Назначение лекции

Вы познакомитесь с особенностями ввода целых чисел с контролем и созданием нового класса.

Новые слова

**Модификатор** – понятие в ООП которое определяет доступность данных из других объектов.

**public** – модификатор для общедоступных данных из всех объектов приложения

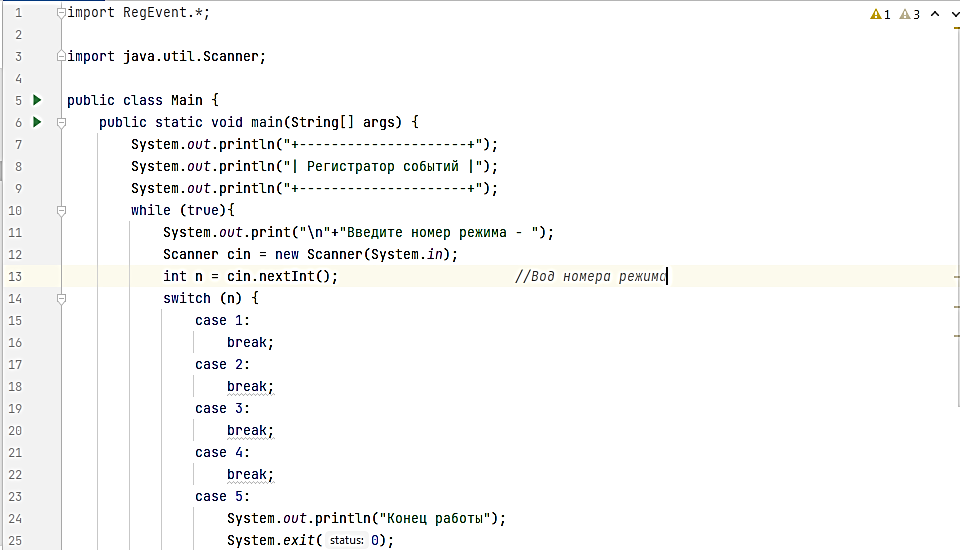
**static –** модификаторозначает, что можно обращаться к переменным с указанием имени класса, не создавая объекта. Т.е. данные с таким модификатором принадлежат классу, а не объекту.

Что было сделано

В нашем приложении **MyEvent** был создан шаблон для ввода и обработки режима работы.

Продолжение

Продолжим разбор кода, в котором будет реализован ввод и контроль номера режима, т.к. прежний контроль был недостаточен. Вспомним код в классе Main для ввода номера режима:



Конец лекции

В данном варианте программа аварийно завершает свою работу после ввода вещественного числа или буквы. Программист не должен расчитывать на аккуратность пользователей. Разработчик должен предполагать самые невероятные варианты. Поэтому в нашем случае необходимо предусмотреть вариант с вещественым числом и буквой.

Для контроля ввода, программист может самостоятельно выболнить разбор введенных данных, выполнить их анализ и решить вопрос о допустимости данных в качестве номера режима. Все это запрограммировать. И это не совсем тревиальная задача.

Есть и второй вариант: реализовать механизм исключений. Он заключается в перехвате аварийной ситуации и самостоятельной обработке этого события. Например, выдачи своего сообщения на экран и повторный ввод.

Третий вариант: использование стандартных функций Java для контроля введенных данных. Этим вариантом и воспользуемся.

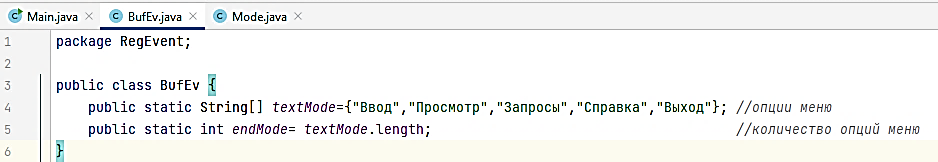
Класс BufEv

Для ввода и контроля разработаем отдельный класс Mode и дополнительный класс BufEv.

Класс BufEv будет хранить некоторые общие данные с модификаторами **public** и **static**

**public** – общедоступные данные для всех объектов приложения

**static –** означает, что можно обращаться к переменным с указанием имени класса, не создавая объекта. Т.е. данные с таким модификатором принадлежат классу, а не объекту.



Переменная **textMode** – массив строк, где каждая строка это название режима. Режимов пять, поэтому размерность массива тоже пять. Известно, что каждый элемент массива имеет индекс 0, 1, 2 и т.д. Указывая это индекс можно прочитать элемент массива.

Может возникнуть **вопрос**: зачем создали такой массив.

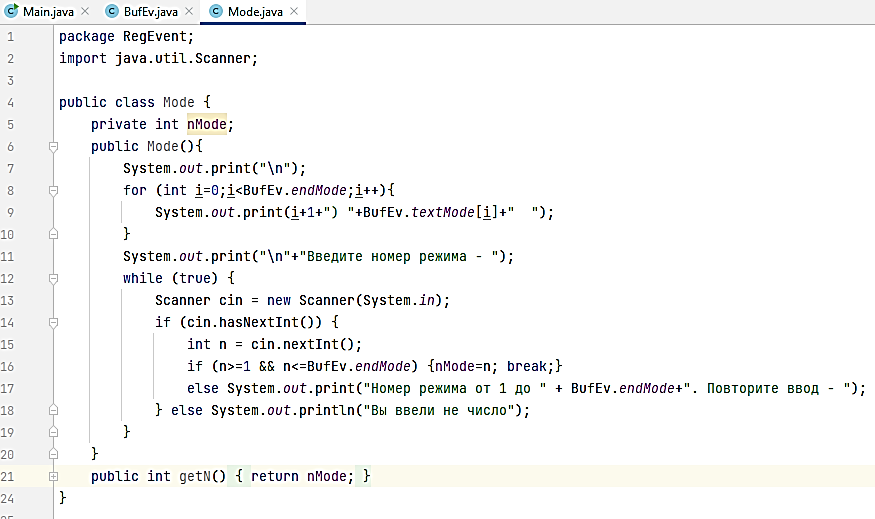
**Ответ**: реальная программа должна должна строиться так, что бы было удобно вводить изменения и желательно в одном месте. В нашем случае такой массив организовали для того, что бы вставить новый режим было достаточно просто.

В это класс будем вставлять описание и других данных, по мере необходимости.

Необходимо обратить внимание на отсутствие конструктора. Это допустимо в данном случае.

Класс Mode

Опишем новый класс **Mode:**



**Анализ кода**

**private int nMode;**

Внутреняя переменная с введенным номером режима. Этот номер можно прочитать в другом объекте через метод **getN**

**System.*out*.print("\n");**

Переход на новую строку.

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

Цикл от 0 до BufEv.*endMode* , где *endMode* находится в классе BufEv

**System.*out*.print(i+1+") "+BufEv.*textMode*[i]+" ");**

Выдача в цикле всех режимов в виде:



System.*out*.print("\n"+"Введите номер режима - ");

Выдача на экран приглашения ввода на следующей строке экрана и курсор устанавливается в конце приглашения.

**while (true) {**

Начало бесконечного цикла (true), пока не встретится команда выхода из цикла ***break***

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

Ввод с клавиатуры в поток **cin**

**if (cin.hasNextInt()) {**

Проверка на целое число данных потока **cin**

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

Чтение числа из потока **cin** в переменную n

**if (n>=1 && n<=BufEv.*endMode*) {nMode=n; break;}**

Проверка переменной n на допустимость режима (1, 2, 3, 4, 5). Если проверка прошла, то в локальную переменную **nMode** записываем номер выбранного режима.

**else System.*out*.print("Номер режима от 1 до " + BufEv.*endMode*+". Повторите ввод - ");**

Если проверка на допустимость номера режима не прошла, то выдается сообщение с допустимыми режимами и цикл ввода повтояется.

**} else System.*out*.println("Вы ввели не число");**

Если введена буква или служебный символ, то цикл повторяется

**public int getN() {return nMode;}**

Метод для чтения извне выбранного режима.

Изменения в классе Main

Далее требуется ввести изменения в классе Main:

заменим фрагмент:



на

**Mode mode=new Mode(); *//Вод номера режима*int n=mode.getN(); *//чтение номера режима***

**Анализ кода**

**Mode mode=new Mode();**

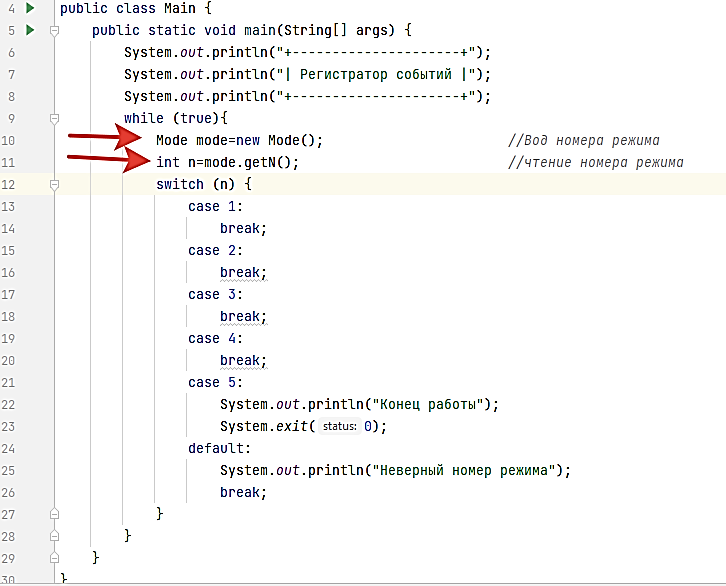
Создание нового объекта **mode** класса **Mode**

При этом выполняется код конструктора класса **Mode**

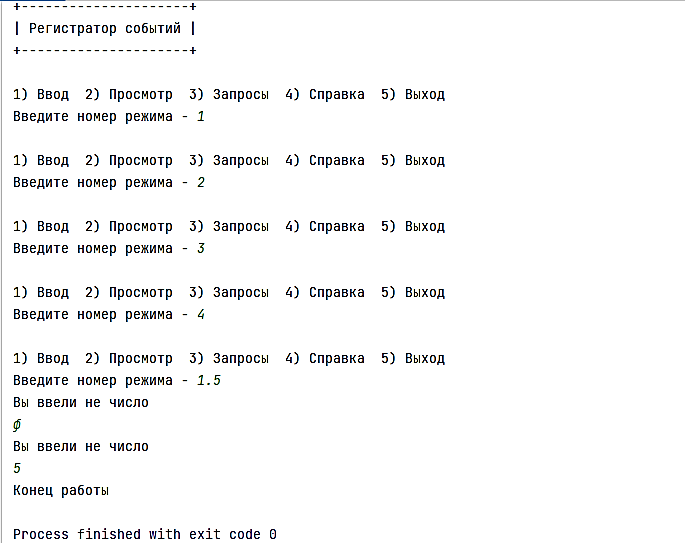
**int n=mode.getN();**

Обращение к методу **getN** из класса **mode** для получения номера режима и занесения номера в переменную **n** целого типа.

Полный текст класса Main:



Результат работы программы:



Конец лекции