# 1 JDK定义的Serializable接口

在java.io.Serializable接口定义为：

public interface Serializable {  
}

注意java.io.Serializable接口是一个空接口，里面没有任何方法。java.io.Serializable接口的一个例子如下所示：

public class Person implements Serializable {  
 private String name;  
 private int age;  
  
 //注意这里没有缺省构造  
 public Person(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
  
 public String toString() {  
 return "name = " + name + ", age = " + age;  
 }  
  
 public static void main(String[] args) {  
 //序列化对象到文件  
 try (//创建一个ObjectOutputStream输出流  
 //写到二进制文件，因此d:/object.txt打开都是乱码  
 ObjectOutputStream oos = new ObjectOutputStream(

new FileOutputStream("d:/object.txt"))) {  
 //将对象序列化到文件  
 Person person = new Person("aaa", 23);  
 oos.writeObject(person);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
  
 //然后从d:/object.txt文件里反序列化，直接产生新的对象  
 try{  
 ObjectInputStream ois = new ObjectInputStream(

new FileInputStream("d:/object.txt"));  
 //注意这个时候不是调用构造函数，而是直接从文件反序列化到内存里产生对象  
 Person newPerson = (Person) ois.readObject();  
 System.*out*.println(newPerson);  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
}

**注意上面代码调用ObjectOutputStream的writeObject方法将对象序列化到文件，参数是被序列化的对象；调用ObjectInputStream的readObject方法从文件反序列化出对象，不带参数，返回Person类型对象。**

序列化机制允许将实现序列化的Java对象转换位字节序列，这些字节序列可以保存在磁盘上，或通过网络传输，以达到目的主机的目标进程，以后通过反序列化恢复成原来的对象。序列化机制使得对象可以脱离程序的运行而独立存在。**序列化和反序列化不需要我们去关心文件的存储格式，特别是反序列化时不需要管对象的每个属性内容保存的格式。**

# 2 实验一所定义的FileSerializable接口

FileSerializable接口定义为：

*/\*\*  
 \* 定义文件序列化接口  
 \*/*public interface FileSerializable extends java.io.Serializable{  
 */\*\*  
 \* 写到二进制文件  
 \** ***@param*** *out :输出流对象  
 \*/* public abstract void writeObject(ObjectOutputStream out);  
  
 */\*\*  
 \* 从二进制文件读  
 \** ***@param*** *in ：输入流对象  
 \*/* public abstract void readObject(ObjectInputStream in);  
}

由于定义了二个接口方法，方法参数分别是ObjectOutputStream和ObjectInputStream类型对象，**因此这二个接口方法必须通过实现该接口的类的实例去调用，因此如果定义Person类来实现FileSerializable接口，那么实现方式和前面的例子是有区别的，特别是Person类需要定义缺省构造函数。下面是例子：**

public class Person1 implements FileSerializable {  
 private String name;  
 private int age;  
  
 //需要定义一个缺省构造函数，因为现在Person1实现接口FileSerializable  
 //必须通过对象调用FileSerializable方法  
 public Person1(){}  
  
 public Person1(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
  
 @Override  
 public void writeObject(ObjectOutputStream out) {  
 try {  
 //将this对象的成员依次序列化  
 out.writeObject(this.name);  
 out.writeObject(this.age);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void readObject(ObjectInputStream in) {  
 //将this对象的成员依次反序列化，注意和序列化次序要一致  
 try {  
 this.name = (String)(in.readObject());  
 this.age = (int)(in.readObject());  
 } catch (IOException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public String toString() {  
 return "name = " + name + ", age = " + age;  
 }  
  
 public static void main(String[] args){  
 try {  
 ObjectOutputStream out = new ObjectOutputStream(

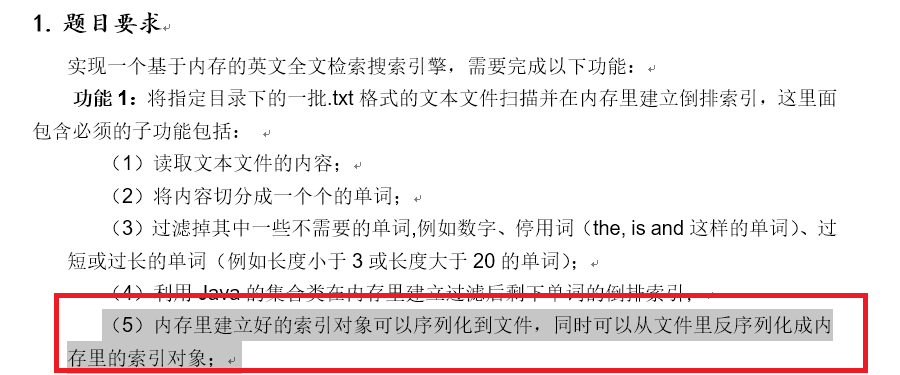
new FileOutputStream("d:\\object.txt")) ;  
 //将对象序列化到文件  
 Person1 person = new Person1("aaa", 23);  
 person.writeObject(out); //注意现在是通过对象调用writeObject，参数是out  
 } catch (IOException e) {  
 e.printStackTrace();  
 }

//反序列化  
 try {  
 ObjectInputStream in = new ObjectInputStream(

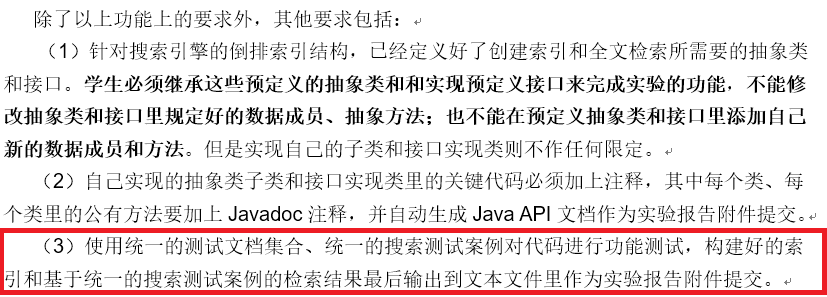
new FileInputStream("d:\\object.txt"));  
 //先构造一个空的Person1对象  
 Person1 newPerson= new Person1();  
 //再从文件反序列化出来对象所有成员  
 newPerson.readObject(in);  
 System.*out*.println(newPerson);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

**请大家注意二个例子的区别。**

# 3 实验一的序列化要求

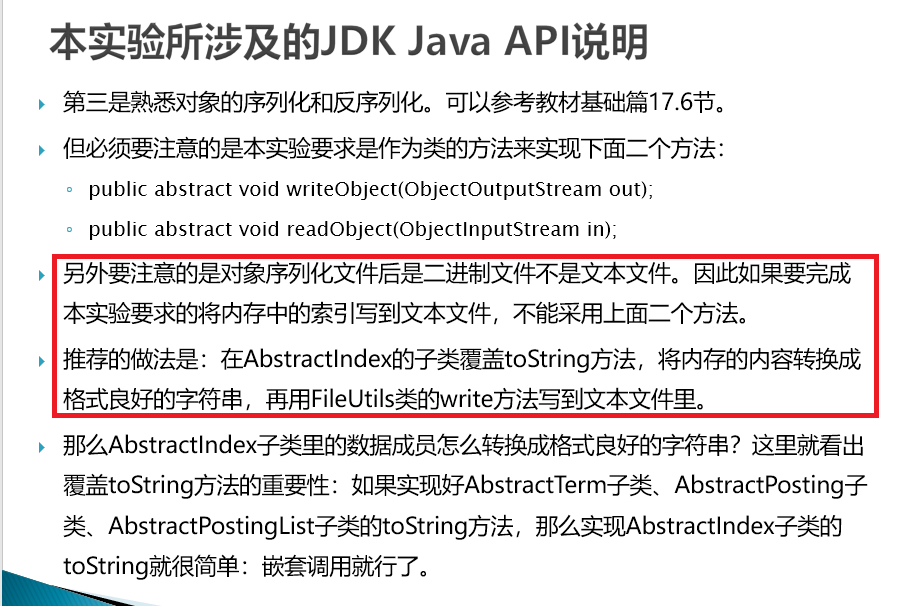


首先在实验一的实验报告里功能1的（5）要求为：内存里建立好的索引对象可以序列化到文件，同时可以从文件里反序列化成内存里的索引对象；**这是指序列化功能**。

****

实验一其他功能要求里的（3）为：使用统一的测试文档集合、统一的搜索测试案例对代码进行功能测试，构建好的索引和基于统一的搜索测试案例的检索结果最后输出到文本文件里作为实验报告附件提交。这是为了检查索引的构建结果，因为序列化文件是二进制文件，老师无法看到索引文件内容。所以需要把索引写到文本文件里。**这不是序列化。你甚至可以把控制台输出的索引对象的toString方法返回的字符串内容拷贝到一个文本文件提交上了都行。**

在实验一PPT里最后一页的解释同上面。

****

**这是为了检查索引的构建结果，因为序列化文件是二进制文件，老师无法看到索引文件内容。所以需要把索引写到文本文件里。这不是序列化。你甚至可以把控制台输出的索引对象的toString方法返回的字符串内容拷贝到一个文本文件提交上了都行。**