Назначение лекции

Знакомство с особенностями вывода на экран текстовых данных из файла.

Новые слова

**Шапка –** понятие из бухгалтерии, связанное с отчетами в табличной форме. Означает название столбцов в таблице.

**Терминал** – старое название устройства для ввода и отображения данных. Входит монитор и клавиатура.

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.

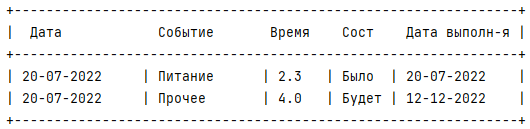
Разработан механизм ввода даты для будущего события.

Разработали механизм записи данных о событии в файл в текстовом виде.

Реализовано чтение, ранее записанных событий, из файла и запись в массив.

Продолжение

В этой лекции подробно рассмотрим вывод на экран текстовых данных из массива в следующем виде:



Структура выдачи следующая:

Шапка



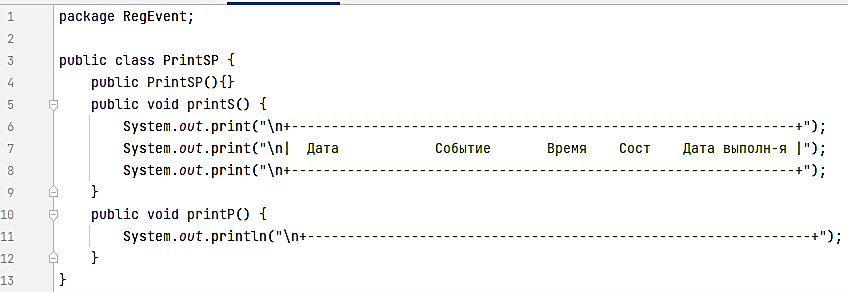
Строка данных:



И завершающая строка:



Создадим маленький класс для выдачи шапки и завершающей строки. Почему отдельный класс? Для того что бы исправления выполнять в одном месте и использовать в других классах.



public PrintSP(){}

Это конструктор, но пустой. Так можно делать.

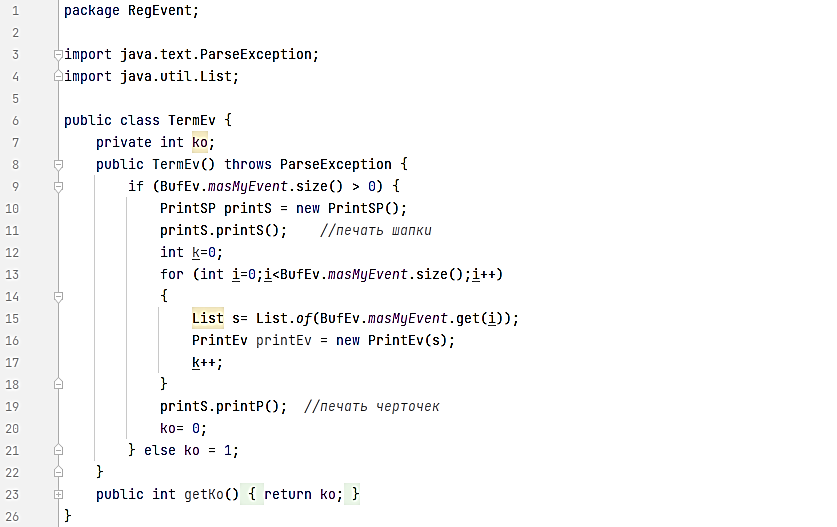
public void printS() {  
 System.*out*.print("\n+---------------------------------------------------------------+");  
 System.*out*.print("\n| Дата Событие Время Сост Дата выполн-я |");  
 System.*out*.print("\n+---------------------------------------------------------------+");  
}

Метод *printS* для печати шапки.

public void printP() {  
 System.*out*.println("\n+---------------------------------------------------------------+");  
}

Метод *printP* для печати заключительной строки (для красоты).

Разработаем еще один класс для выдачи данных на терминал из массива:



private int ko;

Локальная переменная для кода ответа

public TermEv() throws ParseException {

Конструктор класса с возможным исключением ParseException

Это исключение, которое может возникнуть при преобразовании или получении данных.

if (BufEv.*masMyEvent*.size() > 0) {

Проверка, на всякий случай, на пустой массив. Если не пусто то:

PrintSP printS = new PrintSP();

Создание нового объекта класса PrintSP

printS.printS(); *//печать шапки*

Выдача на экран шапки в виде:



for (int i=0;i<BufEv.*masMyEvent*.size();i++)

Цикл по элементам массива *masMyEvent* (каждый элемент содержит 5 строк). Size – размер массива.

List s= List.*of*(BufEv.*masMyEvent*.get(i));

Выборка (get) в список *s* очередного *i*-го элемента из массива *masMyEvent*

PrintEv printEv = new PrintEv(s);

Печать (выдача) списка *s* через класс *PrintEv* (рассмотрим чуть ниже). *s* –параметр, который содержит данные очередного элемента.

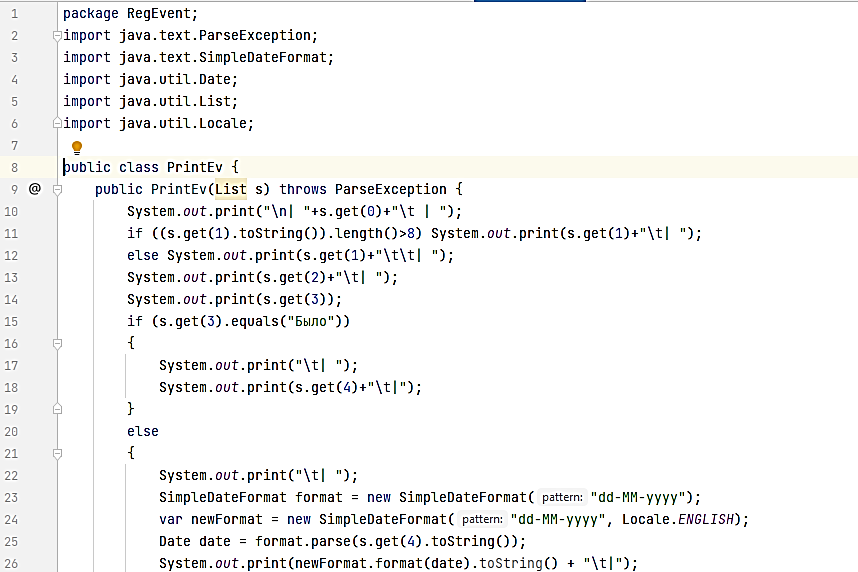
printS.printP(); *//печать черточек*

Печать заключительной строки черточек через метод *printP* класса *PrintSP*

public int getKo() {return ko;}

Метод класса для возврата кода ответа.

Далее создадим класс *PrintEv* в котором и непосредственно выполняется печать, вернее выдача на экран:





Заложен следующий алгоритм:

Из списка s выбираются (*get*) дата (*s.get(0)*), название события (*s.get(1)*), количество затраченного времени (*s.get(2)*), статус события (*s.get(3)*). Если статус или состояние «Было», то просто читаем дату (*s.get(4)*), а если «Будет», дату следует преобразовать в стандартный вид. И теперь более подробно:

System.*out*.print("\n| "+s.get(0)+"\t | ");

Чтение из списка элемента с индексом ноль (текущая дата). \t – это знак табуляции.

if ((s.get(1).toString()).length()>8) System.*out*.print(s.get(1)+"\t| ");

Если длина имени события больше 8, то выдаем с табуляцией

else System.*out*.print(s.get(1)+"\t\t| ");

Иначе с двумя табуляциями. Это определили методом подбора.

System.*out*.print(s.get(2)+"\t| ");

Выдача количества времени на событие

System.*out*.print(s.get(3));

Выдача состояния (статуса)

if (s.get(3).equals("Было"))

Проверка на состояние “Было”

System.*out*.print("\t| ");  
System.*out*.print(s.get(4)+"\t|");

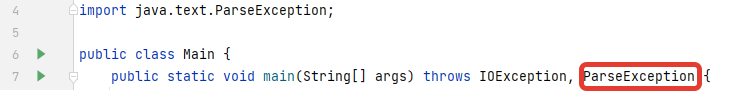
Выдача даты для состояния “Было”. Совпадает с датой ввода.

System.*out*.print("\t| ");  
SimpleDateFormat format = new SimpleDateFormat("dd-MM-yyyy");  
var newFormat = new SimpleDateFormat("dd-MM-yyyy", Locale.*ENGLISH*);  
Date date = format.parse(s.get(4).toString());  
System.*out*.print(newFormat.format(date).toString() + "\t|");

Выдача даты для состояния “Будет”. Эта дата была введена вручную и по формату отличается от даты ввода. Сначала берется дата по формату *format*. Затем объявляется новый формат *newFormat* по локали *ENGLISH,* а затем дата будущего события формата *format* печатается по формату *newFormat.*

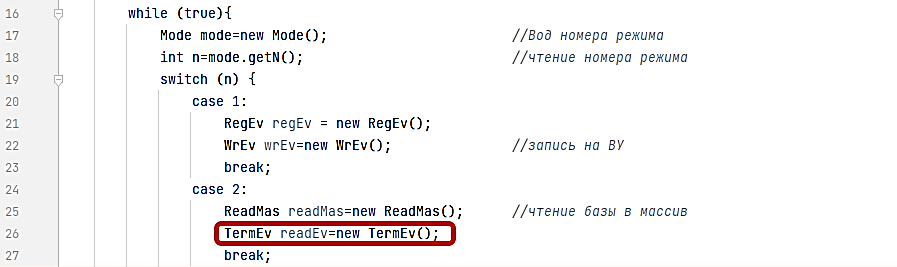
И далее изменим код класса Main.

Для возможного исключения при преобразовании даты:



В противном случае может быть сообщение об ошибке.

И далее вставим код выдачи событий на экран (терминал), для чего создадим новый объект класса *TermEv*:



Конец лекции