Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Чувашский государственный университет имени И.Н. Ульянова»

ФАЙЛЫ С ПРОИЗВОЛЬНЫМ ДОСТУПОМ И ФАЙЛЫ ДЛЯ ЗАПИСИ ОБЪЕКТОВ

Назарова Ольга Васильевна

Файлы с произвольным доступом

В языке Java файл с произвольным доступом к данным реализуется с помощью класса **RandomAccessFile**.

Один из базовых методов **seek ()** позволяет переместиться к требуемой позиции в файле.

При записи или чтении из файла данных известного числового типа всегда можно рассчитать указатель на конкретный элемент (на номер байта).

Например, для файла с данными типа double (8 байт):

номер элемента 0 – позиция указателя 0*8= 0;

номер элемента 1– позиция указателя 1*8= 8;

номер элемента 2 – позиция указателя 2*8= 16 и т.д.

Перевод курсора на 2-й элемент и его считывание осуществляется следующим и операциями:

```
file.seek (2*8); //перемещение
file.readDouble (); //считывание
```

Перевод курсора в конец файла и запись нового числа:

```
file.seek (file.length());  //length () дает позицию конца файла
file.writeDouble  (x); //x – число для записи
```

В данном примере file – переменная типа **RandomAccessFile**, связанная с конкретным файлом на диске.

Название метода	Тип	Выполняемые действия
read(), **read**(byte b[]),	int	Читает один байт или массив байтов
skipBytes(int n)	int	Пропускает *n* байт. Т.е. перемещает указатель на *n* байт вперед
write(int b)	void	Пишет один байт в то место, где стоит указатель
write(byte b[])	void	Пишет массив байтов в то место, где стоит указатель
write(byte b[], int off, int len)	void	Пишет массив байтов в то место, где стоит указатель
getFilePointer()	long	Возвращает номер байта, на который указывает «указатель»
seek(long pos)	void	Перемещает указатель, используемый для чтения/записи, в указанное место pos= номер_элемента * размер_элемента_в_байтах, соответствующее его типу данных
length()	long	Возвращает длину файла в байтах
setLength(long newLength)	void	Устанавливает новую длину файла. Если файл был больше – он обрезается, если меньше – расширяется и новое место заполняется нулями
readBoolean(), **readByte**() , **readChar**(), **readInt**(), **readLong**(), **readFloat**(), **readDouble**() , **readLine**()	-	Читает число/строку/символ соответствующего типа данных с текущей позиции указателя в файле
close()	void	Закрывает файл
writeByte(int v), **writeInt**(int v), **writeLong**(long v), **writeBytes**(String s), **writeChar**(int v), **writeUTF**(str)	void	Пишет число/строку/символ соответствующего типа данных с текущей позиции указателя в файле

Понятия сериализации и десериализации

Записать в файл или прочитать из файла информацию об объектах, в том числе объектах собственных классов сложнее, чем данных примитивных и символьных типов. Для этой цели используются классы ***ObjectOutputStream*** (наследник *OutputStream*), ***ObjectInputStream*** (наследник *InputStream*)и механизм сериализации.

Сериализация – это процесс сохранения состояния объекта в последовательность байтов; десериализация – процесс восстановления объекта из этих байтов.

Для того чтобы иметь возможность сохранить объект в байтовый поток, необходимо чтобы класс, на базе которого объект создан, реализовал стандартный интерфейс java.io.Serializable.