Назначение лекции

Знакомство с особенностями записи текстовых данных в файл.

Новые слова

**Файл** – именованная структура хранения данных на внешнем носители, например на диске или флешке.

Что было сделано

В предыдущей работе наше приложение MyEvent научилось выдавать список доступных режимов работы и выполнять контроль правильности номера режима. Если номер был введен не верно, то ввод повторялся.

Далее рассмотрели реализацию режима «Ввод» в части ввода текущей даты и названия события.

Рассмотрели реализацию режима ввода в части ввода количества времени на событие (вещественное число) и состояния события. В программе реализовано два состояния: «было» и «будет». Для состояния «будет» понадобится дата этого события.

Разработан механизм ввода даты для будущего события.

В этой лекции разберем механизм записи данных о событии в файл в текстовом виде.

Напомним, что в режиме «Ввод» приложение вводит следующие данные:

1. Текущая дата
2. Название события
3. Количество времени события
4. Состояние (было или будет)
5. Дата окончания события для состояния «будет»

Таких событий может быть много. Целесообразно их записывать в файл или базу данных. Работа с базой данных – это отдельная тема и пока этот вопрос не рассматривается. В этом курсе организуем подобие базы данных в файле.

Продолжение

Файл это структура хранения данных. Файл содержит имя и расширение. Обычно расширение определяет тип данных файла. В нашем случае это файл *myEvent* с расширением *txt*. Расширение *txt* означает что файл содержит текст и его можно просматривать (но не редактировать) в текстовом редакторе, к сожалению не в любом.

Структуру файла определим следующим образом:

- первая строка содержит название приложения, например, “Программа <MyEvent>. Автор Касымов Д.Д.”

- все последующие строки - это данные о событиях, например,

Программа <MyEvent>. Автор Касымов Д.Д.

16-06-2022

Учеба

5.0

Было

16-06-2022

16-06-2022

Спорт

3.0

Будет

25-06-2022

16-06-2022

Развлечения

2.0

Было

16-06-2022

16-06-2022

Питание

2.0

Было

16-06-2022

17-06-2022

Дорога

2.0

Было

17-06-2022

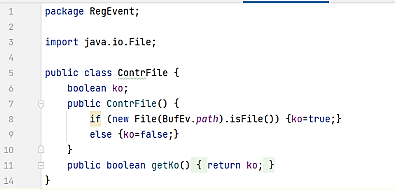
На каждое событие отводится пять строк.

Итак, работа с нашей базой должна иметь следующие механизмы:

1. Контроль наличия файла
2. Первоначальное создание файла
3. Запись текущего события в файл

Контроль наличия файла.

Для контроля создадим класс ContrFile



import java.io.File;

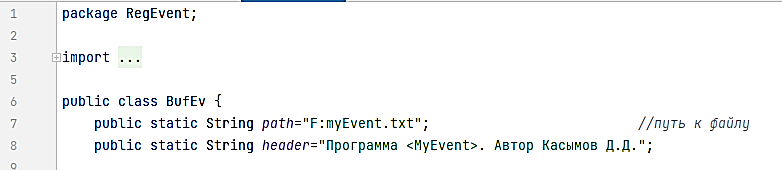
Импорт стандартной библиотеки для ввода-вывода файла

boolean ko;

Внетренняя переменная для кода ответа *ko*. True – файл есть, false - нет

if (new File(BufEv.*path*).isFile()) {ko=true;}

Проверка (*isFile*) наличия файла из переменной *path*, которая находится в классе *BufEv*. Следует дописать переменные (остальное оставить неизменным):



Путь к файлу в данном случае указывает на «флешку» F.

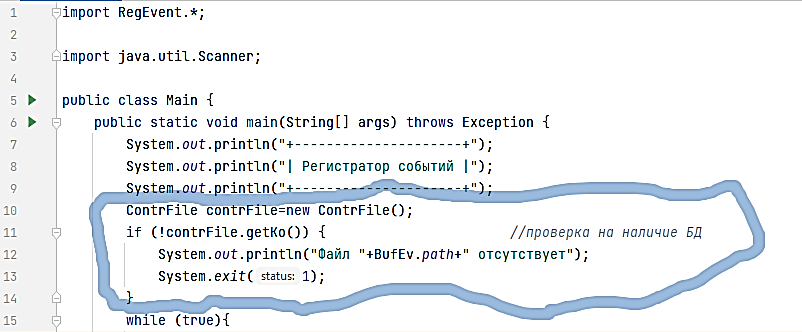
else {ko=false;}

Вариант отсутствия файла из *path* определяется как *ko=false*

public boolean getKo() { return ko;}

Метод getKo для возврата *ko*

Контроль наличия файла необходимо выполнить в начале работы, в классе *Main*



ContrFile contrFile=new ContrFile();

Создание объекта *contrFile* класса *ContrFile*

if (!contrFile.getKo()) { *//проверка на наличие БД*

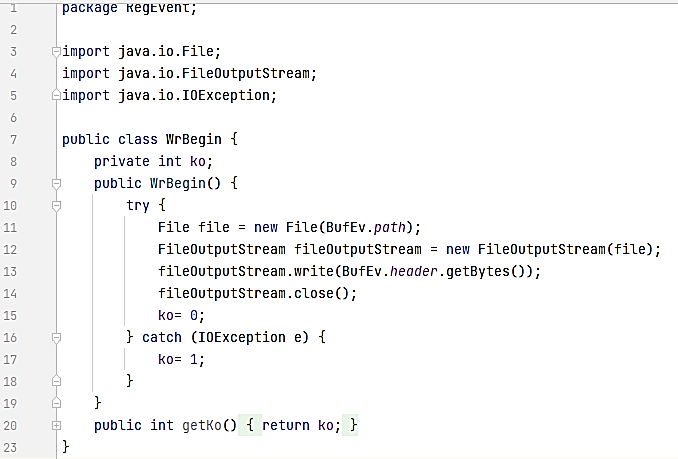
Проверка наличия файла через метод *getKo*

System.*out*.println("Файл "+BufEv.*path*+" отсутствует");  
System.*exit*(1);

Если файла нет то сообщение об ошибке и окончание работы с кодом 1.

Первоначальное создание файла

Новый класс для первоначального формирования файла:



private int ko;

Внутренняя переменная *ko* для возврата кода ошибки

try {

Начало блока, где может быть аварийная ситуация (исключение)

File file = new File(BufEv.*path*);

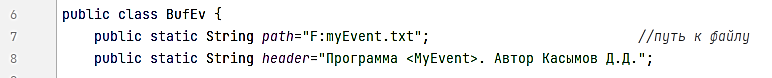
Создание объекта *file* из стандартного класса *File* и указание пути и имени файла BufEv.*path* . Этот объект необходим для записи данных.

FileOutputStream fileOutputStream = new FileOutputStream(file);

Открытие файла по указаному имени и пути, если такого нет то создается новый файл.

fileOutputStream.write(BufEv.*header*.getBytes());

Запись (*getBytes*) в открытый файл заголовка *header* из класса *BufEv*. Эти данные мы уже вставили ранее:



fileOutputStream.close();

Закрытие файла. Если этого не сделать, то данные записаны не будут.

ko= 0;

Формирование кода ответа = 0. Т.е. запись выполнена верно.

} catch (IOException e) {ko= 1;}

Вариант для аварийной ситуации с *ko=1*.

public int getKo() {return ko;}

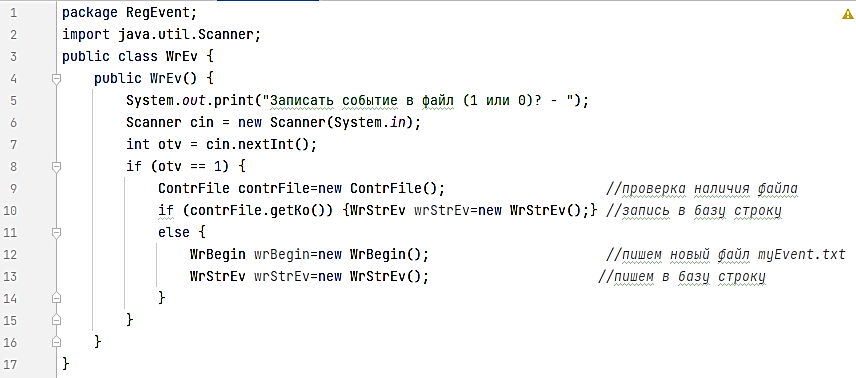
Метод класса для возврата кода ответа.

Запись текущего события в файл

Далее можно записать в созданный файл данные о введенном текущем событии. Они уже записаны в классе BufEv:

public static String *currDate*; *//текущая дата*public static String *nameEv*; *//текущее название события*public static String *timeEv*; *//время события*public static String *nameSos*; *//текущее имя состояния*public static String *futureDate*; *//дата будущего события*

Новый класс для записи события:



System.*out*.print("Записать событие в файл (1 или 0)? - ");

Запрос на запись события в файл. 1 - записать, 0 - не записывать.

Scanner cin = new Scanner(System.*in*);

Открытие переменной *cin* для записи ответа.

int otv = cin.nextInt();

Ответ переводим в целое число и записываем в *otv*

if (otv == 1) {

Сравнение ответа на 1

ContrFile contrFile=new ContrFile(); *//проверка наличия файла*

Проверяем на всякий случай наличие файла.

if (contrFile.getKo()) {WrStrEv wrStrEv=new WrStrEv();}*//запись в базу строку*

Если *ко=0*, то записываем событие (строку) в файл. Класс *WrStrEv* еще не создан.

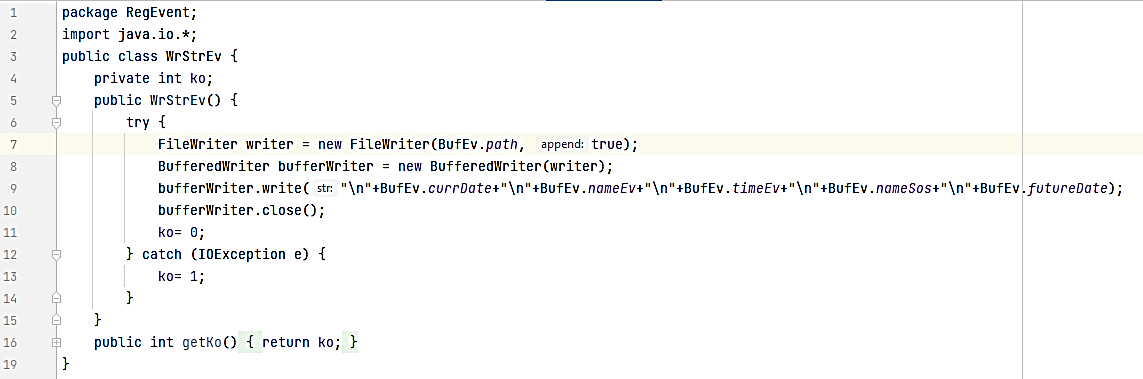
else {

Если файла еще нет, то…

WrBegin wrBegin=new WrBegin(); *//пишем новый файл myEvent.txt*WrStrEv wrStrEv1=new WrStrEv(); *//пишем в базу строку*

Создание файла и запись события в файл.

Далее реализуем основную часть. До этого была подготовка. Создаем новый класс для дозаписи одного события:



private int ko;

Внутренняя переменная с кодом ответа.

try {

Начало блока, где может быть исключение

FileWriter writer = new FileWriter(BufEv.*path*, true);

Открытие объекта *writer* из стандартного класса *FileWriter* для нашего файла, на дозапись (*true*)

BufferedWriter bufferWriter = new BufferedWriter(writer);

Открытие буфера для файла. Говорят, что так работает быстрее.

bufferWriter.write("\n"+BufEv.*currDate*+"\n"+BufEv.*nameEv*+"\n"+BufEv.*timeEv*+"\n"+BufEv.*nameSos*+"\n"+BufEv.*futureDate*);

Дозапись в файл, с указанием перехода на новую строку (\n).

bufferWriter.close();

Закрытие файла. Это обязательно.

ko= 0;

Правильное завершение записи

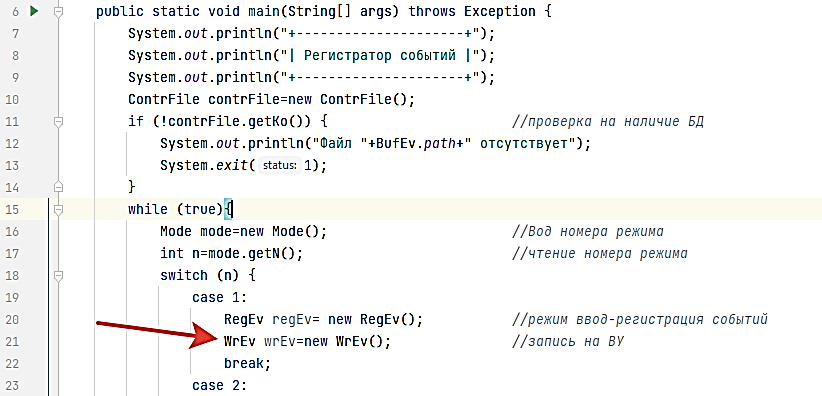
} catch (IOException e) {ko= 1;}

Неправильное завершение записи.

public int getKo() {return ko;}

Метод доступа к внутренней переменной *ko*

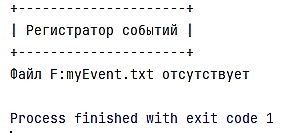
И в итоге изменим код класса *Main*:



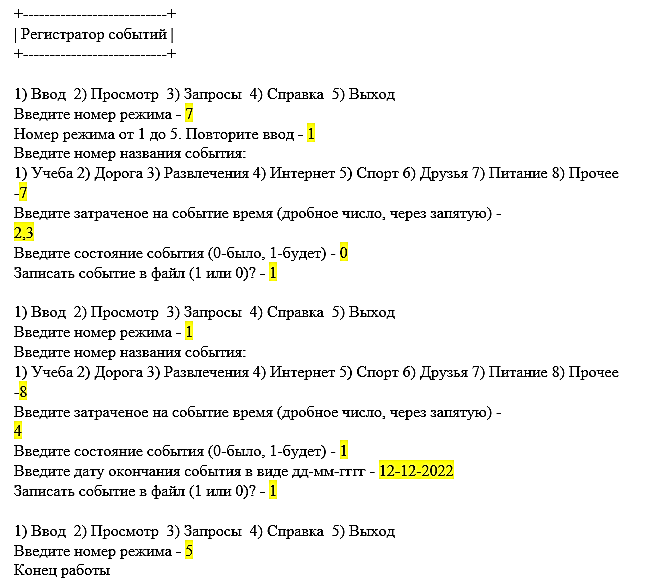
Этим изменением мы инициируем запись события в файл.

Протестируем нашу работу:

Файл с базой отсутствует:



Файл с базой уже есть:



Результат записи в файл:

Программа <MyEvent>. Автор Касымов Д.Д.

20-07-2022

Питание

2.3

Было

20-07-2022

20-07-2022

Прочее

4.0

Будет

12-12-2022

Конец лекции