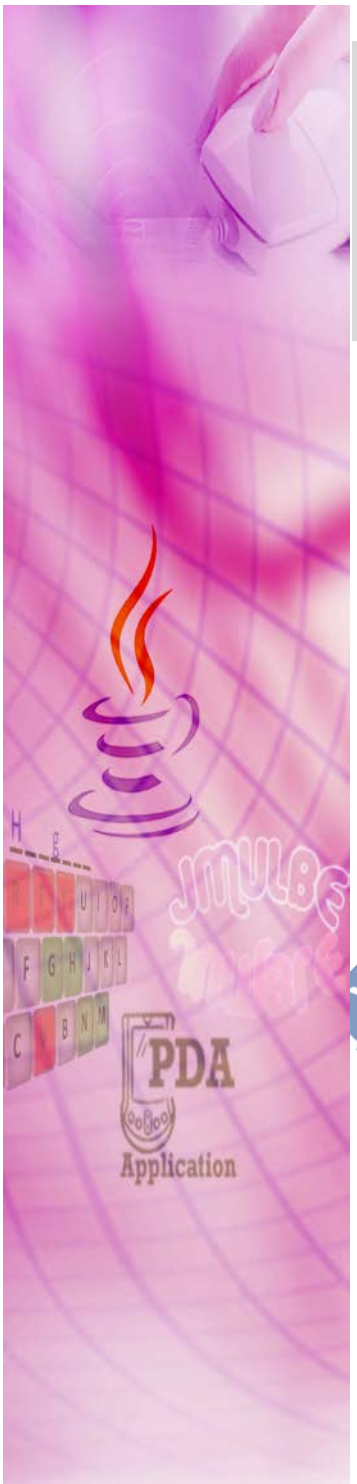


第 7 章

使用流

大多数程序都接受用户输入、处理该输入并生成输出。因此，所有编程语言都支持从输入流（例如文件）读取输入，然后通过输出流（例如控制台）显示输出。Java 以流的形式处理所有输入和输出操作，这些流充当从源流到目的地的字节或字符序列。当发送数据流时，称为写；当接收数据流时，称为读取。

本章重点讲述读取输入流的数据以及将数据写入输出流。

目标

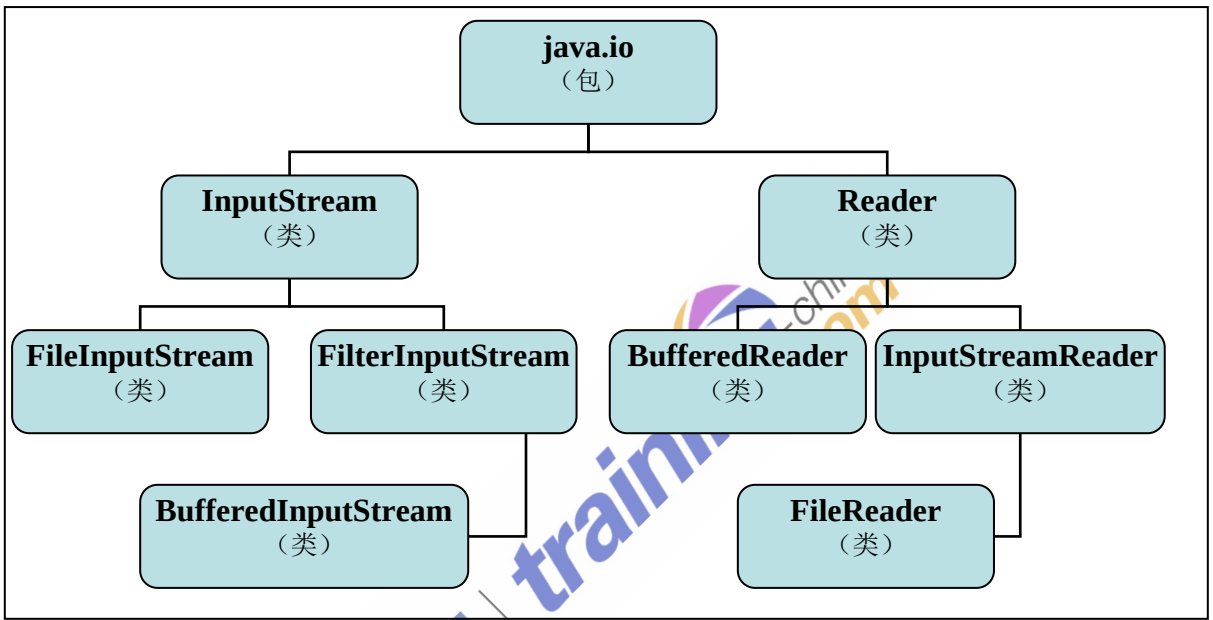
在本章中，您将学习：

- 使用输入流
- 使用输出流


NIIT
training
china
.com

使用输入流

思考以下场景：您需要开发将文档从一种格式转换到另一格式（例如 .txt 到 .pdf）的 Java 应用程序。要开发此类应用程序，需要从文件之类的输入流中读取数据。可以字节或字符的形式从这些流中读取数据。为此，Java 提供 `InputStream` 类及其子类。要以字符的形式读取数据，Java 在 `java.io` 包中提供 `Reader` 类。下图显示 `java.io` 包的类层次结构。



输入流类层次结构

`FileInputStream` 和 `FilterInputStream` 类是 `InputStream` 类的子类。`BufferedInputStream` 类是 `FilterInputStream` 类的子类。`BufferedReader` 和 `InputStreamReader` 类是 `Reader` 类的子类。`FileReader` 类是 `InputStreamReader` 类的子类。

使用 `FileInputStream` 类

*`FileInputStream` 类*用于从文件读取数据和字节流。`FileInputStream` 类提供了可用于创建实例的各种构造函数。

下表列出了 FileInputStream 类的各种构造函数。

构造函数	*描述*
FileInputStream(File file)	*它通过在文件系统中打开实际文件 file 的连接，创建 FileInputStream 的实例。*
FileInputStream(FileDescriptor fdObj)	*它通过使用文件描述符 fdObj 创建 FileInputStream 的实例，该文件描述符表示文件系统中某个实际文件的现有连接。*
FileInputStream(String name)	*它通过打开 name 参数中指定的某个实际文件的连接，创建 FileInputStream。*

FileInputStream 类的构造函数

下表列出了 FileInputStream 类的方法及其描述。

方法	*描述*
int read()	*用于从输入流读取一个数据字节。*
FileDescriptor getFD()	*用于返回 FileDescriptor 对象，该对象表示文件系统中实际文件的连接。*

FileInputStream 类的方法

思考通过 FileInputStream 类读取文件数据的以下代码：

```
import java.io.FileInputStream;
import java.io.IOException;
public class FileInputStreamDemo
{

  public static void main(String[] args)
   {
        int i;
        char c;
        try (FileInputStream f = new FileInputStream("D:\\Files\\File.txt"))
{
            while ((i = f.read()) != -1)
{
                c = (char) i;
                System.out.print(c);
            }
        }
catch (IOException ex)
       {
```

```
            System.out.println(ex);
        }
    }
}
```

在上述代码中，创建了 FileInputStreamDemo 类，它通过 read() 方法读取 File.txt 文件的数据。System.out.print(c); 语句打印文件中存在的数据的字节。

使用 BufferedInputStream 类

*BufferedInputStream 类*用于通过内存中的临时存储区即缓冲区执行读操作。缓冲区用于在作出读取数据的第一个系统调用时存储检索到的数据。此外，对于每个读取数据的后续请求，缓冲区用于供程序检索数据，而不是作出系统调用。使用缓冲区而不是系统调用检索数据要快得多，因此提高了程序的效率。BufferedInputStream 类提供了可用于创建实例的各种构造函数。下表列出了 BufferedInputStream 类的各种构造函数。

构造函数	*描述*
BufferedInputStream(InputStream in)	*它创建 BufferedInputStream 的实例，并将指定的 InputStream 参数保存在 in 参数中供以后使用。*
BufferedInputStream(InputStream in, int size)	*它创建具有指定缓冲区大小的 BufferedInputStream 的实例，并将 InputStream 参数保存在 in 参数中供以后使用。*

BufferedInputStream 类的构造函数

下表列出了 BufferedInputStream 类的方法及其描述。

方法	*描述*
int available()	*用于返回在不阻塞此输入流的下一个方法调用的情况下可从此输入流读取的估计字节数。*
void mark(int readlimit)	*用于标记流中的当前位置。对 reset() 的后续调用尝试将流重新定位到此点。*
void reset()	*reset() 方法使流回到标记点。*
boolean markSupported()	*用于测试此输入流是否支持 mark() 和 reset() 方法。*

方法	*描述*
void close()	*用于关闭输入流并释放与该流关联的任何系统资源。一旦流关闭，接着 read()、available()、reset() 或 skip() 调用将抛出 IOException。*
int read(byte[] b,int off,int len)	*用于将输入流的字节读入指定的字节数组，从给定偏移开始。*
long skip(long n)	*用于跳过此输入流中 n 个字节的数据。*

BufferedInputStream 类的方法

思考在控制台上打印输出的以下 BufferedInputStream 代码：

```
import java.io.BufferedInputStream;
import java.io.ByteArrayInputStream;
import java.io.IOException;

public class BufferedInputStreamDemo
{
public static void main(String[] args)
{
String s = "This is a BufferedInputStream Demo Program";
byte buf[] = s.getBytes();

try(ByteArrayInputStream in = new ByteArrayInputStream(buf);
BufferedInputStream f = new BufferedInputStream(in)){

int c;
while ((c = f.read()) != -1)
{
System.out.print((char) c);
}
}
catch(IOException e)
{
System.out.println(e);
}
}
}
```

在上述代码中，创建了 BufferedInputStreamDemo 类。System.out.print((char) c); 语句在控制台上打印输出。

注释

ByteArrayInputStream 类使您能够在内存中创建用作 InputStream 的缓冲区。输入源是字节数组。

使用 FileReader 类

*FileReader 类*用于从文件读取字符，但是它不定义自己的任何方法。它从自己的基类派生所有方法，例如 Reader 和 InputStreamReader 类。FileReader 类提供了可用于创建实例的各种构造函数。下表列出了 FileReader 类的各种构造函数。

构造函数	***描述***
FileReader(File file)	*它创建 FileReader 的实例，用于读取 file 参数指定的文件。*
FileReader(FileDescriptor fd)	*它创建 FileReader 的实例，用于读取指定的 FileDescriptor 类。*
FileReader(String fileName)	*它创建 FileReader 的实例，用于读取 fileName 参数所指定名称的文件。*

FileReader 类的构造函数

思考以下读取文件数据的代码：

```
import java.io.FileReader;
import java.io.IOException;

class FileReaderDemo {
public static void main(String args[])
{
try (FileReader f = new FileReader("D:\\Files\\file.txt "))
{
char[] a = new char[50];
f.read(a);
for (char c : a)
{
System.out.print(c);
}
}
catch (IOException e)
{
System.out.println(e);
}
}
}
```

在上述代码中，创建了 FileReaderDemo 类，用于读取文件 file.txt 的数据。System.out.print(c); 语句在控制台上打印输出。

使用 BufferedReader 类

*BufferedReader 类*用于在缓存字符时读取字符输入流（例如文件、控制台和数组）的文本。由于数据已缓存，因此使用 BufferedReader 类的读操作变得更加高效。BufferedReader 类提供了可用于创建实例的各种构造函数。下表列出了 BufferedReader 类的各种构造函数。

构造函数	*描述*
BufferedReader(Reader in)	*它创建使用默认大小输入缓冲区的 BufferedReader 实例。*
BufferedReader(Reader in, int sz)	*它创建使用指定大小输入缓冲区的 BufferedReader 实例。*

BufferedReader 类的构造函数

下表列出了 BufferedReader 类的常用方法及其描述。

方法	*描述*
void mark(int readAheadLimit)	*用于标记流中的当前位置。对 reset() 的后续调用尝试将流重新定位到此点。*
boolean markSupported()	*用于告诉此流是否支持 mark() 操作。*
String readLine()	*用于读取文本行。行在以下情况下被视为已终止：换行符（'\n'）、回车符（'\r'）或回车符直接跟有换行符。*
boolean ready()	*用于告诉此流是否已读取就绪。缓存的字符流在缓冲区非空或者底层字符流就绪的情况下处于就绪状态。*

BufferedReader 类的方法

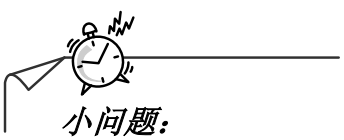
思考以下代码，它使用 BufferedReader 类接受用户输入的两个数字并打印其和：

```
import java.io.*;

public class BufferedReaderDemo
{
public static void main(String args[]) throws IOException
{
try(BufferedReader br = new BufferedReader(new
InputStreamReader(System.in))){
```

```
System.out.println("Enter First number");
String s = br.readLine();
System.out.println("Enter Second number");
String s1 = br.readLine();
int i = Integer.parseInt(s);
int i1 = Integer.parseInt(s1);
int i3 = i + i1;
System.out.println("Sum=" + i3);
}
}
}
```

在上述代码中，创建了 `BufferedReaderDemo` 类的对象，用于打包 `InputStreamReader` 类以从控制台读取数据。`readLine()` 方法接受用户输入并分别存储在字符串变量 `s` 和 `s1` 中。然后字符串值将通过 `parseInt()` 方法转换为整数值，此方法是 `Integer` 类的静态方法。然后 `System.out.println("Sum=" + i3);` 语句打印输出。



小问题：

以下哪个方法用于确保 `FileInpuStream` 类的 `close()` 方法将在不再有对它的引用时被调用？

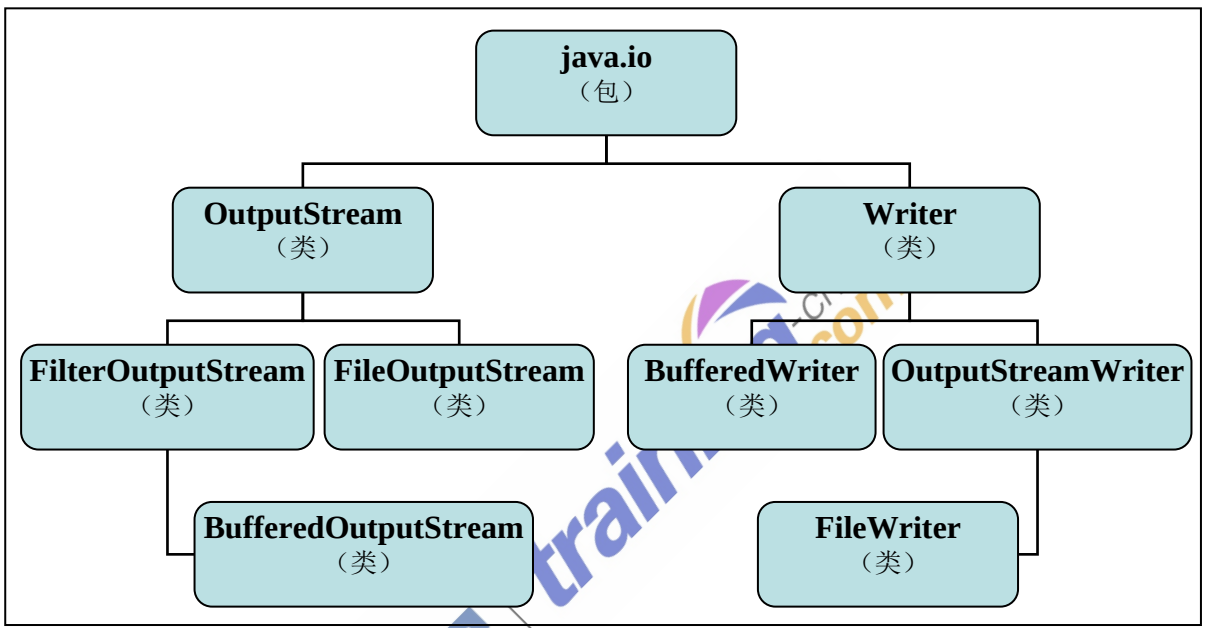
1. *`finalize()`*
2. *`ready()`*
3. *`reset()`*
4. *`close()`*

答案：

1. *`finalize()`*

使用输出流

思考需要开发银行应用程序的场景。在此应用程序中，需要实现一组功能以维护应用程序中执行的每个事务的日志。为此，需要将数据写入文件之类的输出流。可以字节或字符的形式向这些流写入数据。要以字节的形式写入数据，Java 提供 `OutputStream` 类。要以字符的形式写入数据，Java 在 `java.io` 包中提供 `Writer` 类。下图显示 `java.io` 包的类层次结构。



输出流类层次结构

`FileOutputStream` 和 `FilterOutputStream` 类是 `OutputStream` 类的子类。`BufferedOutputStream` 类是 `FilterOutputStream` 类的子类。`BufferedWriter` 和 `OutputStreamWriter` 类是 `Writer` 类的子类。`FileWriter` 类是 `OutputStreamWriter` 类的子类。

使用 `FileOutputStream` 类

*`FileOutputStream` 类*用于将数据逐个字节地写入文件。`FileOutputStream` 类的实例创建不依赖于需要被执行写操作的文件。如果文件不存在，将先创建文件，然后打开它进行输出操作。如果文件处于只读模式，将抛出错误。`FileOutputStream` 类提供了可用于创建实例的各种构造函数。

下表列出了 FileOutputStream 类的各种构造函数。

构造函数	*描述*
FileOutputStream(File file, boolean append)	*它创建 FileOutputStream 的实例，用于写入所指定 File 对象表示的文件。如果 append 参数是 true，那么字节将写到文件结束处。否则，字节将写到开头。*
FileOutputStream(FileDescriptor fdObj)	*它创建 FileOutputStream 的实例，用于写入表示实际文件现有连接的文件描述符。*
FileOutputStream(String name)	*它创建 FileOutputStream 的实例，用于写入 name 参数所指定名称的文件。*
FileOutputStream(String name, boolean append)	*它创建 FileOutputStream 的实例，用于写入 name 参数所指定名称的文件。如果 append 参数是 true，那么字节将写到文件结束处。否则，字节将写到开头。*

FileOutputStream 类的构造函数

下表列出了 FileOutputStream 类的常用方法及其描述。

方法	*描述*
FileDescriptor getFD()	*用于返回与此流关联的文件描述符。*
void write(int b)	*用于将指定字节写到此文件输出流。*

FileOutputStream 类的方法

思考通过 FileOutputStream 类将数据写入文件的以下代码：

```
import java.io.*;
public class FileOutputStreamDemo
{
    public static void main(String[] args) throws IOException
    {
           boolean bool;
           long pos;
           String s = "This is a FileOutputStream Program";
           byte buf[] = s.getBytes();
           try (FileOutputStream fos = new
       FileOutputStream("D:\\Files\\File.txt "))
           {
                  for (int i = 0; i < buf.length; i++)
```

```
                {
                        fos.write(buf[i]);
                }
        }
        catch (Exception e)
        {
                System.out.println(e);
    }
    }
}
```

在上述代码中，创建了 FileOutputStreamDemo 类。fos.write(buf[i]); 语句将数据写入文件。

使用 BufferedOutputStream 类

*BufferedOutputStream 类*使用缓冲区将字节写入输出流以提高效率。BufferedOutputStream 类提供了可用于创建实例的各种构造函数。下表列出了 BufferedOutputStream 类的各种构造函数。

构造函数	***描述***
BufferedOutputStream(OutputStream out)	*它创建缓存输出流的实例，用于将数据写入指定的输出流。*
BufferedOutputStream(OutputStream out, int size)	*它创建缓存输出流的实例，用于将数据写入具有指定缓冲区大小的指定输出流。*

BufferedOutputStream 类的构造函数

下表列出了 BufferedOutputStream 类的常用方法及其描述。

方法	***描述***
void flush()	*用于清空缓存的输出流。这将强制缓存的输出字节写到输出流。*
void write(byte[] b, int off, int len)	*用于写入字节数组中的 len 个字节。它从偏移值 off 开始写到缓存输出流。*

BufferedOutputStream 类的方法

思考在控制台上打印输出的 BufferedOutputStream 类的以下代码：

```
import java.io.BufferedOutputStream;
import java.io.IOException;

public class BufferedOutputStream
{
public static void main(String[] args)
```

```
{
try (BufferedOutputStream b = new BufferedOutputStream(System.out))
{
String s = "This is a BufferedOutputStream Demo Program";
byte buf[] = s.getBytes();

b.write(buf);
b.flush();
}
catch (IOException e)
{
System.out.println(e);
}
}
}
```

在上述代码中，创建了 `BufferedOutputStream` 类。`b.write(buf);` 语句在控制台上打印输出。

使用 BufferedWriter 类

*`BufferedWriter` 类*可用于将文本写到输出流。此类将相对大的数据块一次写到输出流。`BufferedWriter` 类提供了可用于创建实例的各种构造函数。下表列出了 `BufferedWriter` 类的各种构造函数。

构造函数	***描述***
`BufferedWriter(Writer out)`	*它创建使用默认大小输出缓冲区的 `BufferedWriter` 实例。*
`BufferedWriter(Writer out, int size)`	*它创建使用给定大小输出缓冲区的 `BufferedWriter` 实例。*

`BufferedWriter` 类的构造函数

下表列出了 BufferedWriter 类的常用方法及其描述。

方法	*描述*
void newLine()	*用于写入由系统属性 line.separator 所定义的行分隔符。*
void write(char[] cbuf, int off, int len)	*用于写入一组字符的一部分。*

BufferedWriter 类的方法

思考通过 BufferedWriter 类将数据写入控制台的以下代码：

```
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.OutputStreamWriter;

public class BufferedWriterDemo
{
public static void main(String args[])
{
try (BufferedWriter b = new BufferedWriter(new
OutputStreamWriter(System.out)))
{
String fruit[] = {"Apple", "Banana", "Grapes"};
b.write("Different types of fruit are:"+ "\n");
for (int i = 0; i < 3; i++)
{
b.write(fruit[i] + "\n");
b.flush();
}
}
catch (IOException e)
{
System.out.println(e);
}
}
}
```

上述代码的输出是：

```
Different types of fruit are:
Apple
Banana
Grapes
```

在上述代码中，创建了一组水果。然后创建了 BufferedWriter 的对象，用于打包 OutputStreamWriter 类以将数据写到控制台输出流。write() 方法将字符串写到控制台。

使用 FileWriter 类

*FileWriter 类*将字符数据写到文件。此类不定义其自身的任何方法。它从自己的基类派生所有方法，例如 Writer 和 OutputStreamWriter 类。

FileWriter 类提供了可用于创建实例的各种构造函数。下表列出了 FileWriter 类的构造函数。

构造函数	*描述*
FileWriter(File file)	*它从 File 对象创建 FileWriter 对象的一个实例。*
FileWriter(File file,boolean append)	*它从 File 对象创建 FileWriter 对象的一个实例。 如果第二个参数是 true，那么字符将写到文件结尾。否则，字符将写到文件开头。*
FileWriter(FileDescriptor fd)	*它创建与文件描述符关联的 FileWriter 对象的实例。*
FileWriter(String fileName,boolean append)	*它创建具有给定文件名的 FileWriter 对象的实例。如果第二个参数是 true，字符将写到文件结尾。否则，字符将写到文件开头。*

FileWriter 类的构造函数

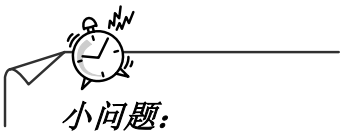
思考将数据写到文件的以下代码：

```
import java.io.FileWriter;
import java.io.IOException;

class FileWriterDemo
{
public static void main(String args[])
{
try (FileWriter f = new FileWriter("D:\\Files\\file.txt "))
{
String source = "This is FileWriter Program";
char buffer[] = new char[source.length()];
source.getChars(0, source.length(), buffer, 0);
f.write(buffer);
}

catch (IOException e)
{
System.out.println(e);
}
}
}
```

在上述代码中，创建了 `FileWriterDemo` 类以将数据写到 `file.txt` 文件。`f.write(buffer);` 语句将数据写入文件。



小问题：

如果需要将数据追加到新文件结尾，您将使用以下哪个 `FileWriter` 构造函数？

1. *`FileWriter(File file)`*
2. *`FileWriter(File file,boolean append)`*
3. *`FileWriter(FileDescriptor fd)`*
4. *`FileWriter(String fileName,boolean append)`*

答案：

4. *`FileWriter(String fileName,boolean append)`*

活动 7.1：使用输入流和输出流

练习问题

1. `FileOutputStream` 类的以下哪个构造函数用于将字节写到给定文件结尾？
   a. `FileOutputStream(File file,boolean append)`
   b. `FileOutputStream(FileDescriptor fdObj)`
   c. `FileOutputStream(File file)`
   d. `FileOutputStream(String name,boolean append)`

2. 说明以下语句是正确还是错误。
   `markSupported()` 方法用于测试输入流是否支持 `mark()` 和 `reset()` 方法。

3. 以下哪个方法用于返回可从输入流读取的估计字节数？
   a. `available()`
   b. `reset()`
   c. `read()`
   d. `ready()`

4. 以下哪个类不具有自己的方法并且继承 `InputStreamWriter` 类的方法？
   a. `FileWriter`
   b. `BufferedWriter`
   c. `BufferedOutputStream`
   d. `FileOutputStream`

5. 以下哪个类不具有自己的方法并且继承 `InputStreamReader` 类的方法？
   a. `FileReader`
   b. `BufferedReader`
   c. `BufferedInputStream`
   d. `FileInputStream`

小结

在本章中，您学习了：

- Java 以流的形式处理所有输入和输出操作，这些流充当从源流到目的地的字节或字符序列。
- 当发送数据流时，称为写；当接收数据流时，称为读取。
- 要以字符的形式读取数据，Java 在 `java.io` 包中提供 `Reader` 类。
- `FileInputStream` 类用于从文件读取数据和字节流。
- `BufferedInputStream` 类用于通过内存中的临时存储区即缓冲区执行读操作。
- `FileReader` 类用于从文件读取字符，但是它不定义自己的任何方法。
- `BufferedReader` 类用于在缓存字符时读取字符输入流（例如文件、控制台和数组）的文本。
- 要以字节的形式写入数据，Java 提供 `OutputStream` 类。
- 要以字符的形式写入数据，Java 在 `java.io` 包中提供 `Writer` 类。
- `FileOutputStream` 用于将数据逐个字节地写入文件。
- `BufferedOutputStream` 类使用缓冲区将字节写入输出流以提高效率。
- `BufferedWriter` 类可用于将文本写到输出流。
- `FileWriter` 类将字符数据写到文件。此类不定义其自身的任何方法。