Файловый ввод и вывод. Текстовые файлы

Запись файлов. Класс FileWriter

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

Пример 1. Запись в файл строки и символов

```
import java.io.*;

public class Main {

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

try(FileWriter writer = new FileWriter("d:\\notes.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());

        }
    }}
```

Часто данные для записи сначала собираются в большие блоки в памяти, а потом только пишутся на диск.

Команда flush требует немедленно записать всю несохраненную информацию на диск.

Чтение файлов. Класс FileReader

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

Пример 2. Чтение текстового файла посимвольно

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

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

try(FileReader reader = new FileReader("d:\\notes.txt"))
{
    // читаем посимвольно
    int c;
    while((c=reader.read())!=-1){

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

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

    }}
```

Пример 3. Чтение и запись по байтам текстового файла

```
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
            int c = reader.read();
            // записываем
            writer.write(c);
        }
        reader.close();
        writer.close();    }}
```

Пример 4. Чтение и запись по байтам текстового файла по блокам

```
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();
    }
}
```

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.

Пример 5. Запись текста через 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());
        }
    }
}
```

Пример 6. 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();
    }
}
```

Пример 7. 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());
       }
    }
}
```

Пример 8. Чтение с консоли (пока не нажата клавиша Esc) 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());
        }    }}
```

Задания

Задания выполнять с помощью BufferedInputStream, BufferedOutputStream.

1. Прочитать строки из текстового файла **task01.txt**, вывести их на экран.

2. Прочитать строки из текстового файла **task01.txt**, вывести длины строк на экран.

3. Прочитать строки из текстового файла **task01.txt**, вывести длину самой длинной строки и саму строку.

4. Прочитать строки из текстового файла **task01.txt.** Вывести сумму длин строк.

5. Вводить строки с помощью клавиатуры, пока не нажата клавиша Esc. Записывать в файл **task05.txt** только те строки, которые состоят из цифр и не начинаются с цифры **0**. Используйте регулярные выражения.

6. Посчитать сумму чисел, записанных в файл **task05.txt**. Вывести результат на экран. Преобразовать строку в целое число можно с помощью

```
int x=Integer.parseInt("123");
```

7. Задать целое случайное число n<=9, и сформировать n строк случайной длины (длина не больше 60 символов) из символов [A-Z]. Каждую строку записывать в отдельный файл. Имена файлам задавать согласно их номеру – 1.txt, 2.txt, …