Назначение лекции

Вы познакомитесь с особенностями формирования и использовании новых классов на примере учебного десктопного приложения Java.

Новые слова

**Класс** – это программа-шаблон для последующего создания объекта, принадлежащего этому классу. В классе описываются свойства и методы.

**Объект** – объект это реализация класса. Объект наследует все свойства и методы родительского класса.

**Пакет** – объединение нескольких классов под одним названием.

# Введение

В первой лекции, говорилось о том, что программа или система на языке Java это взаимосвязанный набор объектов.

Каждый объект это конкретный экземпляр одного из классов. Классы могут быть разработаны или взяты готовыми из соответствующей библиотеки.

Готовых библиотек классов очень большое количество, они хорошо описаны и важно для программиста знать и уметь использовать эти библиотеки. Не следует изобретать велосипед при разработки новых классов, например, разрабатывать свою сортировку, которая давно реализована.

Но готовые библиотеки классов содержат методы общего характера, например, преобразование числа в строку, работу с датами, ввод-вывод с клавиатуры или запись-чтение файла, работу с базами данных или web-технологии. А для конкретного приложения таких возможностей конечно не достаточно.

Поэтому программист вынужден разрабатывать свои классы, с тем, что бы потом объявлять объекты, экземпляры этих классов.

Напомним, что в классе описываются все свойства, поля и методы (подпрограмы), которые будут использоваться в объекте - экземпляре этого класса.

В теории объектно-ориентированного программирования приняты следующие условные приемы:

1. Классы – это предметы (существительные)
2. Методы – действия (глаголы)
3. Свойства – прилагательные

Для знакомства с объектно-ориентированными возможностями Java реализуем учебный проект. Проект будет охватывать множество классов, на основе которых будут создаваться объекты, выполняющие все задачи учебного приложения. В процессе реализации, студенты познакомятся с особенностями программирования на Java на консольном уровне.

Этот курс дает навыки объектно-ориентированного программирования на Java в достаточном объеме, что бы приступить к самостоятельной работе.

Конечно этот объем не может охватить все возможности. Их очень много, и они существенно различаются по назначению: финансовые приложения, web-программирование, мобильные приложения, для задач искуственного интеллекта и пр.

# Описание учебной прикладной системы

Учебная прикладная система представляет собой регистратор событий человека (например, студента).

Имеется стандартный справочник названий событий:

Учеба,

Дорога,

Развлечения,

Интернет,

Спорт,

Друзья,

Питание,

Прочее.

Для простоты этот справочник пополнять нельзя в режиме онлайн.

Потребуется указывать состояние каждого события: «было» (по умолчанию), «будет».

Каждое событие характеризуется:

1. датой ввода,
2. названием,
3. количеством часов и минут этого события
4. состоянием,
5. датой выполнения (для состояния «будет»)

В системе можно использовать запросы:

1. Сколько времени потрачено на указанное событие за период
2. События с максимальным временем за период
3. Событие с минимальным временем временем за период
4. Какие события «будут» (после текущей даты)

# Проектирование

Как назовем нашу систему? Это довольно важный момент, т.к. от удачного названия часто много зависит в дальнейшем. Для нас это не столь важно, поэтому назовем ее: **MyEvent**

Предметы (существительные)

Система - **Main**

Справочник (**перечисление**) наименований событий – **DirEnum**

Список режимов - **Mode**

Регистратор событий – **RegEv**

Контроллер файла – **ContrFile**

Дата регистрация события **– DateReg**

Дата будущего события - **FutureDate**

Справка **– Help**

Принтер справки - **PrintHelp**

Читатель режима – **Mode**

Читатель события – **ReadEv**

Писатель события **– WrEv**

Принтер событий **- PrintEv**

Запрос – **Request**

Принтер результатов запроса **– PrintEvReq**

и другие…

Действия (глаголы)

Чтение режима – **GetMode**

Запись режима - **SetMode**

Чтение события – **GetReg**

Запись события – **SetReg**

Получить начало периода - **getBegPeriod**

Получить конец периода – **getEndPeriod**

Печать топика справки – **printTop**

Печать результатов запроса - **printE**

и другие …

Свойства (прилагательные)

Названия режимов работы **- textMode**

Названия запросов - **textReq**

Номер *выбранного* режима – **nMode**

*Текущая* дата – **CurrDate**

Время *затраченное* на событие – **TimeEv**

Дата *будущего* события – **FuturуDate**

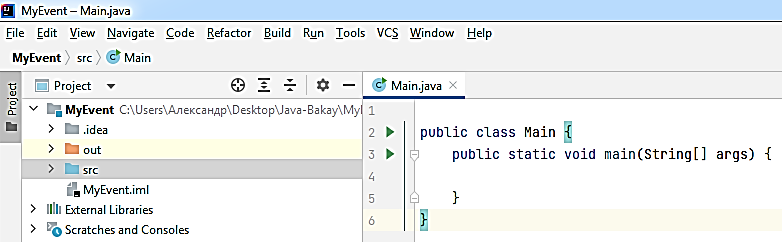
Номер заданного топика справки – **nTop**

и другие …

# Реализация проекта RegEvent

Реализацию проекта будем выполнять по частям. В каждой части – дополнительные классы, объекты и функции.

В начале создаем новый проект **MyEvent:**

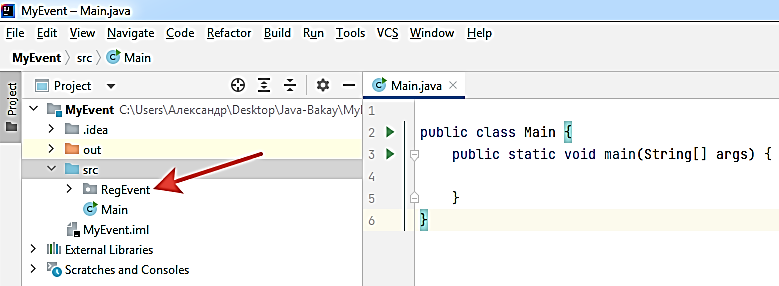
****

В отличии от с++ процедура Main является классом. Класс обозначается латинскими буквами и начинается с большой буквы.

Внутри класса автоматически вставляется конструктор (программа внутри программы), которая называется так же как класс, но начинается с маленькой буквы. В конструкторе имеется параметр args – строковый массив.

# Формирование нового пакета RegEvent

Для удобства можно сформировать новый пакет **RegEvent**, где будут находиться все наши будущие классы:



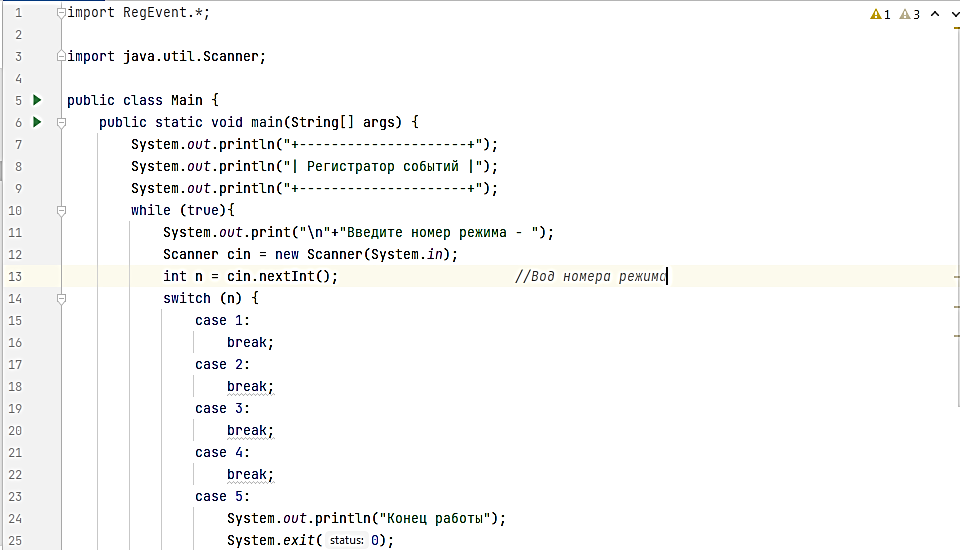
Как было указано ранее прикладное решение Java – это набор объектов, которые взаимодействуют друг с другом. Что бы в каждом классе не перечислять все участвующие классы, можно поместить их в один пакет. После этого достаточно указывать название этого пакета.

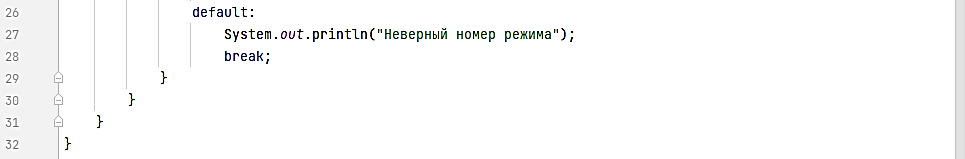
В любой системе полный вариант системы в коде реализовать затруднительно. Часто в качестве главной программы (головной программы) задают шаблон, где только намечены все действия главного меню. А затем, постепенно, по мере разработки, подключают новые блоки.

В нашем случае, должно выдаваться название системы и выполнятся запрос на ввод номера режима работы. Для консольных приложений работа с номерами более удобна.

Примечание. Если бы использовался не консольный режим java, а «оконный», то действия визуально выглядели бы как набор кнопок с названиями.

Рассмотрим подробно шаблон системы, который будет расположен в конструкторе main.





**Анализ кода**

**import RegEvent.\*;**

Заказ пакета RegEvent. Эта команда появляется автоматически, после создания пакета, хотя ничто не мешает выполнить это и вручную.

**import java.util.Scanner;**

Заказ библиотеки java для организации ввода с клавиатуры. Это стандартная библиотека, которая входит в состав java. Автоматически вставляется после написания команд ввода.

**System.*out*.println("+---------------------+");  
System.*out*.println("| Регистратор событий |");  
System.*out*.println("+---------------------+");**

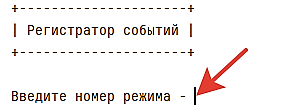
Выдача на экран названия нашего приложения.

**while (true){**

Начало цикла, в котором будет выполнятся ввод номера режима в виде целого числа. Цикл ввода будет выполнятся, до тех пор, пока не встретится команда выхода из цикла.

**System.*out*.print("\n"+"Введите номер режима - ");**Происходит переход на следующую строку: **"\n"**

На экран выдается приглашение: **"Введите номер режима - "** и курсор остается в конце строки, после тире, т.к. используется print, а не println:



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

Выполняется ввод с клавиатуры, как поток символов в объект (поток) cin, который относится к классу Scanner. Параметр **System.*in*** для конструктора Scanner означает, что выполняется именно ввод с клавиатуры.

Объект **cin** может иметь и другое допустимое имя!

**int n = cin.nextInt();**

Из потока cin считываются введенные данные, преобразуются в целое число и это число помещается в переменную **n**

Далее число **n** анализируется с помощью переключателя switch.

**switch (n) {**

где n – номер режима работы приложения.

**case 1:  
 break;  
 case 2:  
 break;  
 case 3:  
 break;  
 case 4:  
 break;  
 case 5:  
 System.*out*.println("Конец работы");  
 System.*exit*(0);  
 default:  
 System.*out*.println("Неверный номер режима");  
 break;  
}**

Выполняется проверка на номера режима 1, 2, 3, 4, 5 и прочие. С 1 по 4 режима пока отсутствуют, поэтому после case стоит break – это означает, что дальнейшую проверку выполнять не требуется.

Режим 5 – это завершение работы программы, поэтому выдается сообщение:

**System.*out*.println("Конец работы");** послечего выполняется выход из программы **System.*exit*(0); ,** где ***exit*(0)** ноль означет выход с кодом ответа ноль. Т.е. считаем, что ноль – это нормальный код ответа.

**default:  
 System.*out*.println("Неверный номер режима");  
 break;**

Любой номер режима, отличный от 1, 2, 3, 4, 5, считается ошибкой, для этого используется команда **System.*out*.println("Неверный номер режима");** и цикл **while (true){** выполняется вновь.

**}**

Конец переключателя switch (n) {

**}**

Конец цикла while (true){

**}**

Конец конструктора public static void main(String[] args) {

**}**

Конец класса public class Main {

В данной реализации шаблона главной программы имеется существенный недостаток, а именно, если пользователь введет вместо номера букву или служебный символ, то программа “вылетит”:



В реальном приложении такие недостатки недопустимы. Как это исправить, рассмотрим на следующих занятиях.

Конец лекции