Java

Файловый ввод / вывод

Работа с текстовыми файлами

Буферизация потоков

Лекция #7 B

Пустовалова О.Г.
доцент. каф. мат.мод.
ИММиКН ЮФУ

Содержание

Иерархия классов для управления потоками Ввода и Вывода

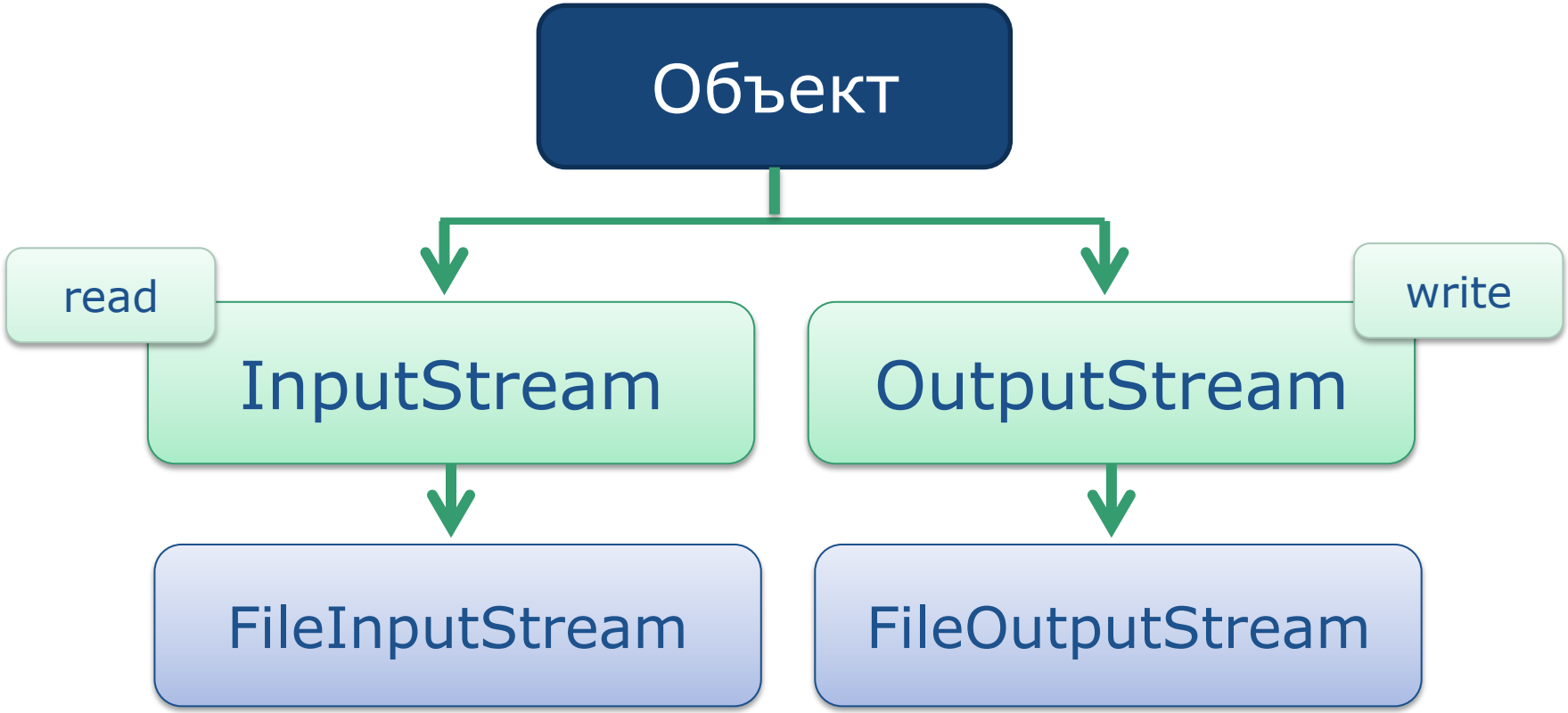
В абстрактных классах **InputStream** и **OutputStream** определяется ряд ключевых методов, реализуемых в других классах потоков ввода-вывода.

Наиболее важными среди них являются методы **read**() и **write**() , читающие и записывающие байты данных соответственно.

Оба эти метода объявлены как абстрактные в классах **Inputstream** и **OutputStream**, а в производных классах они переопределяются.

Необходимо импортировать пакет **java.io**, чтобы воспользоваться классами потоков ввода-вывода.

Иерархия классов для управления потоками Ввода и Вывода



Абстрактный класс похож на обычный **класс**. В **абстрактном классе** также можно определить поля и методы, в то же время нельзя создать объект или экземпляр **абстрактного класса**. Абстрактные **классы** призваны предоставлять базовый функционал для **классов**-наследников. А производные **классы** уже реализуют этот функционал.

Абстрактный метод не завершён. Он состоит только из объявления и не имеет тела.

Запись и чтение текстовых файлов в Java

Запись текстовых файлов. Класс FileWriter

Класс FileWriter является производным от класса Writer. Он используется для записи текстовых файлов.

Чтобы создать объект FileWriter, можно использовать один из следующих конструкторов:

- FileWriter(File file)
- FileWriter(File file, **boolean** append)
- FileWriter(FileDescriptor fd)
- FileWriter(String fileName)
- FileWriter(String fileName, **boolean** append)

В конструктор передается либо путь к файлу в виде строки, либо объект File, который ссылается на конкретный текстовый файл.

Параметр append указывает, должны ли данные дозаписываться в конец файла (если параметр равен **true**), либо файл должен перезаписываться.

Запись текстовых файлов в Java

Класс **FileWriter** используется для записи текстовых файлов

```
import java.io.*;

public class Main {
    public static void main(String... args) {

        try (FileWriter writer = new FileWriter("d:\\test.txt", false)) {

            // запись всей строки
            String text = "Hello!";
            writer.write(text);

            // запись по символам
            writer.append('\n');
            writer.append('A');
            writer.flush();

        } catch (IOException ex) {
            System.out.println(ex.getMessage());

        }   }}
```

В конструкторе использовался параметр **append** со значением **false** - то есть файл будет перезаписываться.

Чтение текстовых файлов. Класс FileReader

Класс **FileReader** наследуется от абстрактного класса Reader и предоставляет функциональность для чтения текстовых файлов.

Для создания объекта FileReader мы можем использовать один из его конструкторов:

- FileReader(String fileName)
- FileReader(File file)
- FileReader(FileDescriptor fd)

Чтение текстовых файлов посимвольно в Java

```
import java.io.*;

public class Main {
    public static void main(String... args) {

        try (FileReader reader = new FileReader("d:\\test.txt")) {

            // читаем посимвольно
            int c;
            while ((c = reader.read()) != -1) {

                System.out.print((char) c);
            }

        } catch (IOException ex) {

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

Чтение и запись текстовых файлов по байтам в Java

```
import java.io.*;
public class Main {

 public static void main(String... args) throws IOException {

        FileReader reader = new FileReader("d:\\from.txt");

        FileWriter writer = new FileWriter("d:\\to.txt");

        while (reader.ready()) {

            // читаем по байтам
            int c = reader.read();

            // записываем по байтам
            writer.write(c);
        }

        reader.close();
        writer.close();
    }}
```

Метод **ready**() возвращает истину, если остались непрочитанные символы.

Чтение и запись текстовых файлов по блокам в Java

```
import java.io.*;
public class Main {

 public static void main(String... args) throws IOException {

char[] buff = new char[10];

 FileReader reader = new FileReader("d:\\from.txt");
 FileWriter writer = new FileWriter("d:\\to.txt");

while (reader.ready()) {

    int nSymbols = reader.read(buff);

    writer.write(buff, 0, nSymbols);
}

// Также методу write() можно передать строку:
writer.write("Привет!");

        reader.close();
        writer.close();
    }}
```

Читать и писать можно блоками.

Буферизация символьных потоков

BufferedReader и BufferedWriter

Буферизация символьных потоков

В ряде случаев можно значительно ускорить работу приложения с файлами, воспользовавшись буферизацией потоков Java.

В библиотеке java.io.* предусмотрены классы

BufferedOutputStream и **BufferedInputStream**,

специально предназначенные для буферизации ввода/вывода приложений Java.

Буферизация символьных потоков

BufferedInputStream, BufferedOutputStream, BufferedReader и BufferedWriter— осуществляют буферизацию данных, позволяющую избежать необходимости обращения к источнику (получателю) данных при выполнении каждой отдельной операции read или write.

Доступ к данным на диске выполняется намного медленнее, нежели к информации в буфере памяти, и средства буферизации помогают снизить потребность в обращениях к диску.

Запись текста через буфер и BufferedWriter

Класс **BufferedWriter** записывает текст в поток, предварительно буферизируя записываемые символы, тем самым снижая количество обращений к физическому носителю для записи данных.

Класс BufferedWriter имеет следующие конструкторы:

- BufferedWriter(Writer out)
- BufferedWriter(Writer out, **int** sz)

В качестве параметра он принимает поток вывода, в который надо осуществить запись.

Второй параметр указывает на размер буфера.

Запись текста через буфер и BufferedWriter

```
import java.io.*;


public class Main {

    public static void main(String... args) {

     try(BufferedWriter bw = new BufferedWriter(new FileWriter("d:\\test.txt")))

      {
         String text = "One\nTwo Three\nFour ";

         bw.write(text);
      }

      catch(IOException ex){

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

  }}
```

Чтение текста и BufferedReader

Класс **BufferedReader** считывает текст из символьного потока ввода, буферизируя прочитанные символы. Использование буфера призвано увеличить производительность чтения данных из потока.

Класс BufferedReader имеет следующие конструкторы:

- BufferedReader(Reader in)
- BufferedReader(Reader in, int sz)

В качестве параметра он принимает поток вывода, в который надо осуществить запись.

Второй параметр указывает на размер буфера.

Чтение текста и BufferedReader. Посимвольное чтение

```
import java.io.*;


public class Main {

 public static void main(String... args) {

      try(BufferedReader br = new BufferedReader (new FileReader("d:\\test.txt")))
      {
         // чтение посимвольно
         int c;
         while((c=br.read())!=-1){

            System.out.print((char)c);
         }
      }
      catch(IOException ex){

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


   }
}
```

Чтение текста и BufferedReader. Построчное чтение

```
import java.io.*;


public class Main {

 public static void main(String... args) {

try(BufferedReader br = new BufferedReader(new FileReader("d:\\test.txt")))
{
   //чтение построчно
   String s;
   while((s=br.readLine())!=null){

      System.out.println(s);
   }
}
      catch(IOException ex){

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


   }
}
```

Построчное чтение текстовых файлов в Java

```
import java.io.*;

public class Main {

public static void main(String... args) throws IOException {

        FileReader reader = new FileReader("d:\\from.txt");


        BufferedReader buffReader = new BufferedReader(reader);

        while (buffReader.ready()) {

            // вывод на экран прочитанной строки из файла
            System.out.println(buffReader.readLine());

        }

        reader.close();

        buffReader.close();
    }}
```

Методом **readLine**() класса BufferedReader читать отдельные строки.

Пример. Чтение с консоли и запись в файл

```
import java.io.*;


public class Main {

 public static void main(String... args) {

try(BufferedReader br = new BufferedReader (new InputStreamReader(System.in));

    BufferedWriter bw = new BufferedWriter(new FileWriter("d:\\test.txt")))
{
    // чтение построчно
    String text;
    while(!(text=br.readLine()).equals("ESC")){

        bw.write(text + "\n");
        bw.flush();
    }
}       catch(IOException ex){

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

Здесь объект **BufferedReader** устанавливается для чтения с консоли с помощью объекта new InputStreamReader(System.in). В цикле while считывается введенный текст. И пока пользователь не введет строку "ESC", объект BufferedWriter будет записывать текст файл.

Спасибо за внимание!