**package** javaIO1;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.OutputStream;

/\*\*

\* Данный код копирует файл побайтно удаляя байт 0

\* **@author** JavaUser

\*

\*/

**public** **class** DeleteZero {

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

InputStream in = **new** FileInputStream("C:/Users/JavaUser/Videos/f3m.jpg");

OutputStream out = **new** FileOutputStream("C:/Users/JavaUser/Videos/1\_f3m.jpg", **false**);

**long** startTime = System.*currentTimeMillis*();

**while**(**true**) {

**int** value = in.read();

**if**(value > 0) {

out.write(value);

} **else** **if** (value == -1) {

**break**;

}

}

**long** deltaTime = System.*currentTimeMillis*() - startTime;

in.close();

out.flush();

out.close();

System.***out***.print(deltaTime);

}

}

1. Добавить корректную обработку IOExeption (в любом случае оба потока должны быть закрыты), в случае сбоя записи – удалить недописанный файл (new java.io.File(“…”).delete()) и выкинуть исключение (throw IOException(“…”, e));
2. Добавить буферизацию (BufferedInputStream/BufferedOutputStream) на чтение и запись, в случае больших файлов (более 1Мб) будет заметное увеличение скорости;
3. Перепишите на чтение/ запись массивами – усложнение в том, что вычитав данные в буфер (byte[] buff) необходимо наити в нем ДИАПАЗОНЫ не включающие 0 и делать запись этих диапазонов (write (byte[], int , int));

Рещение 1-2:

**package** javaIO1;

**import** java.io.BufferedInputStream;

**import** java.io.BufferedOutputStream;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.OutputStream;

/\*\*

\* Данный код копирует файл побайтно удаляя байт 0

\* **@author** JavaUser

\*

\*/

**public** **class** DeleteZero {

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

**long** startTime = System.*currentTimeMillis*();

**long** deltaTime;

InputStream in = **null**;

OutputStream out = **null**;

**try** {

in = **new** BufferedInputStream (

**new** FileInputStream ("C:/Users/JavaUser/Videos/f3m.jpg"));

out = **new** BufferedOutputStream (

**new** FileOutputStream ("C:/Users/JavaUser/Videos/1\_f3m.jpg", **false**));

**int** value;

**while**((value = in.read()) != -1) {

**if**(value > 0) {

out.write(value);

}

}

out.flush();

} **catch** (IOException e) {

**new** File("C:/Users/JavaUser/Videos/1\_f3m.jpg").delete();

**throw** **new** IOException("Write error occurred", e);

}

**finally** {

deltaTime = System.*currentTimeMillis*() - startTime;

**if** (in != **null**) {

**try** {

in.close();

} **catch** (IOException e) {

System.***err***.println("InputStream tread has not been closed: " + e.getCause());

}

}

**if** (out != **null**) {

**try** {

out.close();

} **catch** (IOException e) {

System.***err***.println("OutputStream tread has not been closed: " + e.getCause());

}

}

} // finally

System.***out***.print(deltaTime);

} // main (String[]) method

} //DeleteZero class

Решение 1,3:

package javaIO1;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

public class DeleteZeroOnArray {

public static void main (String[] args) throws IOException {

long startTime = System.currentTimeMillis();

long deltaTime;

InputStream in = null;

OutputStream out = null;

try {

in = new FileInputStream ("C:/Users/JavaUser/Videos/f3m.jpg");

out = new FileOutputStream ("C:/Users/JavaUser/Videos/1\_f3m.jpg", false);

readAndCopy(in, out);

} catch (IOException e) {

new File("C:/Users/JavaUser/Videos/1\_f3m.jpg").delete();

throw new IOException("Write error occurred", e);

}

finally {

deltaTime = System.currentTimeMillis() - startTime;

if (in != null) {

try {

in.close();

} catch (IOException e) {

System.err.println("InputStream tread has not been closed: " + e);

}

}

if (out != null) {

try {

out.close();

} catch (IOException e) {

System.err.println("OutputStream tread has not been closed: " + e);

}

}

} // finally

System.out.print(deltaTime);

} // main (String[]) method

private static void readAndCopy(InputStream in, OutputStream out) throws IOException {

byte[] buff = new byte[64];

int count;

while ((count = in.read(buff)) != -1) {

copy(out, buff, count);

}

} //readAndCopy(InputStream, OutputStream method

public static void copy(OutputStream out,byte[] buff, int count) throws IOException {

int writeBy = 0;

int writeTo = 0;

int status = 0;

for(int i = 0; i < count; i++) {

if (buff[i] == 0) {

if (status == 1) {

out.write(buff, writeBy, writeTo);

writeBy = 0;

writeBy = 0;

status = 0;

}

} else if ((buff[i] > 0) && (i == count - 1)) {

out.write(buff, writeBy, count - writeBy);

} else if(buff[i] > 0) {

if(status == 0) {

status = 1;

writeBy = i;

writeTo = 1 + i - writeBy;

} else {

writeTo ++;

}

}

}

} //copy (OutputStream, byte[], int) method

} //DeleteZero class