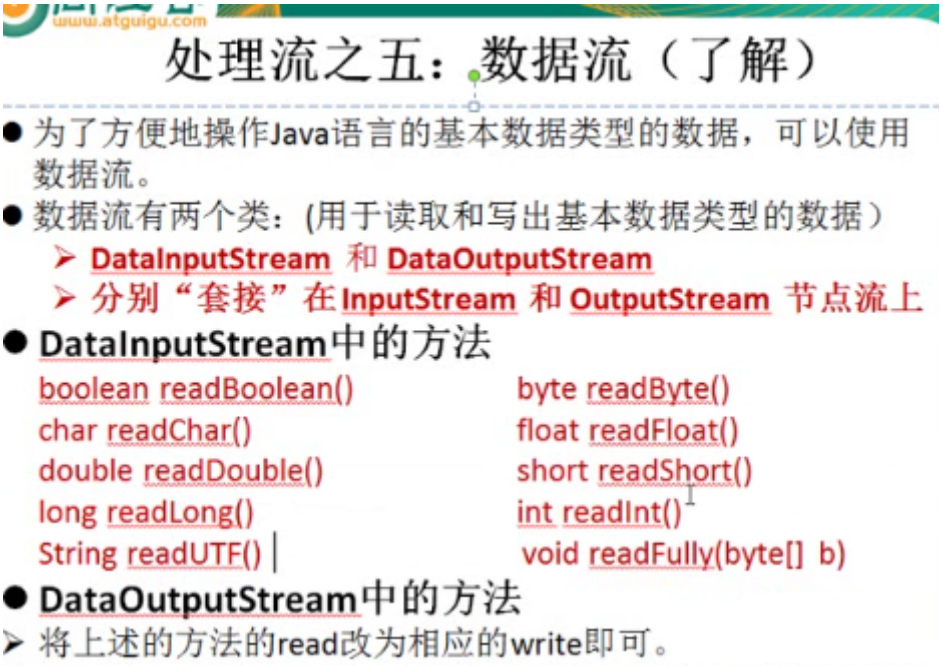
一，打印流：

```
//打印刘：字节流printstream，字符流printwriter
		FileOutputStream  fos = null;
		try {
			fos = new FileOutputStream(new File("print.txt"));

		} catch (Exception e) {
			// TODO: handle exception
			e.printStackTrace();
		}
		//创建打印输出流，设置为自动刷新模式，写入换行符或字节‘\n’时都会刷新输出缓冲区
		PrintStream pStream = new PrintStream(fos,true);
		if (pStream != null) {
			//把标准输出流（控制台输出）改成文件输出
			//输出到一个新的打印台的位置
			System.setOut(pStream);
		}
		for (int i = 0; i <= 255; i++) {
			//输出ASCII字符
			System.out.print((char)i);
			if (i % 30 ==0) {
				//每30个数据换一行
				System.out.println();
			}
		}
		pStream.close();
```

二，数据流：



三，对象流

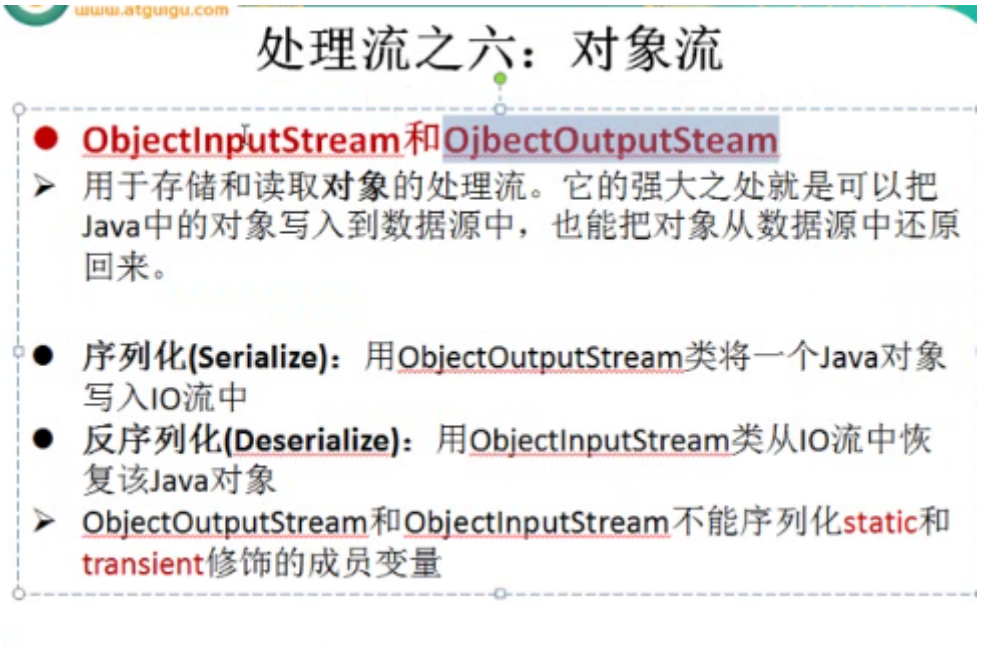
1，对象的序列化

①对象的序列化机制：允许把内存中的java对象转换成为与平台无关的二进制流，从而允许把这种二进制流持久的保存在磁盘上，或者通过网络将这种二进制流传输到另一个网络节点，当其他程序获取了这种二进制流，就可以恢复成为原来的java对象

②序列化的好处在于可将任何实现了serializable接口的对象转化为字节数据，使其在保存和传输的时候可以被还原

③序列化是rmi（remote method invoke---远程方法调用）过程的参数和返回值都必须实现的机制，而rmi是javaee的基础，因此序列化机制是javaee平台的基础

④如果需要让某一个对象支持序列化机制，则必须让其类是可序列化的，为了让某个类是可序列化的，该类必须实现如下俩个接口之一：serializable和externalizable



要求实现序列化的类：

①要求此类是可序列化的，实现serializable接口

②要求类的属性同样的要实现serializable接口

③提供一个版本号，privatestatic final long serialversionUID

④使用static或transient修饰的属性，不可实现序列化

四，随机访问流

alt+向上键或者向下键表示向上或者向下移动一行

```
* RandomAccessFile:支持随机访问
        * 1，既可以充当一个输入流，又可以充当一个输出流
        * 2，支持从文件的开头读取和写入
        * 3，支持从任意位置的读取和写入
//进行文件的读和写
        RandomAccessFile randomAccessFile=null;
        RandomAccessFile  rand =null;
        try {
                randomAccessFile = new RandomAccessFile(new
File("C://Users//Anly//Desktop//55.txt"), "r");
                rand = new RandomAccessFile (new
File("C://Users//Anly//Desktop//55.txt"),"rw");
                byte []b = new byte[100];
                int len;
                try {
                        while((len = randomAccessFile.read(b)) != -1){
                                rand.write(b,0,len);
                        }
                } catch (IOException e) {
                        // TODO Auto-generated catch block
```

```
                e.printStackTrace();
        }
} catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
finally {
        if (rand!=null) {
                try {
                        rand.close();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }
        if (randomAccessFile != null) {
                try {
                        randomAccessFile.close();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }
}
```