数：

```
1 public static void blahDouble(double x) {
2     System.out.println("double: " + x);
3 }
```

我们可以这样调用它：

```
1 int x = 20;
2 blahDouble(x);
```

不过要把内存大的类型转换成内存小的类型，就必须进行强制类型转换：

```
1 public static void blahInt(int x) {
2     System.out.println("int: " + x);
3 }
```

我们需要这样调用它：

```
1 double x = 20;
2 blahInt((int) x);
```

## 2. Immutability

数据类型分为 **可变量型(mutable)** 和 **不变型(immutable)**。

一个不变的数据类型它的 **实例** 任何情况下都不会发生改变。比如说，Java中的 **String** 类型就是不变的，一旦字符串被确定，它就再也不会改变，即使一些改变，也要返回一个新的字符串。

数据类型的不变性使用 **final** 关键字定义，比如说下面这个Date类：

```
1 public class Date {
2     public final int month;
3     public final int day;
4     public final int year;
5     private boolean contrived = true;
6     public Date(int m, int d, int y) {
7         month = m; day = d; year = y;
8     }
9 }
```

当实例化一个 **Date** 类后，就无法再改变它的属性了。

我们再来看一个特殊的例子：

```
1 public final ArrayDeque<String>() deque = new ArrayDeque<String>();
```

变量deque被定义为**final**代表它不会进行再次分配，但是我们知道deque里面存储的是引用，引用不能改变，但是引用所指向内存中的内容是可以改变的。**ArrayDeque永远都是可以改变的**。

## 3. Generics

### 3.1 Creating Another Generic Class

在此之前，我们已经创建了一些泛型链表，下面让我们创建一种新的数据类型——**maps**。

我们将建立 **ArrayMap** 类，它实现 **Map61B** 接口，这个接口目前有几个基本方法：

```
1 put(key, value): Associate key with value.
2 containsKey(key): Checks if map contains the key.
3 get(key): Returns value, assuming key exists.
4 keys(): Returns a list of all keys.
5 size(): Returns number of keys.
```

为了快速建立模型，我们忽略内存的重新分配问题，需要说明一下，我们的类中 **一个key只能对应一个value**，下面是它的实现：

```
1 package Map61B;
2
3 import java.util.List;
4 import java.util.ArrayList;
5
6 /**
7  * An array-based implementation of Map61B.
8  */
9 public class ArrayMap<K, V> implements Map61B<K, V> {
10
11     private K[] keys;
12     private V[] values;
13     int size;
14
15     public ArrayMap() {
16         keys = (K[]) new Object[100];
17         values = (V[]) new Object[100];
18         size = 0;
19     }
20
21     /**
22      * Returns the index of the key, if it exists. Otherwise returns -1.
23      */
24     private int keyIndex(K key) {
25         for (int i = 0; i < size; i++) {
26             if (keys[i].equals(key)) {
27                 return i;
28             }
29         }
30         return -1;
31     }
32
33     public boolean containsKey(K key) {
34         int index = keyIndex(key);
35         return index > -1;
36     }
37
38     public void put(K key, V value) {
39         int index = keyIndex(key);
40         if (index == -1) {
41             keys[size] = key;
42             values[size] = value;
43             size += 1;
44         } else {
45             values[index] = value;
46         }
47     }
48
49     public V get(K key) {
50         int index = keyIndex(key);
51         return values[index];
52     }
53
54     public int size() {
55         return size;
56     }
57
58     public List<K> keys() {
59         List<K> keyList = new ArrayList<>();
60         for (int i = 0; i != size; i++) {
61             keyList.add(keys[i]);
62         }
63         return keyList;
64     }
65 }
```

这个map基于array数组，它不是最好的，但是可以用来说明一些问题。

我们写一个测试类来测试一下：

```

1 @Test
2 public void test() {
3     ArrayMap<Integer, Integer> am = new ArrayMap<Integer, Integer>();
4     am.put(2, 5);
5     int expected = 5;
6     assertEquals(expected, am.get(2));
7 }

```

你将会发现报了一个运行时错误：

```

$ javac ArrayMapTest.java
ArrayMapTest.java:11: error: reference to assertEquals is ambiguous
assertEquals(expected, am.get(2));
^
both method assertEquals(long, long) in Assert and method assertEquals(Object, Object) in Assert match

```

错误的原因是JUnit的 `assertEquals` 方法重载了，我们只需要把最后一行代码改为 `assertEquals((Integer)expected, am.get(2));`，  
`assertEquals(Object, Object)` 方法即可。

用的是

## 3.2 Generic Methods

接下来我们将建立一个新的类 `MapHelper`，它有以下两个方法：

- `get(Map61B, key)`: 根据给出的key值返回Map61B中对应的value，如果key不存在的话，返回null
- `maxKey(Map61B)`: 返回Map61B中key的最大值。

### 3.2.1 Implementing get

在这个类中，我们没有在类名的后边声明这个类是一个泛型，而是在动议每个方法时声明，所以get方法就会是这样：

```

1 public static <K,V> V get(Map61B<K,V> map, K key) {
2     if map.containsKey(key) {
3         return map.get(key);
4     }
5     return null;
6 }

```

我们可以这样使用它：

```

1 ArrayMap<Integer, String> isMap = new ArrayMap<Integer, String>();
2 System.out.println(mapHelper.get(isMap, 5));

```

### 3.2.1 Implementing maxKey

一开始实现这个方法的时候你可能会这样写：

```

1 public static <K, V> K maxKey(Map61B<K, V> map) {
2     List<K> keylist = map.keys();
3     K largest = map.get(0);
4     for (K k: keylist) {
5         if (k > largest) {
6             largest = k;
7         }
8     }
9     return largest;
10 }

```

不过很快你将发现一些问题，K类型之间的比较发生了错误，这就意味着我们要考虑对象的比较规则，所以要实现 `Comparable` 接口：

```

1 public static <K extends Comparable<K>, V> K maxKey(Map61B<K, V> map) {
2     List<K> keylist = map.keys();
3     K largest = map.get(0);
4     for (K k: keylist) {
5         if (k.compareTo(largest)) {
6             largest = k;
7         }
8     }
9     return largest;
10 }

```

```
8     }
9     return largest;
10 }
```

可能你对这个函数的定义有些困惑，特别是返回类型 `<K extends Comparable<K>, V> K` 这一部分。

首先 `Comparable` 接口本身就是一个范式的接口，所以我们必须指明我们要比较的类型，也就是在`Comparable`后面加 `<K>`。

那么为什么不用 `implements` 反而使用 `extends` 呢？其实，这里的`extends`有了不一样的含义。当我们说 `K extends Comparable` 时，仅仅意味着K必须 `extends`不同种的使用方法称为 `type upper bounding`。

CS61B sp2018笔记 | Introduction to Java

Introduction to Java Essentials 1. Reading 1.1 Hello World 这门课程虽然是用Java教授数据结构，但重点不在Ja...

想对作者说点什么？ 

我来说一句

**(原创笔记) 加州伯克利大学CS61b数据结构 (Java描述) 一：对象** 👁 2326  
OOP(object-oriented programming): object: a repository of data; class: type of object.... 来自: [Emacsor的博客](#)

**CS61B sp2018笔记 | Lists** 👁 188  
Lists 1. IntLists 下面我们来一步一步的实现List类，首先你可以实现一个最简单的... 来自: [隐秀\\_](#)

**final review for berkeley cs61b**  
关于伯克利公开课cs61b的期末复习文档



**招聘海归硕士**

百度广告

**CS61B sp2018笔记 | Inheritance, Implements** 👁 47  
1. Intro and iterfaces 1.1 The Problem 回想我们之前写过的两个类SLList和AList, ... 来自: [隐秀\\_](#)

**CS61B sp2018笔记 | Testing and Selection Sort** 👁 40  
调试程序可以说是最能体现程序员水平的能力之一了，接下来我们将讨论如何写t... 来自: [隐秀\\_](#)

**CS61B sp2018笔记 | Exceptions, Iterators, Iterables** 👁 41  
1. Throwing and catching 1.1 Throwing Exceptions 在程序运行过程中，可能会遇到某... 来自: [隐秀\\_](#)

**CS61B Homework5作业回顾 (附代码)** 👁 408  
HW5一开始做不出来的同学可以继续先看Lecture，Encapsulated List那一节老师会回... 来自: [everest115的博客](#)

**自动装箱 (Autoboxing) 和自动拆箱 (AutoUnboxing)** 👁 1014  
JDK1.5提供了自动装箱 (Autoboxing) 和自动拆箱 (AutoUnboxing) 功能。所谓自... 来自: [zlx18225318697...](#)

**股市第三次机会来了！不看就亏大了！**  
唐煌投资 · 熯熯

**CS61B Homework6作业回顾 (附代码)** 👁 92  
本次作业重点是hascode和compress function，评价一个compress function好坏的标... 来自: [everest115的博客](#)

**【Java】泛型 (Generics)** 👁 518  
What 顾名思义，泛型：一般类型，也就是说可以为任何类型，泛型的本质是“参... 来自: [Just do it!](#)



**Danny\_姜**

[关注](#) 151篇文章



**coffeecato**

[关注](#) 122篇文章



**vincent\_chan7**

[关注](#) 11篇文章

**CS61BHomework3作业回顾 (附代码)** 👁 127  
Part 1.smoosh()函数，目的是将数组中出现的所有数字产生重复的部分删除，例如11... 来自: [everest115的博客](#)

CS61B sp2018笔记 | Efficient Programming

Efficient Programming "An engineer will do for a dime what any fool will do for a dolla...

12

来自: 隐秀\_

c#2.0新特性: (一) 泛型 (Generics)

为了提高应用程序的效率和可用性, C#2.0引入了泛型概念.C#泛型有些类似C++的摸...

780

来自: 十二随风

股市第三次机会来了! 不看就亏大了!

唐煌投资 · 熯熯

下载

Generics\_in\_the\_Java\_Programming\_Language.pdf

Generics\_in\_the\_Java\_Programming\_Language, 英文原文

11-21

Java 性能要点: 自动装箱/ 拆箱 (Autoboxing / Unboxing)

本文作者为 Ali Kemal TASCI, 最早于2016年4月9日发布于DZONE社区。文章主要介... 来自: wangpeng19868...

888

伯克利计算机低年级核心课程之CS61A-SICP

http://bbs.eol.cn/forum.php?mod=viewthread&extra=page%3D1&tid=2391800 给想... 来自: fengjiexyb的专栏

8775

伯克利大学数据结构课程

http://datastructur.es/sp17/ CS61B-Data Structures UC berkeley 来自: yaoxiaofeng\_000...

104

Android 性能优化——小心自动装箱 (Autoboxing)

小心自动装箱(Autoboxing) 有时性能瓶颈是由小问题累积到一起产生的。一个典型例...

761

来自: 四季逗

 别再用MongoDB了

百度广告

C++实现控制台版2048 (内附彩蛋)

前言 之前做过一个JavaScript版本的2048游戏, 最近在学习C++, 昨天晚上突然... 来自: 隐秀\_

124

CS61B Homework7作业回顾(附代码)

发现编程真的是很强调逻辑。首先Track down the whole tree, split 3-key node into 2 ... 来自: everest115的博客

36

CS61B Homework9作业回顾 (附代码)

本次作业主要学会运用Disjoin Sets, DisjointSets的class作业已经写好给出。class中... 来自: everest115的博客

57

cs61a课程总结--lecture7 递归 (和一种数据结构)

数据结构 tuple 是一种至类型 来自: vczhfan的专栏

520

下载

Data Structures Lecture Notes (UCB CS61b)

Data Structures Lecture Notes (UCB CS61b) Data Structures Lecture Notes (UCB CS61b)

10-20

老中医说: 男人多吃它, 性生活时间延长5倍

新方向 · 熯熯

多态 (polymorphism) --清楚解释了为什么重载 (overload) 不是多态

来源: csdn论坛帖子多态性, 这个面向对象编程... 来自: 把思想充满在语...

439

JavaGenericsFAQ 目录---Java泛型的知识, 这一篇就够了

本文只是将JavaGenericsFAQ的目录列在了这里, 具体请见: http://www.angelikalan... 来自: coolaaron的专栏

187

一场由Java堆污染(Heap Pollution)引发的思考

1 Kotlin的数组比Java的更加安全, 可以避免Heap Pollution 2 Kotlin代码比Java更加... 来自: xzm317122667...

1642

CS61A 系列课程笔记 (一)

嗯 今天刚看第二周的课程, 大量的 reading 材料我是真的要崩溃了, 全英。。。我... 来自: 柠檬黄先生的博客

1338

java7新特性 当使用可变并且非具体类型形式化参数的方法时候，改进警告...

5034

原文 本页涵盖以下主题： Heap Pollution带可变参数方法与非具体化参数安全漏洞可... 来自： [sixianfeng的专栏](#)



招聘海归硕士

百度广告

Generics

[code=java]package com.my.example;rn rn class A implements Runnablern rn @Overridern public void run() rn rn rn...

unboxing and autoboxing

Integer i="aaa";rn rn Integer i=integer.valueOf("aaa");rn rn jdk1.5 以后提供了自动拆箱与自动装箱功能，高手们提供下...

自动装箱 autoBoxing

1.8万

自动装箱是java 1.5 引入的新技术，主要是为了解决 原始类型和对象的转换， 也就是... 来自： [勿在浮沙筑高台](#)

下载

A Java Reference: Assorted Java Reference Material

10-08

UC Berkeley CS61B Data Structure推荐阅读 Fall 2018

下载

CS61B java intro

03-31

berkeley cs61b java introduction，对初学者很有帮助

靠死工资怎么买房买车，他们是这样赚钱的...

虹服 · 嬉游

TypeScript迅速入门与应该知道

3228

TypeScript应该知道 简述： TypeScript基础知识。 链接TypeScript 官网： http://www.t... 来自： [Blog for 明月依稀](#)

基本数据类型的 autoboxing

86

// 测auto package 对于基本数据类型的 指定cache public static void main(String[] args...

来自： [abcdef](#)

自动装箱 (autoboxing)

59

某些时候，主数据类型无法使用，你必须使用对象（比如ArrayList）所以需要装箱操... 来自： [wYuQi的博客](#)

关于autoboxing

自学JAVA小白一枚，望大神指点rn public class test4 rn rn Integer i;rn rn int j;rn rn public static void main(String[] arg...

Autoboxing（自动装箱）

1098

As any Java programmer knows, you can't put an int (or other primitive value) into a c... 来自： [KNIGHTRCOM\(r...](#)

老中医说：男人多吃它，性生活时间延长5倍

新方向 · 嬉游

自动装箱 (Autoboxing)

447

前段时间考试用着了。JDK5的autoboxing。 官方解释： http://docs.oracle.com/javas... 来自： [雨打蕉叶又潇潇...](#)

Generics的用法

2

/\* \* To change this template, choose Tools | Templates \* and open the template in the ... 来自： [makao4568396...](#)

下载

【伯克利CS61B教材——Java 学习教程】 Head First Java 中文版

05-08

中文版的伯克利CS61B教材——经典Java 学习教程 《Head First Java 第二版》

泛型 (Generics)

23

作者： VieLei链接： https://blog.csdn.net/s10461/article/details/53941091来源： CS... 来自： [Frank Cui Blog](#)

泛型 (Generics)

462

C# 2.0引入了很多语言扩展，最重要的就是泛型（Generics）、匿名方法（Anonymo... 来自： [doubleyou的专栏](#)



十大笔记本电脑排行榜

百度广告

下载

【Java 学习教程】Head First Java 第二版 英文版

05-08

经典Java 学习教程——Head First Java 第二版 英文版，伯克利CS61b教材

下载

CS61B 教材1 Head First Java 2nd Edition (2005)

04-10

CS61B 教材1 Head First Java 2nd Edition (2005) 高清英文版

下载

Java Generics and Collections.pdf

09-21

Java Generics and Collections

下载

generics C#

04-13

c# generics DEMO

Java高级系列——如何使用、何时使用泛型（Generics）？

594

一、介绍 泛型的概念代表了对类型的抽象（C++开发人员熟知的模板）。它是一个非...

来自：Ron.Zheng



你的显卡在什么档位 天梯图出炉

百度广告

java的autoboxing 问题

现有如下代码，请高手解答一下原因。  
Integer i = new Integer(128);  
Integer j = new Integer(128);  
System.out...



隐秀\_

关注

原创	粉丝	喜欢	评论
26	22	24	13

等级： 博客 访问： 3万+  
积分： 503 排名： 11万+  
勋章： 1024



about

cs专业一名挣扎的后端  
热爱交友

GitHub: [github.com/seriouszyx](https://github.com/seriouszyx)  
cnblog: [cnblogs.com/henuzyx](https://cnblogs.com/henuzyx)  
独立博客: [seriouszyx.github.io](https://seriouszyx.github.io)

热门文章

最新超详细VMware虚拟机下载与安装  
阅读量：19735



最新超详细虚拟机VMware安装Kali Linux

阅读量：5596

上机考试作弊，也不是不可以呢

阅读量：2642

hexo搭建个人独立博客 | NexT主题深度美化方案

阅读量：660

JSONArray.fromObject不执行且不报错问题的解决

阅读量：431

最新文章

Tomcat 引入jar包导致的NoClassDefFoundError 报错的问题解决

Spring IoC容器浅析及简单实现

CS61B sp2018笔记 | Efficient Programming

Linux 基本操作总结（后端必备）

Java生成解析二维码

博主专栏



CS61B sp2018笔记

阅读量：514      6 篇

个人分类

C++4篇

JavaScript5篇

Linux4篇

游戏2篇

Java13篇

展开

归档

2018年10月2篇

2018年9月1篇

2018年8月4篇

2018年7月8篇

2018年6月4篇

展开

最新评论

hexo搭建个人独立博客 | Ne...  
qq\_40950957：[reply]qq\_39901385[/reply] 我用的是这个主题 Next.Gemini，我...

hexo搭建个人独立博客 | Ne...  
qq\_39901385：我知道什么问题了是没清缓存 楼主就是那个主页怎么把一篇文字折叠起来呀 用一个按钮来控制打开这个怎么...

hexo搭建个人独立博客 | Ne...  
qq\_39901385：本地能设置next的背景和头像旋转 使用了hexo g -d放在了git上 绑定了域名 但是通过g...

hexo搭建个人独立博客 | Ne...  
qq\_39901385：楼主方便加个q吗我有些hexo搭配next的主题优化的问题想问你

最新超详细VMware虚拟机下载与...

qq\_43318544：不错顶起来，现在有15了VMware Workstation Pro15虚拟机破解版： [https...](https://www.csdn.net/qq_43318544/article/details/81270547)

### 电脑配置组装



### 联系我们



扫码联系客服



下载CSDN APP

 QQ客服

 [kefu@csdn.net](mailto:kefu@csdn.net)

 客服论坛

 400-660-0108

工作时间 8:00-22:00

关于我们 招聘 广告服务 网站地图

 百度提供站内搜索 京ICP证09002463号

©2018 CSDN版权所有

网络110报警服务 经营性网站备案信息

北京互联网违法和不良信息举报中心

中国互联网举报中心