1. Java实现排序

（一） 两种方式

Java要实现排序有两种方式，一种是在要排序的类中实现Comparable接口，该接口有一个compareTo方法需要实现，在排序时调用该类的compareTo方法即可。另一种是创建一个比较器类，该类实现了Comparator接口，在排序时使用Arrays.sort，传入该比较器，即可实现排序。

（二）Comparable接口

每个实现了Comparable接口的类都要实现compareTo方法，该方法用于定义该类的排序应该通过什么参数来比较。

如，类Employee实现了Comparable接口，实现compareTo方法，通过比较参数salary来排序，若第一个参数小于第二个参数，则返回一个负值，如果二者相等，则返回0，否则返回一个正值。这种比较方式数组就是按照升序排列的，可以这样理解，第一个参数就是靠前的参数，返回负值说明前面的参数小于后面的参数。

public int compareTo(Object otherObject){

Employee other = (Employee) otherObject;

return Double.compare(salary, other.salary);

}

在排序时，直接调用对应的sort方法即可，传入需要排序的数组。

（二）Comparator接口

第二种方法是创建一个比较器类，该类要实现Comparator接口，该接口同样有一个compare方法要实现，该方法就是定义如何进行比较的。同理，也是将两个对象要比较的参数进行相减，得到负数说明要升序排列。

class LengthComparator implements Comparator<String>{

public int compare(String first, String second){

return (first.length() – second.length());

}

}

调用时，实例化一个比较器类，然后在使用sort函数传入该比较器即可。

var comp = new LengthComparetor();

String[] friends = {“peter”, “paul”, “jane“};

Arrays.sort(friengs, new comp);

1. Java实现克隆
2. 概述

通常情况下，如果我们要将一个基本类型或者数值的对象赋给另一个对象时，我们直接用=完成这种操作，然而，这种方法实现的克隆只是对同一个对象的简单引用而已。如果是自定义的类，这种方法是行不通的。Object类的clone方法能够解决这种问题

1. clone()方法

Object类中有一个clone()方法，这是一个protected方法，说明只有同类可以调用clone方法进行克隆。若类中只有数值或者其他基本类型，那么这种克隆是可以将两个对象完全分离开的，两个对象有不同的状态，没有关联。但是若类中包含着对子对象的引用，克隆的字段就会得到相同子对象的一份引用，这样，原对象和克隆对象之间依然会共享一些信息。这种克隆称为“浅克隆”，如果源对象和克隆对象共享的子对象是不可变的，那么这种共享就是安全的，否则，就会出现问题。

1. Cloneable接口

要实现真正的深拷贝，拷贝所有的子对象，就要实现Cloneable接口，重新定义clone方法，并且指定为public修饰符。

1. 由于Object类中的clone()方法定义为protected，因此只有要克隆的类的子类才能够调用clone方法来克隆。要想实现类中子对象的克隆，就要开放权限，修改为public，让所有的方法能够克隆对象。
2. 这个Cloneable接口与真正的clone方法的实现没有关系,clone方法是从Object中集成的，这个接口只是一个标记，称为标记接口。
3. 重新定义的clone方法中需要调用super.clone()，用于克隆非子对象的子段。
4. 在前面克隆了非子对象的克隆类的基础上，添加克隆类中子对象的语句。

class Employee implements Cloneable{  
 …

public Employee clone() throws CloneNotSupportedException{

Employee cloned = (Employee) super.clone();

cloned.hireDay = (Date) hireDay.clone();

return cloned;

}

}