第四讲 JavaIO和反射

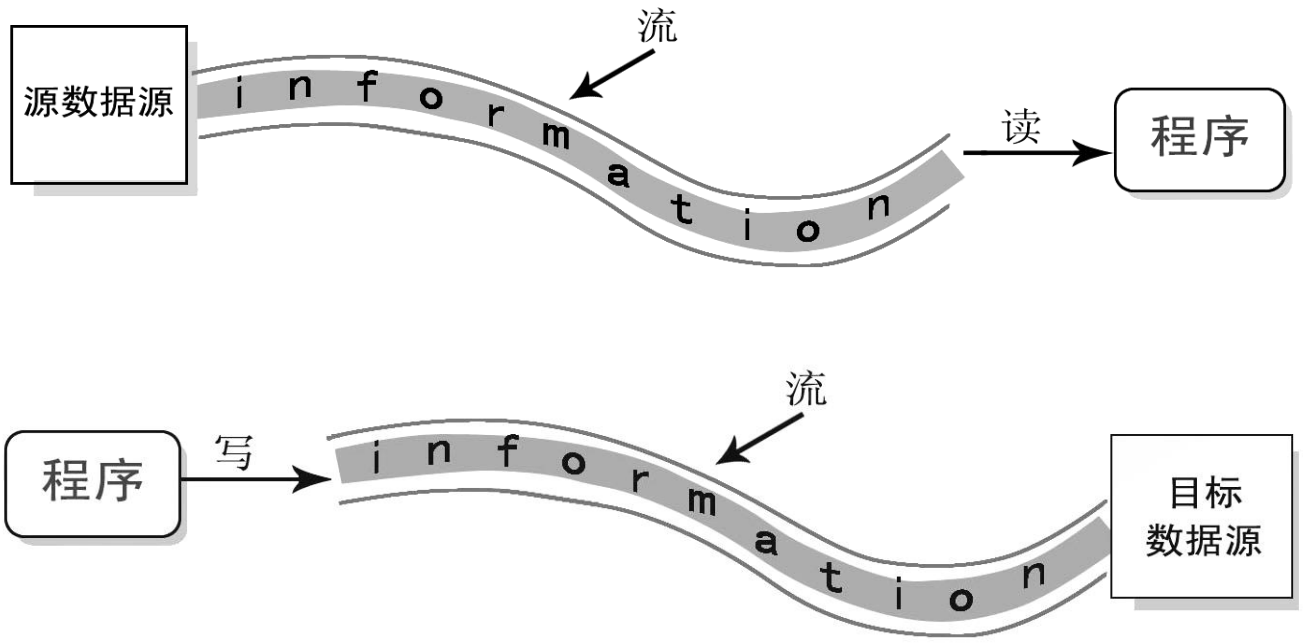
初版时间: 2017年5月1日
初版制作者: 林焕彬
教材版本号：szlanyou-V1.1

修订时间:
修订者:
教材版本号：szlanyou-V1.1

目录

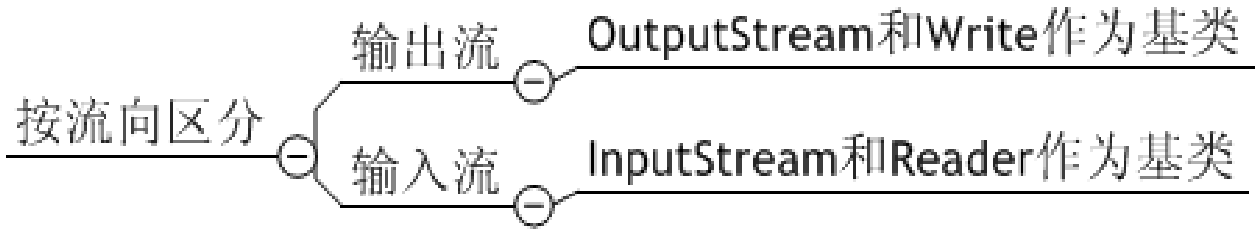
一、理解流的概念 1-1

- 流是指一连串流动的字符，是以先进先出方式发送信息的通道

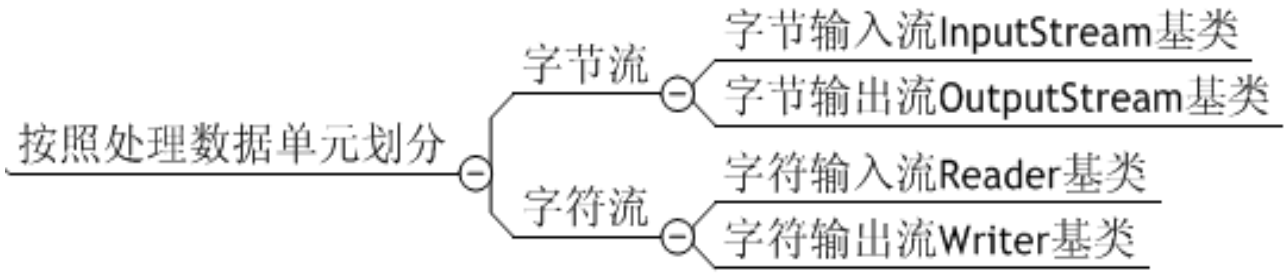


一、流的分类 1-2

- Java流的分类



输入输出流是相对于计算机内存来说的



字节流是 8 位通用字节流，字符流是 16 位 Unicode 字符流

一、流的分类 1-3

- 文本文件的读写
  - 用FileInputStream和FileOutputStream读写文本文件
  - 用BufferedReader和BufferedWriter读写文本文件
- 二进制文件的读写
  - 使用DataInputStream和DataOutputStream读写二进制文件

目录

二、读写文件 2-1

提问

使用FileInputStream 读文本文件的步骤是什么？

引入相关的类

```
import java.io.IOException;
import java.io.FileInputStream;
```

构造文件输入流 FileInputStream 对象

```
FileInputStream fis= new
    FileInputStream("text.txt");
```

读取文本文件的数据

```
fis.available();
fis.read();
```

关闭文件流对象

```
fis.close();
```

演示示例：使用FileInputStream 读文本文件

二、读写文件 2-2

提问

- 使用FileOutputStream 写文本文件的步骤是什么？与读文件的步骤有何不同？

引入相关的类

```
import java.io.IOException;
import java.io.FileOutputStream;
```

↓

构造文件输出流 FileOutputStream 对象

```
FileOutputStream fos = new
    FileOutputStream(“text.txt");
```

↓

把数据写入文本文件

```
String str ="好好学习Java";
byte[] words  = str.getBytes();
fos.write(words, 0, words.length);
```

↓

关闭文件流对象

```
fos.close();
```

二、读写文件 2-3

提问

- 用 BufferedReader 读文本文件的步骤是什么？

引入相关的类

↓

构造BufferedReader 对象和FileReader 对象

↓

调用readLine ()方法读取数据

↓

关闭文件流对象

FileReader 为读取字符文件的便捷类

注意

- 通过字符流的方式读取文件，并使用缓冲区，提高读文本文件的效率

演示示例：使用 BufferedReader 读文本文件

二、读写文件 2-4

提问

- 用 BufferedWriter 写文本文件的步骤是什么？

引入相关的类

↓

构造BufferedWriter对象和FileWriter对象

↓

调用write()方法写数据据

↓

流对象的清空和关闭flush()和close()

FileWriter为
写入字符文件的便捷类

注意

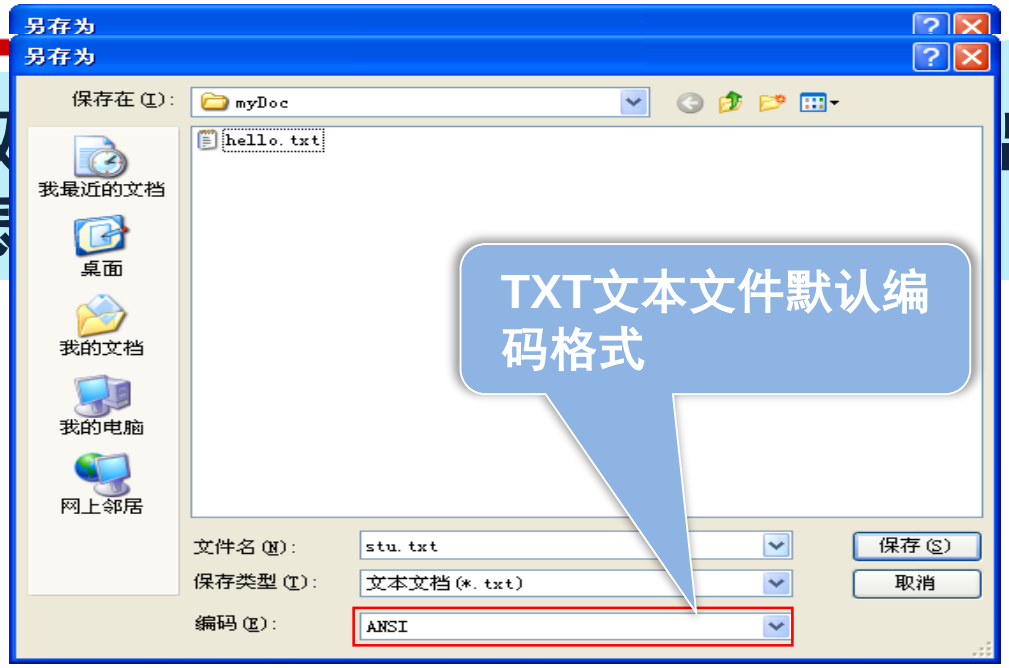
- 通过缓冲区把数据写入文本文件，提高写文本文件的效率

演示示例：使用 BufferedWriter 写文本文件

二、读写文件 2-5

问题

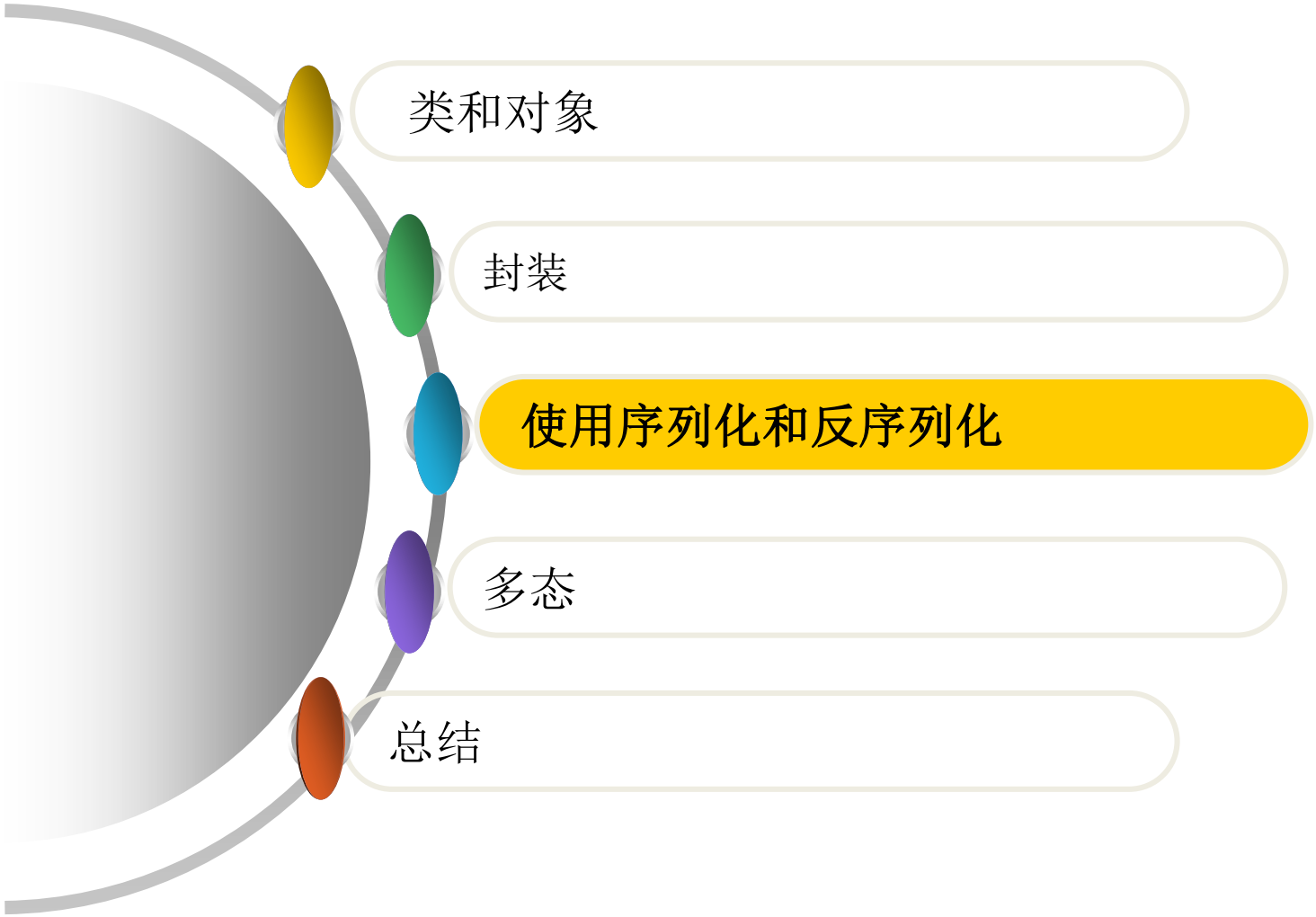
- 读取，怎出现中文乱码




```
FileInputStream fis=new FileInputStream("c:\\myDoc\\hello.txt");
//使用InputStreamReader并设置编码格式
InputStreamReader fr=new InputStreamReader(fis,"UTF-8");
BufferedReader br=new BufferedReader(fr);
```

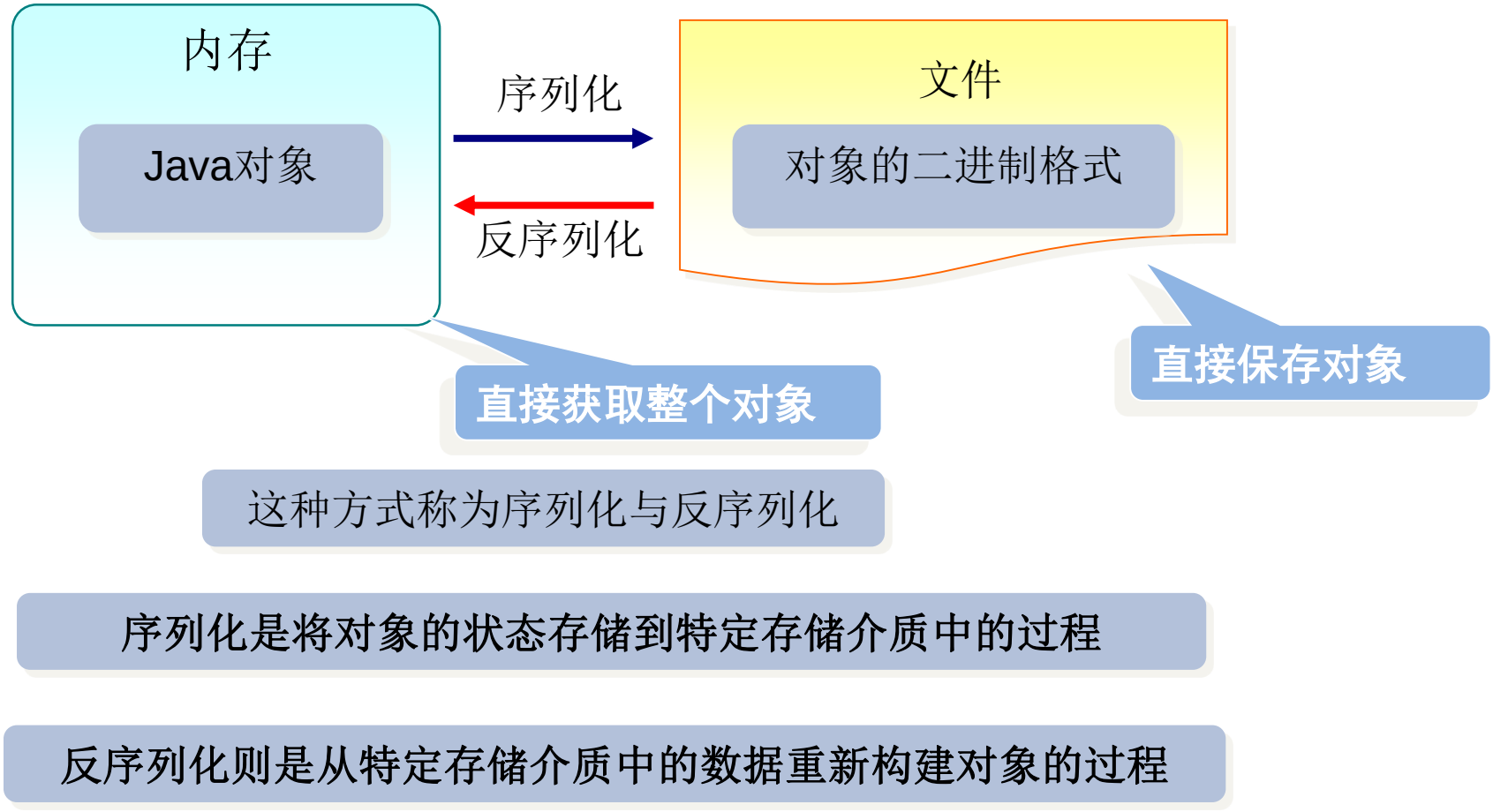
演示示例：解决中文乱码

目录



三、序列化和反序列化 3-1

- 序列化和反序列化的过程



序列化是将对象的状态存储到特定存储介质中的过程

反序列化则是从特定存储介质中的数据重新构建对象的过程

三、序列化和反序列化 3-2

提问

- 序列化的步骤是什么？

实现Serializable接口

↓

创建对象输出流

↓

调用writeObject()方法
将对象写入文件

↓

关闭对象输出流

注意

- 可以使用集合保存对象，然后将集合中所有的对象进行序列化

三、序列化和反序列化 2-3

提问

- 反序列化的步骤是什么？

实现Serializable接口

↓

创建对象输入流

↓

调用readObject ()方法读取对象

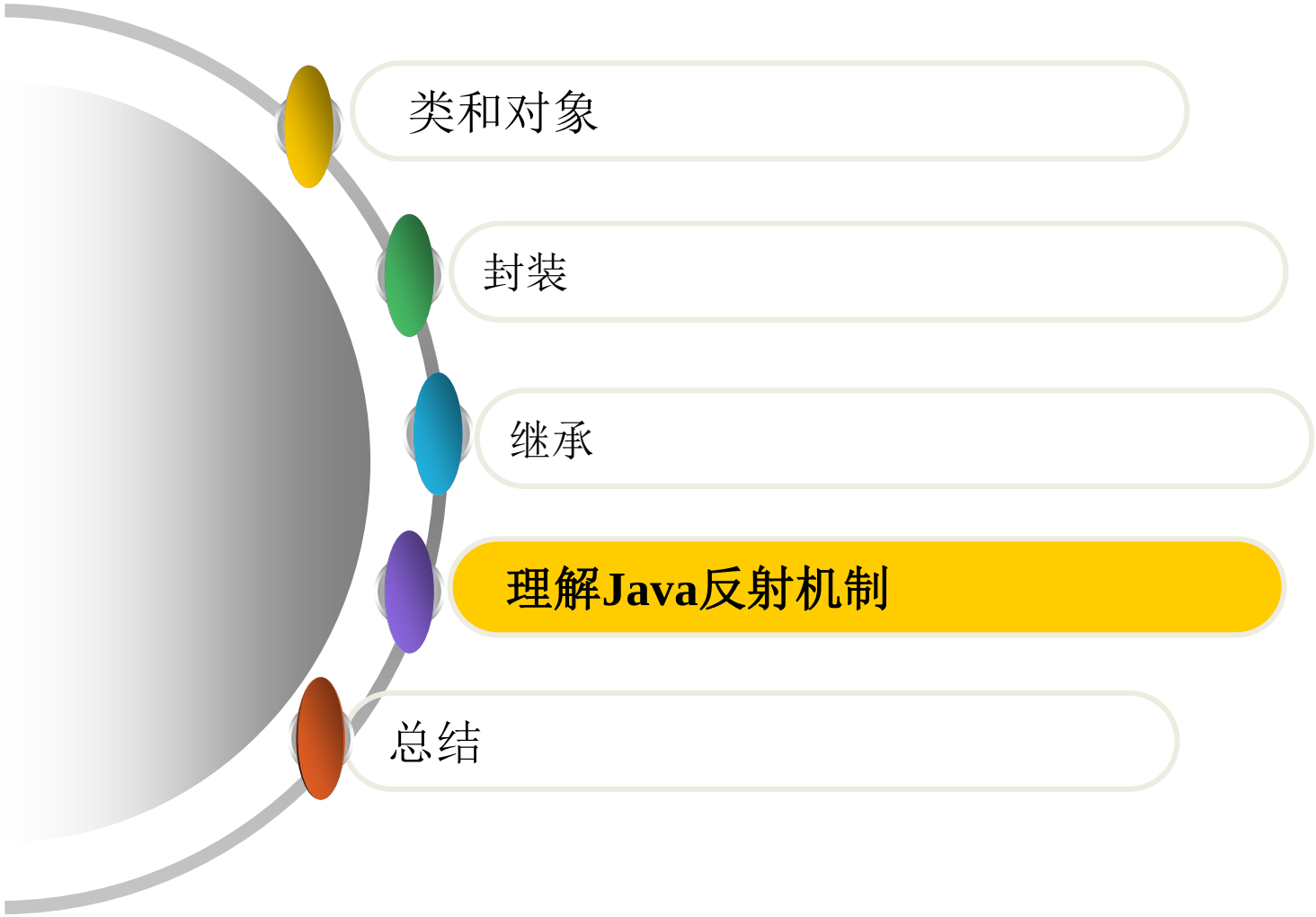
↓

关闭对象输入流

注意

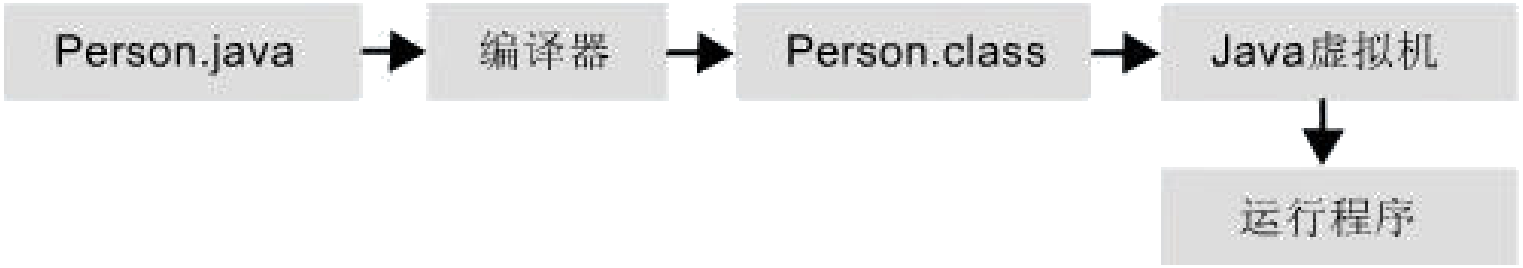
- 如果向文件中使用序列化机制写入多个对象，那么反序列化恢复对象时，必须按照写入的顺序读取

目录



四、什么是反射 4-1

- Java程序的运行过程



- 使用反射后Java 程序的运行过程

- Java反射：在编译时不确定哪个类被加载，而在程序运行时才加载、探知、使用

四、反射技术常用API 4-2

- 反射常用的Java 类型
  - Class类—可获取类和类的成员信息
  - Field类—可访问类的属性
  - Method类—可调用类的方法
  - Constructor类—可调用类的构造方法

上述类都位于java.lang.reflect包中

四、使用反射技术步骤 4-3

- 使用反射的基本步骤
  - 1.导入java.lang.reflect.*
  - 2.获得需要操作的类的Java.lang.Class对象
  - 3.调用Class的方法获取Field、Method等对象
  - 4.使用反射API进行操作（设置属性、调用方法）

四、反射的入口类Class 4-4

- 使用Class类获取类的结构信息

```
Class clazz = Class.forName("java.lang.Object");
```

获取类的Class 对象

```
Field fields[ ] = clazz.getDeclaredFields();
```

获取Field 对象

```
Method methods[] = clazz.getDeclaredMethods();
```

获取Method 对象

```
Constructor constructors[ ] = clazz.getDeclaredConstructors();
```

获取Constructor 对象

四、反射的入口类Class 4-5

- 获取 Class对象的方式

```
//方法1：对象.getClass()
Student stu=new Student();
Class clazz=stu.getClass();
```

```
//方法2：类.class
clazz= Student.class;
clazz=String.class;
```

```
//方法3：Class.forName()
clazz=Class.forName("java.lang.String");
clazz=Class.forName("java.util.Date");
```

四、反射的入口类Class 4-6

- 方法一：使用Class的newInstance()方法，仅适用于无参构造方法

```
Class clazz=Class.forName("cn.jbit.reflection.Student");
Object obj=clazz.newInstance();
```

相当于执行语句：
Student stu = new Student();

- 方法二：调用Constructor的newInstance()方法，适用所有构造方法

```
Constructor cons = clazz.getConstructor(new Class[ ]
                { String.class,int.class, float.class });
Object obj = cons.newInstance(new Object[ ] { "lkl", 32, 56.5f });
```

相当于执行语句：
Student stu=new Student("lkl",32,56.5f);

四、使用反射动态操作属性值 4-7

- 操作属性的基本步骤
  - 1、通过Class对象获取Field 对象
  - 2、调用Field 对象的方法进行取值或赋值操作

方法	说　　明
Xxx getXxx(Object obj)	获取基本类型的属性值
Object get(Object obj))	得到引用类型属性值
void setXxx(Object obj,Xxx val)	将obj对象的该属性设置成val值
void set(Object obj,object val)	将obj对象的该属性设置成val值
void setAccessible (bool flag)	对获取到的属性设置访问权限

四、使用反射动态执行方法 4-8

- 调用方法基本步骤
  - 1.通过Class对象获取Method 对象
  - 2.调用Method对象的invoke()方法

Object invoke(Object obj,Object [] args)

- 当前方法的返回值
- 当前方法所属对象的名字
- 当前方法的参数列表

演示示例：使用反射动态执行方法

四、反射技术的优点 4-9

- 提高了Java程序的灵活性和扩展性，降低了耦合性，提高自适应能力

允许程序创建和控制任何类的对象，无需提前硬编码目标类

- 反射的应用领域
  - 如软件测试、 EJB、JavaBean等
  - 开源框架例如Struts、Hibernate、Spring

四、反射技术的缺点 4-10

- 性能问题

  反射机制主要应用在对灵活性和扩展性要求很高的系统框架上

- 代码维护问题

  反射会模糊程序内部逻辑，可读性较差

目录

五、总结

提问

简述流的分类。

读写文件常用的类有哪些？

简述序列化和反序列化的步骤。

通过反射创建对象有几种方式，如何实现？

五、上机练习

练习

- *需求说明*
  - 使用BufferedReader和BufferedWriter实现文件的拷贝。

- *需求说明*
  - 使用反射修改和查询Student类的姓名属性。
  - 使用反射动态执行Student类输出信息的方法

完成时间：10分钟

共性问题集中讲解

Thank you

Q&A