Пример использования канала. Данные помещаются с одного "конца" и извлекаются с другого.

**package** \_java.\_se.\_04.\_iostream;

**import** java.io.IOException;

**import** java.io.PipedInputStream;

**import** java.io.PipedOutputStream;

**publicclass** PipedStreamExample {

**publicstaticvoid** main(String[] args) {

PipedInputStream pipeIn = **null**;

PipedOutputStream pipeOut = **null**;

**try** {

**int** countRead = 0;

**int**[] toRead = **null**;

pipeIn = **new** PipedInputStream();

pipeOut = **new** PipedOutputStream(pipeIn);

**for** (**int** i = 0; i < 20; i++) {

pipeOut.write(i);

}

**int** willRead = pipeIn.available();

toRead = **new int**[willRead];

**for** (**int** i = 0; i < willRead; i++) {

toRead[i] = pipeIn.read();

System.*out*.print(toRead[i] + " ");

}

} **catch** (IOException e) {

System.*out*.println("Impossible IOException occur: ");

e.printStackTrace();

}

}

}

Последовательность отдельных потоков, которые можно объединить в одном потоке.

SequenceInputStream сливает 2 потока (или более)в единый поток.

**package** \_java.\_se.\_04.\_iostream;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.SequenceInputStream;

**public class** SequenceInputStreamExample {

**public static void** main(String[] args) {

FileInputStream inFile1 = **null**;

FileInputStream inFile2 = **null**;

SequenceInputStream sequenceStream = **null**;

FileOutputStream outFile = **null**;

**try** {

inFile1 = **new** FileInputStream("file 1.txt");

inFile2 = **new** FileInputStream("file 2.txt");

sequenceStream = **new** SequenceInputStream(inFile1, inFile2);

outFile = **new** FileOutputStream("file 3.txt");

**int** readedByte = sequenceStream.read();

**while** (readedByte != -1) {

outFile.write(readedByte);

readedByte = sequenceStream.read();

}

} **catch** (IOException e) {

System.*out*.println("IOException: " + e.toString());

} **finally** {

**try** {

sequenceStream.close();

} **catch** (IOException e) {

}

**try** {

outFile.close();

} **catch** (IOException e) {

}

}

}

}

Пример использования байтовых потоков

**byte**[] bytesToWrite = { 1, 2, 3 };

**byte**[] bytesReaded = **new** **byte**[10];

String fileName = "d:\\test.txt";

FileOutputStream outFile = **null**;

FileInputStream inFile = **null**;

outFile = **new** FileOutputStream(fileName);

System.*out*.println("Файл открыт для записи");

outFile.write(bytesToWrite);

System.*out*.println("Записано: "

+ bytesToWrite.length + " байт");

outFile.close();

System.*out*.println("Выходной поток закрыт");

inFile = **new** FileInputStream(fileName);

System.*out*.println("Файл открыт для чтения");

**int** bytesAvailable = inFile.available();

System.*out*.println("Готово к считыванию: "

+ bytesAvailable + " байт");

**int** count = inFile.read(bytesReaded, 0, bytesAvailable);

System.*out*.println("Считано: " + count + " байт");

inFile.close();

System.*out*.println("Входной поток закрыт");

Пример использования канала.

PipedInputStream pipeIn = **null**;

PipedOutputStream pipeOut = **null**;

**int** countRead = 0;

**int**[] toRead = **null**;

pipeIn = **new** PipedInputStream();

pipeOut = **new** PipedOutputStream(pipeIn);

**for** (**int** i = 0; i < 20; i++) {

pipeOut.write(i);

}

**int** willRead = pipeIn.available();

toRead = **new** **int**[willRead];

**for** (**int** i = 0; i < willRead; i++) {

toRead[i] = pipeIn.read();

System.*out*.print(toRead[i] + " ");

}

Последовательность отдельных потоков, которые можно объединить в одном потоке.

FileInputStream inFile1 = **null**;

FileInputStream inFile2 = **null**;

SequenceInputStream sequenceStream = **null**;

FileOutputStream outFile = **null**;

inFile1 = **new** FileInputStream("file 1.txt");

inFile2 = **new** FileInputStream("file 2.txt");

sequenceStream

= **new** SequenceInputStream(inFile1, inFile2);

outFile = **new** FileOutputStream("file 3.txt");

**int** readedByte = sequenceStream.read();

**while** (readedByte != -1) {

outFile.write(readedByte);

readedByte = sequenceStream.read();

}

Применение символьных потоков

String tmp = "abcdefghijklmnopqrstuvwxyz";

**int** length = tmp.length();

**char** c[] = **new** **char**[length];

tmp.getChars(0, length, c, 0);

CharArrayReader input1 = **new** CharArrayReader(c);

CharArrayReader input2 = **new** CharArrayReader(c, 0, 5);

**int** i;

System.*out*.println("input1 is:");

**while** ((i = input1.read()) != -1) {

System.*out*.print((**char**) i);

}

System.*out*.println();

System.*out*.println("input2 is:");

**while** ((i = input2.read()) != -1) {

System.*out*.print((**char**) i);

}

Еще применение символьных потоков

Writer out = **new** BufferedWriter(

**new** OutputStreamWriter(

**new** FileOutputStream("outfilename"), "UTF8"));

out.write("asdf");

out.close();

Еще пример

String s = "if (a == 4) a = 0;\n";

**char** buf[] = **new char**[s.length()];

s.getChars(0, s.length(), buf, 0);

CharArrayReader in = **new** CharArrayReader(buf);

PushbackReader f = **new** PushbackReader(in);

**int** c;

**while** ((c = f.read()) != -1) {

**switch** (c) {

**case**'=':

**if** ((c = f.read()) == '=')

System.*out*.print(".eq.");

**else** {

System.*out*.print("<-");

f.unread(c);

}**break**;

**default**:

System.*out*.print((**char**) c);**break**;

}

}