Язык программирования JAVA

Потоки ввода/вывода

Содержание

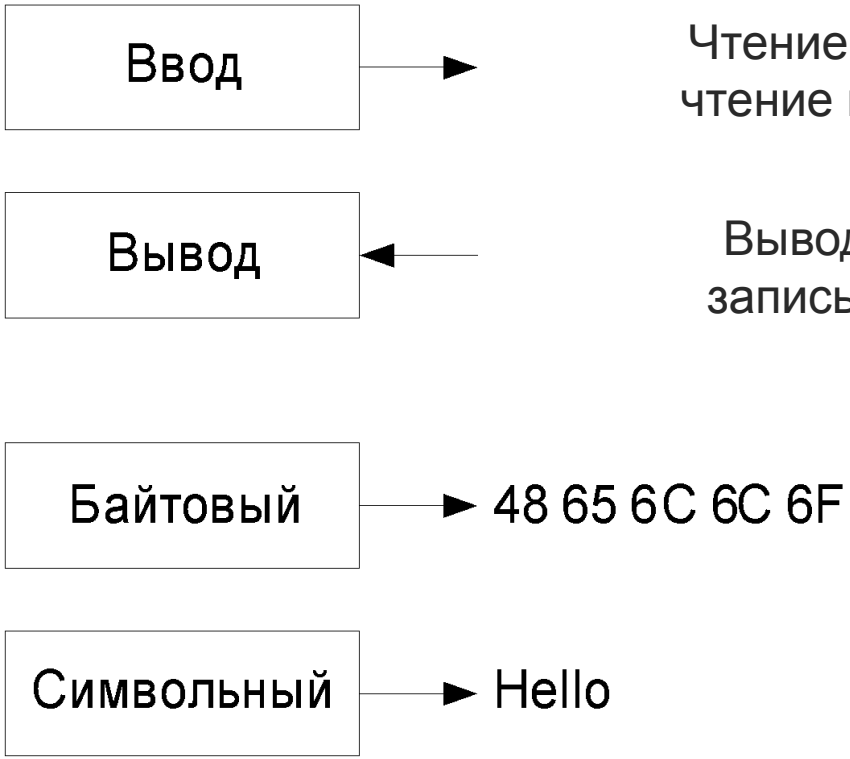
- Поток.
- Байтовые и символьные потоки.
- Базовые классы.
- Блочное копирование.
- Класс File.
- Ввод/вывод на консоль.
- Чтение строк с консоли.
- Сериализация.
- Байтовый в символьный и наоборот.
- Перекодирование файла.

Поток

- Поток является абстракцией, которая или производит, или потребляет информацию.
- Все потоки ведут себя одинаковым образом, хотя фактические физические устройства, с которыми они связаны, могут сильно различаться.
- Благодаря потокам ваша программа выполняет ввод/вывод, не понимая различий между клавиатурой и сетью.
- Поток скрывает детали низкоуровневых процессов ввода/вывода.
- В Java система ввода-вывода представлена в java.io.*

Байтовые и символьные потоки

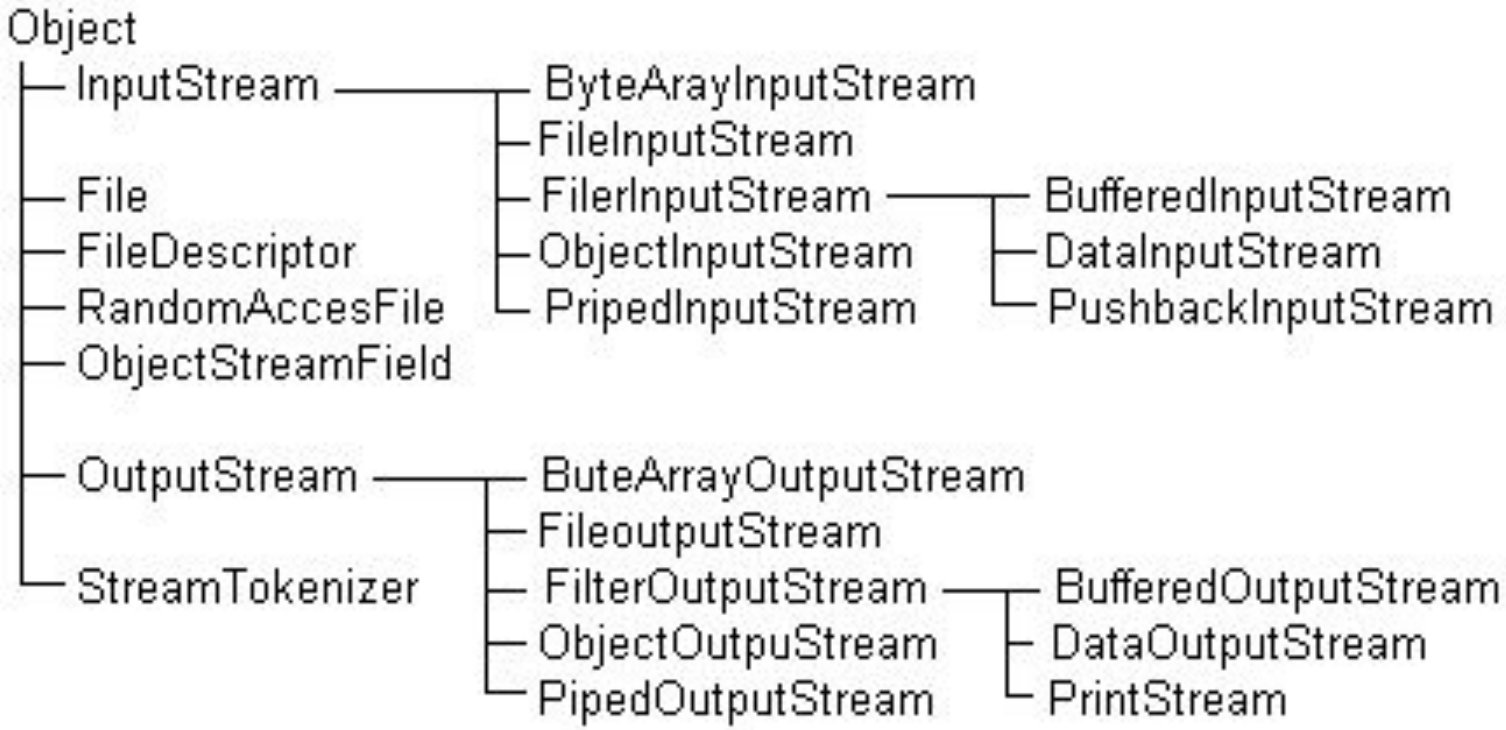
- Потоки бывают для чтения(ввод) и записи (вывод).
- Два типа потоков – байтовый и символьный.



Базовые классы

	Байтовый	**Символьный**
Ввод	InputStream	Reader
Вывод	OutputStream	Writer

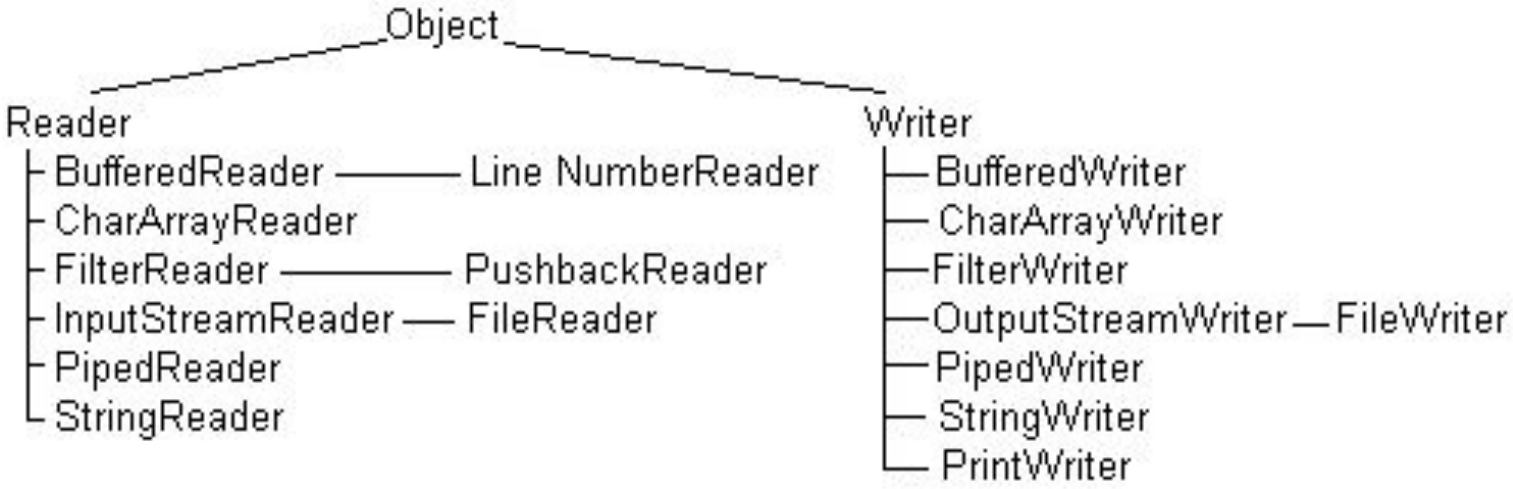
Байтовые потоки



Классы - байтовые потоки

Поточный класс	Значение
BufferedInputStream	Буферизованный поток ввода
BufferedOutputStream	Буферизованный поток вывода
ByteArrayInputStream	Поток ввода, который читает из байт-массива
ByteArrayOutputStream	Поток вывода, который записывает в байт-массив
DataInputStream	Поток ввода, который содержит методы для чтения данных стандартных типов
DataOutputStream	Поток вывода, который содержит методы для записи данных стандартных типов
FileInputStream	Поток ввода, который читает из файла
FileOutputStream	Поток вывода, который записывает в файл
FilterInputStream	Реализует Inputstream
FilterOutputStream	Реализует OutputStream
InputStream	Абстрактный класс, который описывает поточный ввод
OutputStream	Абстрактный класс, который описывает поточный вывод
PipedInputStream	Канал ввода
PipedOutputStream	Канал вывода
PrintStream	Поток вывода, который поддерживает print() и println()
PushbackInputStream	Поток (ввода), который поддерживает однобайтовую операцию "unget", возвращающую байт в поток ввода
RandomAccessFile	Поддерживает ввод/вывод файла произвольного доступа
SequenceInputStream	Поток ввода, который является комбинацией двух или нескольких

Символьные потоки



Классы - символьные потоки

Поточный класс	Значение
BufferedReader	Буферизированный символьный поток ввода
BufferedWriter	Буферизированный символьный поток вывода
CharArrayReader	Поток ввода, которой читает из символьного массива
CharArrayWrite	Выходной поток, который записывает в символьный массив
FileReader	Поток ввода, который читает из файла
FileWriter	Выходной поток, который записывает в файл
FilterReader	Отфильтрованный поток ввода
FilterWriter	Отфильтрованный поток вывода
InputStreamReader	Поток ввода, который переводит байты в символы
LineNumberReader	Поток ввода, который считает строки
OutputStreamWriter	Поток ввода, который переводит символы в байты
PipedReader	Канал ввода
PipedWriter	Канал вывода
PrintWriter	Поток вывода, который поддерживает print() и println()
PushbackReader	Поток ввода, возвращающий символы в поток ввода
Reader	Абстрактный класс, который описывает символьный поток ввода
StringReader	Поток ввода, который читает из строки
StringWriter	Поток вывода, который записывает в строку
Writer	Абстрактный класс, который описывает символьный поток вывода

Класс ByteArrayInputStream

Поток, считывающий (записывающий) данные из массива байт.

```
byte[] bytes ={1,-1,0};
ByteArrayInputStream in =new ByteArrayInputStream(bytes);
int readedInt =in.read();  //readedInt=1
readedInt =in.read();      //readedInt=255 ?
readedInt =in.read();      //readedInt=0
```

Класс ByteArrayOutputStream

Применяется для записи в массив.

```
ByteArrayOutputStream out =new
ByteArrayOutputStream();

out.write(10);

out.write(11);

byte[] bytes =out.toByteArray();
```

Блочное копирование

Процедура копирования

```
void copy(InputStream is, OutputStream os) throws IOException {
    byte[] b = new byte[1024];
    int c = 0;
    while ((c = is.read(b)) >= 0) {
        os.write(b, 0, c);
    }
}
```

Класс File

Позволяет осуществлять манипуляции с файлами и директориями

Создание дескриптора по имени:

- File(pathname) – абсолютный или относительный путь

Создание дескриптора по имени и директории:

- File(File dir, name) File(String dir, name)

Получение информации

- getName() – получить имя
- getPath() – получить имя и путь
- getAbsolutePath() – получить абсолютный путь
- getAbsoluteFile() – получить абсолютный дескриптор

Определение родителя

- String getParent() – как строки
- File getParetnFile() – как дескриптора

Платформозависимые разделители

- separator / separtorChar – разделитель директорий
- pathSeparator / pathSeparatorChar – разделитель в файлов и директорий в путях

Операции с файлами

Проверка типа
- isFile() – является ли файлом
- isDirectory() – является ли директорией
- isHidden() – является ли скрытым

Получение информации о файле
- exist() – проверка существования
- length() – длина файла
- lastModifier() – время последней модификации

Создание
- mkdir() – создать одну директорию
- mkdirs() – создать все директории
- createNewFile() – создать пустой файл

Удаление
- delete() – удалить немедленно
- deleteOnExit() – удалить после завершения

Переименование / перенос
- renameTo(file) – переименовать / перенести в заданное место

Листинг директории

Листинг всех файлов

- String[] list() – получить имена файлов
- File[] listFiles() – получить дескрипторы файлов

Листинг по критерию

- String[] list(FileNameFilter) – получить имена файлов
- File[] listFiles(FileFilter) – получить дескрипторы файлов

Класс FileInputStream

- Поток для чтения из файла по байтам.

```
try {
        FileInputStream fin = new FileInputStream("/etc/hosts");
        try {
            int i = -1;
            while ((i = fin.read()) != -1) {
                System.out.print((char) i);
            }
        } finally {
            fin.close();
        }
    } catch (FileNotFoundException ex) {
        System.out.println(ex.getMessage());
    } catch (IOException ex) {
        System.out.println(ex.getMessage());
    }
```

Класс FileOutputStream

- Поток для записи в файл по байтам.

```
String text = "Hello world!"; // строка для записи
        try {
            FileOutputStream fos = new FileOutputStream("notes.txt");
            try {
                // перевод строки в байты
                byte[] buffer = text.getBytes();
                fos.write(buffer, 0, buffer.length);
            } finally {
                fos.close();
            }
        } catch (FileNotFoundException ex) {
            System.out.println(ex.getMessage());
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
```

Классы BufferedInputStream и BufferedOutputStream

- Чтение и запись данных происходит из буфера.
- Ускоряет процесс в несколько раз.
- Запись в устройство вывода произойдет, когда буфер заполнится.
- метод flush() или close() - сброс и запись данных

```
try {BufferedInputStream bis = new BufferedInputStream(
     new FileInputStream(new File("/etc/hosts")));
          try {
              int c;
              while ((c = bis.read()) != -1) {
                  System.out.print((char) c);
              }
          } finally { bis.close(); }
      } catch (Exception e) {
          System.out.println(e.getMessage());
      }
```

Классы FileReader и FileWriter

```
Writer writer;
 try {
     Reader reader = new FileReader("notes.txt");
     try {
         writer = new FileWriter("output.txt");
         int c = 0;
         while ((c = reader.read()) >= 0) {
             writer.write(Character.toUpperCase((char) c));
         }
         writer.close();
     } finally {
         reader.close();
     }
 } catch (FileNotFoundException e) {
     System.out.println(e.getMessage());
 } catch (IOException e) {
     e.printStackTrace();
 }
```

Байтовый поток в символьный

При чтении возможно преобразование байтового потока в символьный, с указанием кодировки

Класс InputStreamReader
InputStreamReader(InputStream, encoding)

Ввод/вывод на консоль

```
char ch = 0;
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Введите символы, 'q' - для завершения.");
do {
    try {
       ch = (char) br.read();
    } catch (IOException e) {
        e.printStackTrace();
    }
    System.out.print(ch);
} while(ch != 'q');
PrintWriter pw = new PrintWriter(System.out, true);
pw.println("Это строка:");
int i = -7;
pw.println(i);
double d = 4.5e-7;
pw.println(d);
```

Чтение строк с консоли

```
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
List<String> list = new ArrayList<>();
System.out.println("Введите строки текста.");
System.out.println("Введите 'stop' для завершения.");
String s = null;
while(true){
    try {
        s = br.readLine();
    } catch (IOException e) {
        e.printStackTrace();
    }
    if(s.equals("stop")) break;
    list.add(s);
}
System.out.println("Вот ваши строки:");
for (String str :list) {
    System.out.println(str);
}
```

Символьный поток в байтовый

При записи возможно преобразование символьного потока в байтовый, с указанием кодировки

Класс OutputStreamWriter
OutputStreamWriter(OutputStream, encoding)

Пример: перекодирование файла

```
try {
        Reader reader = new InputStreamReader(
                new FileInputStream("input.txt"), "Cp1251");
        Writer writer = new OutputStreamWriter(
                new FileOutputStream("output.txt"), "Cp866");
        int c = 0;
        while ((c = reader.read()) >= 0) writer.write(c);
        reader.close();
        writer.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
```

Сериализация

- Сериализация – процесс преобразования объекта в последовательность байт.
- Сериализованный объект легко передать по сети, сохранить в файл.
- Чтобы объект мог быть сериализован, он должен реализовывать интерфейс java.io.Serializable.
- java.io.Serializable — маркер, нет ни одного метода.
- Классы ObjectInputStream и ObjectOutputStream производят сериализацию и десериализацию объектов.
- Свойства класса, помеченные модификатором transient, не сериализуются.

Пример сериализации

```
class Person implements Serializable {
    public String name = null;
    public int age = 0;
}
ObjectInputStream objectInputStream;
try {
    ObjectOutputStream objectOutputStream = new ObjectOutputStream(
new FileOutputStream("person.bin"));
    try {
        Person person = new Person();
        person.name = "Jakob Jenkov";
        person.age = 40;
        objectOutputStream.writeObject(person);
        objectOutputStream.close();
    } finally {objectOutputStream.close(); }
} catch (FileNotFoundException e) {e.printStackTrace();
} catch (IOException e) {e.printStackTrace();}
```

Пример десериализации

```
try {
    objectInputStream = new ObjectInputStream(new FileInputStream("person.bin"));
    try {
        Person personRead = (Person) objectInputStream.readObject();
        System.out.println(personRead.name);
        System.out.println(personRead.age);
    } finally {
        objectInputStream.close();
    }
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    e.printStackTrace();
}
```

Вопросы?