实验1 基本类型包装器

【实验目的】

1. 掌握基本类型包装器的作用
2. 掌握装箱与拆箱操作

【实验要求】

1. 编写一个Java程序，实现字符串转化为整型，在实现将整型数据包装成Integer类的对象，再将其转换为基本数据类型。。
2. 编写一个Java程序，实现自动装箱、拆箱操作。

【实验步骤】

1. 启动Eclipse软件，在出现的设置工作空间对话框中设置工作目录，进入Eclipse工作界面。
2. 在Eclipse中，选择菜单【File】|【new】|【java Project】，在弹出的【New Java Project】窗口的【Project name】选项中设置Java项目的名称为【JavaExecise】，其它默认即可，点击Finish按钮，完成Java项目创建。
3. 在新建的Java项目中，选中src文件夹，鼠标右键单击，在弹出的快捷菜单中，选择【New】|【Class】命令，在弹出的【New Java Class】窗口中的【Name】文本框中，输入Java文件的类名【TestInteger】，同时选中【public static void main(String[] args)】选项，在程序中自动生成main主方法，点击Finish按钮后，Eclipse会在Java项目的src文件目录下创建一个名称为TestInteger的java类文件。
4. 在代码编辑器中输入以下代码，并将代码补充完整

public class TestInteger {

public static void main(String[] args) {

String str="123";

//将字符串转化为整数

int a=【代码1】;

System.out.println(a);

int b=10;

//将基本类型变量b转化为Integer类型对象num

【代码2】;

//将num对象转化成基本类型

int x=【代码3】;

System.out.println("x="+x);

}

}

(5)在创建的项目【JavaExecise】中，选中src文件夹，鼠标右键单击，在弹出的快捷菜单中，选择【New】|【Class】命令，在弹出的【New Java Class】窗口中的【Name】文本框中，输入Java文件的类名【TestAutoBox】，同时选中【public static void main(String[] args)】选项，在程序中自动生成main主方法，点击Finish按钮。

(6)在代码编辑器中输入以下代码，分析结果

public class TestAutoBox {

public static void main(String[] args) {

//将整数5自动装箱为Integer对象num

【代码1】

//将num对象自动拆箱为int类型变量t

【代码2】

System.out.println(t\*num);

}

}

【实验指导】

1. Java将基本数据类型进行了包装。
2. 装箱：将基本类型转化为引用类型
3. 拆箱：将引用类型转化为基本类型
4. 基本类型和包装类型的对应关系：



实验2 集合类

【实验目的】

1. 掌握ArrayList类的使用
2. 掌握HashMap类的使用
3. 掌握迭代器的使用

【实验要求】

1. 编写一个Java程序，利用ArrayList实现元素的添加和删除，并将ArrayList集合转变为数组输出。
2. 编写一个Java程序，利用HashMap实现元素的添加和删除，并将输出元素的值。
3. 编写一个Java类，利用迭代器分别实现对ArrayList和HashMap的迭代输出。

【实验步骤】

(1) 启动Eclipse软件，在出现的设置工作空间对话框中设置工作目录，进入Eclipse工作界面。

(2) 在Eclipse中，选择菜单【File】|【new】|【java Project】，在弹出的【New Java Project】窗口的【Project name】选项中设置Java项目的名称为【JavaExecise】，其它默认即可，点击Finish按钮，完成Java项目创建。

(3) 在新建的Java项目中，选中src文件夹，鼠标右键单击，在弹出的快捷菜单中，选择【New】|【Class】命令，在弹出的【New Java Class】窗口中的【Name】文本框中，输入Java文件的类名【TestArrayList】，同时选中【public static void main(String[] args)】选项，在程序中自动生成main主方法，点击Finish按钮后，Eclipse会在Java项目的src文件目录下创建一个名称为TestArrayList的java类文件。

(4) 在代码编辑器中输入以下代码，并将代码补充完整

import java.util.ArrayList;

public class TestArrayList {

public static void main(String[] args) {

// 创建ArrayList对象list

【代码1】

//给list对象增加10个int元素，元素的值对应i的值

for( int i=0;i<10;i++ )

{

【代码2】

}

//利用for循环输出list中元素

for(int i=0;i<【代码3】;i++)

{

System.out.print(【代码4】+" ");

}

System.out.println();

//利用增强型for循环输出list中的元素

for(【代码5】)

{

System.out.print(o+" ");

}

System.out.println();

//将第6个元素移除

【代码6】

//再增加3个元素

for( int i=0;i<3;i++ )

【代码7】

//将list对象转化成整型数组

Integer[] al=【代码8】;

for(int i=0;i<al.length;i++)

System.out.print(al[i]+" ");

}

}

(5)在创建的项目【JavaExecise】中，选中src文件夹，鼠标右键单击，在弹出的快捷菜单中，选择【New】|【Class】命令，在弹出的【New Java Class】窗口中的【Name】文本框中，输入Java文件的类名【TestHashMap】，同时选中【public static void main(String[] args)】选项，在程序中自动生成main主方法，点击Finish按钮。

(6)在代码编辑器中输入以下代码，并将代码补充完整

import java.util.HashMap;

public class TestHashMap {

public static void main(String[] args) {

// 创建一个HashMap对象 map

【代码1】

//map中添加元素，key为name，value为tom

【代码2】

//map中添加元素，key为price，value为800

【代码3】

//map中添加元素，key为total，value为Integer对象，其值为1600

【代码4】

//输出map中name元素的值

System.out.println("name : "+【代码5】);

//输出map中price元素的值

System.out.println("price : "+【代码6】);

//输出map中total元素的值

System.out.println("total : "+【代码7】);

map.put("name", "jack");

System.out.println("map集合中元素的数量是："+【代码8】);

//输出map中name元素的值

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

//移除key值为name的元素

【代码9】

System.out.println("map中是否包含key值为name的元素 : "+【代码10】);

//移除所有的元素

【代码11】

System.out.println("map是否为空 : "+【代码12】);

}

}

(7)在创建的项目【JavaExecise】中，选中src文件夹，鼠标右键单击，在弹出的快捷菜单中，选择【New】|【Class】命令，在弹出的【New Java Class】窗口中的【Name】文本框中，输入Java文件的类名【TestIterate】，同时选中【public static void main(String[] args)】选项，在程序中自动生成main主方法，点击Finish按钮。

(8)在代码编辑器中输入以下代码，并将代码补充完整

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import java.util.Set;

public class TestIterate {

public static void main(String[] args) {

// 创建List对象list，指向具体的一个ArrayList

List list = 【代码1】;

list.add("one");

list.add(new Integer(4));

list.add("");

list.add(new Float(5.0F));

list.add("one"); // true

//获得迭代器iterator

Iterator iterator = 【代码2】;

//判断迭代器中是否还有元素

while (【代码3】)

{

//返回迭代器中的下一个元素的值

System.out.print(【代码4】+" ");

}

System.out.println();

// 创建Map对象map，指向具体的一个HashMap

Map map =【代码5】;

//map中添加元素，key为name，value为tom

map.put("name", "tom");

//map中添加元素，key为price，value为800

map.put("price", 800);

//map中添加元素，key为total，value为Integer对象，其值为1600

map.put("total", new Integer(1600));

//获得map中的set对象

Set entryset = 【代码6】;

//通过set对象获得迭代器

Iterator ite = 【代码7】;

//判断迭代器中是否还有元素

while (【代码8】)

{

//获得迭代器中的下一个元素，并强转成Entry类型

Entry entry =【代码9】;

//通过entry对象分别获得key值和value值

System.out.println(【代码10】+" : "+【代码11】);

}

}

}

【实验思考】

1. ArrayList与HashSet的区别是什么？
2. 思考两者的使用场合？
3. 思考为什么两者中可以存放不同类型的元素？如何限制集合中只能存放相同类型的元素？

【实验指导】

(1)Java中集合类：用来存放对象的。相当于一个容器，里面包含着一组对象，其中每个对象作为集合的一个元素出现。

(2) Collection中可放不同类型数据，是Set接口和List接口的超类

* Set是无序的集合，元素存入顺序和集合内存储的顺序不同，不允许元素重复
* List是有序的集合，允许元素重复，List中的元素都对应着一个整数型的序号，记载其在容器中的位置，可以根据序号存取容器中的元素，主要实现类有ArrayList类及LinkedList类。

1. ArrayList类：线性顺序存储，是一种线性表，可以存储重复数据。
2. ArrayList的大小可动态改变，可看成动态数组或Array的复杂版本，而数组大小是不变的

可增加和减少元素

1. ArrayList常用方法：

* add(E e) ：将指定的元素添加到此列表的尾部。
* add(int index, E element) ：将指定的元素插入此列表中的指定位置。
* clear() ：移除此列表中的所有元素。
* contains(Object o) ：如果此列表中包含指定的元素，则返回 true。
* get(int index) ：返回此列表中指定位置上的元素。
* remove(int index) ：移除此列表中指定位置上的元素。
* remove(Object o) ：移除此列表中首次出现的指定元素（如果存在）。

1. HashMap是基于哈希表的 Map 接口的实现。
2. HashMap中存放的元素以“key-value”的形式出现，其中”key”是元素的名字，”value”是元素的值。
3. HashMap中不能存放key同名的元素，若key同名，则后添加的元素的value会覆盖前面元素的value
4. HashMap常用方法：

* clear() ：从此映射中移除所有映射关系。
* containsKey(Object key) ：如果此映射包含对于指定键的映射关系，则返回 true。
* entrySet() ：返回此映射所包含的映射关系的 Set 视图。
* get(Object key) ：返回指定键所映射的值；如果对于该键来说，此映射不包含任何映射关系，则返回 null。
* isEmpty() ：如果此映射不包含键-值映射关系，则返回 true。
* put(K key, V value) ：在此映射中关联指定值与指定键。
* remove(Object key) ：从此映射中移除指定键的映射关系（如果存在）。
* size() ：返回此映射中的键-值映射关系数。

(11) iterator迭代器：一种设计模式，是一个对象

作用：在于对数据的遍历与数据的内部表示进行分离。

(12) iterator常用方法

* iterator()：要求容器返回一个iterator。
* next()：第一次调用iterator的next()方法时，它返回序列的第一个元素，以后再次调用时将获得序列中的下一个元素。
* hasNext()：使用检查序列中是否还有元素。
* remove()：将迭代器返回的元素删除。