

Основы программирования

Лекция 1 (Введение)

Дорогу осилит идущий...

Разработал: Цымбалюк А.Н.

Некоторые организационные вопросы

<http://prog.kiev.ua/forum> - форум где оказывается поддержка слушателем курса

Скачать эту презентацию и остальные учебные материалы можно в разделе «Учебные материалы» форума (внимание доступно только зарегистрированным участникам). Пароль на презентацию java-1-0626

Краткие сведения о языке Java

Java — объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретенной компанией Oracle).

Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине вне зависимости от компьютерной архитектуры.

Дата официального выпуска — 23 мая 1995 года.

Достоинство языка Java

Программы на Java транслируются в байт-код JVM

- Независимость байт-кода от операционной системы и оборудования (Windows, Mac OS X, Linux etc.)
- Автоматическое управление памятью (GC)
- Гибкая система безопасности
- Скорость

Платформы Java

Java SE (J2SE) - Java для настольных компьютеров

Java EE (J2EE) — Java для работы на серверах

Java ME (J2ME) — Java для работы на микроконтроллерах

JavaFX — предназначена для создания графических интерфейсов корпоративных приложений и бизнеса.

Java Card - технология предоставляет безопасную среду для приложений, работающих на смарт-картах и других устройствах с очень ограниченным объёмом памяти и возможностями обработки.

Android

Google App Engine (GAE/Java)

Почему Java хороший выбор?

Jul 2015	Jul 2014	Change	Programming Language	Ratings	Change
1	2	⬆	Java	17.728%	+2.04%
2	1	⬇	C	16.147%	-1.00%
3	4	⬆	C++	8.641%	+3.12%
4	6	⬆	C#	5.652%	+1.60%
5	8	⬆	Python	4.257%	+1.60%
6	3	⬇	Objective-C	3.344%	-6.95%
7	7		PHP	2.893%	-0.02%
8	12	⬆	Visual Basic .NET	2.423%	+0.93%
9	9		JavaScript	2.194%	+0.39%
10	-	⬆	Visual Basic	1.946%	+1.95%

! Этот язык популярен

Используется в крупных проектах

ebay

MINECRAFT

Linked in

YAHOO!

Приват 24

одноклассники

Средства Java

Дистрибутивы

- JRE — виртуальная машина (необходима для выполнения программ)
- JDK — комплект разработчика

Среды разработки

- Eclipse
- NetBeans
- IntelliJIDEA

! Также программы на Java можно писать в блокноте

Цикл разработки программ на Java

Структура программы на Java

- **class** (минимальная единица)
- **package** — группа файлов
- **jar** — группа пакетов

Иерархия проектов

- **java** файлы
- Структура каталогов

Компиляция программы:

*.java -> компилятор -> *.class -> JAR

Первая программа

```
public class MyClass{  
  
    public static void main(String [] arg){  
        System.out.println("Hello World");  
    }  
}
```

Наберите этот текст в блокноте и сохраните файл как MyClass.java

Компиляция Java программ (пример)

Войдите в командную строку, и в папке с файлом выполните команду

```
javac MyClass.java
```

После этого появиться файл `MyClass.class` который можно запустить на выполнение командой

```
java MyClass
```

Пример компиляции и запуска java приложения

```
[alexander@localhost Is_Java]$  
[alexander@localhost Is_Java]$ javac MyClass.java  
[alexander@localhost Is_Java]$ java MyClass  
Hello World  
[alexander@localhost Is_Java]$
```

IDE упрощает труд программиста



Eclipse



NetBeans



IntelliJIDEA

IntelliJIDEA

! В программе курса будет изучаться IDE Eclipse

Eclipse

Бесплатен. Скачать можно тут <https://www.eclipse.org/downloads/>

Внимание скачивать Eclipse нужно той же разрядности
(32 и 64) что и JDK

При первом старте Eclipse выведет сообщение о расположении WorkSpace (рабочего пространства) выберите папку где будут храниться ваши проекты и нажмите ОК.

Создание проекта в eclipse

Для создания нового проекта выберите
File -> New-> Java Project

New Java Project

Create a Java Project

Enter a project name.

Project name:

☒ Use default location

Location:

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE (currently 'java-1.7.0-openjdk-1.7.0.71-2.5.3.0.fc20.i386') [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

Введите имя проекта
и нажмите Finish

Выполнив щелчок мышью на проекте и вызвав контекстное меню добавляем новый проект (New->Package)

New Java Package

Java Package

Create a new Java package.

Creates folders corresponding to packages.

Source folder: Lesson/src Browse...

Name: lesson1

☐ Create package-info.java

? Cancel Finish

Дадим имя новому проекту и нажимаем Finish

Далее добавляем новый класс в проект для этого вызовем контекстное меню проекта и выберем New-> Java Class

The screenshot shows the 'New Java Class' dialog box. The 'Name' field is set to 'Lesson1'. The 'Modifiers' section has 'public' selected. The 'Superclass' is set to 'java.lang.Object'. The 'Which method stubs would you like to create?' section has 'public static void main(String[] args)' selected. The 'Finish' button is highlighted with a red arrow.

Даем имя классу

Устанавливаем галочку для создания главного метода

Нажимает Finish

Появилась окно с текстом проекта. Весь код программы должен принадлежать главному методу класса.

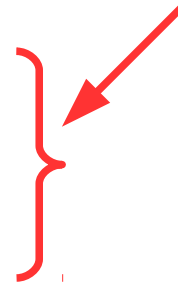
```
package lesson1;

public class Lesson1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("HELLO WORLD");
    }

}
```

Код программы должен быть
внутри тела главного метода



Запустить программу на выполнение можно или выбрав Run -> Run
либо нажав на пиктограмму запуска

Основы компьютерной грамотности

Размерность данных:

- 1 бит : 0 или 1
- 1 байт = 8 бит (10101110)
- 1 килобайт = 1024 байт
- 1 мегабайт = 1024 килобайт
- 1 гигабайт = 1024 мегабайт
- 1 терабайт = 1024 гигабайт

Системы счисления

- Десятичная
- Двоичная - об
- Шестнадцатеричная -ох
-

Алгоритм преобразования числа из любой системы в десятичную

Основание системы (первое число состоящее из 2 цифр)

Например:

- 10-для десятичной
- 10-для двоичной
- 10- шестнадцатеричная

Значение в десятичной системе получается умножением числа на основание системы счисления в степени равной порядку следования разряда (начиная с нуля)

Hex = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Пример преобразования:

$$0x5A3 = 3 \cdot 16^0 + 10 \cdot 16^1 + 5 \cdot 16^2 = 1443$$

Преобразование из десятичной системы счисления в произвольную

Вычисляется частное числа и основания системы счисления, остаток от деления записывается в младший разряд, как только результатом деления станет 0, то алгоритм заканчивается.

$$A = 43 / 2 = 21, B = 43 \% 2 = 1$$

$$A = 21 / 2 = 10, B = 21 \% 2 = 1$$

$$A = 10 / 2 = 5, B = 10 \% 2 = 0$$

$$A = 5 / 2 = 2, B = 5 \% 2 = 1$$

$$A = 2 / 2 = 1, B = 2 \% 2 = 0$$

$$A = 1 / 2 = 0 \rightarrow \text{STOP!!} \rightarrow \text{ob101011}$$

Рекомендуемая литература

1. Герберт Шилд — Java Полное руководство 8-е издание
2. Брюс Эккель — Философия Java
3. Кэти Сьерра и Берт Бейтс - Изучаем Java

Полезные ссылки

1. <http://docs.oracle.com/javase/7/docs/>
2. <http://docs.oracle.com/javase/7/docs/api/>
3. <http://prog.kiev.ua/forum>

Домашнее задание

1) Разобраться с 16-й системой исчисления.

Основы программирования

Лекция 2 (Переменные)

Переменная – именованная область памяти которую можно использовать для осуществления доступа к данным.

Данные, находящиеся в переменной, называются **значением** этой переменной.

Разработал: Цымбалюк А.Н.

Объявление переменных в Java

Тип Имя;

Тип (встроенные типы в Java — примитивные):

- **Числовые**
 - byte
 - short
 - int
 - long
 - float
 - double
- **Символьные**
 - char
- **Логические**
 - boolean

Имя — имя которое вы даете переменной !

Числовой тип переменных

Тип переменной	Размер (бит)	Диапазон
byte	8	-128..127
short	16	-32768..32767
int	32	-2 147 483 648.. 2 147 483 648
long	64	-9 223 372 036 854 775 808.. ..9 223 372 036 854 775 807
float	32	$-3.4 \cdot 10^{38} \dots 3.4 \cdot 10^{38}$
double	64	$-1.8 \cdot 10^{308} \dots 1.8 \cdot 10^{308}$

Символьный и логический тип переменных

Тип переменной	Размер (бит)	Диапазон
char	16	0..65535
boolean		true/ false;

Присваивание значений переменных

Имя=**Значение**;

Значение - значение подходящее соответствующему типу переменной.

Инициализация -
присваивание объявленной переменной значения.

! Можно объявлять переменную одновременно с инициализацией.

Тип Имя=**Значение**;

Пример:

```
package repetishion2;
```

```
public class Prim1 {
```

```
public static void main(String[] args) {
```

```
int x;  
x=5;
```

Объявление переменной x
Инициализация переменной x

```
x=15;
```

Изменение значения переменной x

```
double y=3.5;
```

Объявление с присваиванием

```
}  
}
```

Переменные числового типа:

- Целочисленные (выражаются целыми числами) :
 - byte
 - short
 - int
 - long
- Вещественные (выражаются вещественными числами):
 - float
 - double

! Существуют префиксы для указания системы счисления:
(указывается перед значением)

- 0x - шестнадцатеричная система
- 0 - восьмеричная система
- 0b — двоичная система

Преобразование типов переменных


! Выражение обычно расширяется в сторону более широкой переменной.


- Если один операнд имеет тип `double`, другой тоже преобразуется к типу `double`.
- Иначе, если один операнд имеет тип `float`, другой тоже преобразуется к типу `float`.
- Иначе, если один операнд имеет тип `long`, другой тоже преобразуется к типу `long`.
- Иначе оба операнда преобразуются к типу `int`.

Переполнение целочисленных переменных

! Как только значение переменной превысит максимальный размер ее диапазона, значение станет равно минимальному значению диапазона + значение этой переменной .

Пример: преобразование типов

```
package repetishion2;  
  
public class Prim1 {  
  
    public static void main(String[] args) {  
  
        int x=1;  
        double y=2.3;  
  
        double z=x+y-5.2;  Преобразование типов  
  
        System.out.println("z= "+z);  
    }  
}
```



Вывод значения переменной в консоль

Результат работы программы

z= -1.900000000000000004

Пример: переполнение

```
package repetishion2;  
  
public class Prim1 {  
  
    public static void main(String[] args) {  
  
        byte x=127;  
        System.out.println("x= "+x);  
        x=(byte)(x+1);  
        System.out.println("x="+x);  
    }  
}
```



Переполнение

! Указание на принудительное преобразование
(byte)
(short)
(int)
(long)

Результат работы программы:

x= 127

x=-128

Строковый тип данных

Тип — String;

Значение — "Текст заключенный в кавычки "

Пример:

```
String s1="HELLO";
```

Операторы

Запись	Описание
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Деление по модулю
++	Инкремент
--	Декремент

! Сокращенная запись некоторых операторов

Запись	Альтернатива
a+=1;	a=a+1;
a-=1;	a=a-1;
a*=2;	a=a*2;
a/=2;	a=a/2;
a%=2	a=a%2;

Операторы

Для численных данных действие операторов аналогично их математическому представлению.

! Внимание для целочисленных переменных / - целочисленное деление - Например $5/3=1$

Для строковых переменных используется только оператор «+»

В таком случае результатом будет строка равная первой строке к которой дописана вторая строка.
Например : "abc"+"def"="abcdef";

НУЖНО БОЛЬШЕ МАТЕМАТИКИ !!!

Дополнительные математические операции

Синтаксис	Математическое значение
<code>Math.sqrt(x)</code>	\sqrt{x}
<code>Math.abs(x)</code>	$ x $
<code>Math.pow(x,y)</code>	x^y
<code>Math.exp(x)</code>	e^x
<code>Math.log(x)</code>	$\ln(x)$
<code>Math.cos(x)</code>	$\cos(x)$
<code>Math.round(x)</code>	Округление

И еще много полезных вещей в пакете Math

Ввод переменных с клавиатуры

```
package repetishion2;
```

```
import java.util.Scanner;
```

Обязательные строки
(пока нужно запомнить)

```
public class Prim1 {
```

```
public static void main(String[] args) {
```

```
Scanner sc=new Scanner(System.in);
```

```
int x;
```

```
System.out.println("Input x");
```

```
x=sc.nextInt();
```

Считывание переменной
С клавиатуры

```
System.out.println(x);
```

```
}
```

```
}
```

Что и как считывать

Тип переменной	Как считывать
byte	sc.nextByte()
short	sc.nextShort()
int	sc.nextInt();
long	sc.nextLong()
float	sc.nextFloat()
double	sc.nextDouble()
String	sc.nextLine()


Преобразование строки в число и обратно

Классы:

Integer, Long, Short, Boolean, Char, Byte.

Преобразование числа в строку:

```
String t = Integer.toString(123);  
String t = Long.toString(123);  
String t = Integer.toString(123, 16);
```



Основание системы
счисления

Преобразование строки в число:

```
int n = Integer.parseInt("123");  
long n = Long.parseLong("123");  
double n = Double.valueOf("123");
```


Домашнее задание

1) Написать программу которая считывает 5-и значное число с клавиатуры и выводит цифры из которого оно состоит.

Например : Считывается число 54698

Выводиться:

5

4

6

9

8

2) Написать программу которая вычислит и выведет на экран площадь треугольника если известны его стороны.

3) Написать программу которая вычислит и выведет на экран длину окружности, если ее радиус считывается с клавиатуры.

Основы программирования

Лекция 3 (Условные операторы и циклы)

Управляющие операторы используются для того, чтобы разветвлять ход выполнения программы, основываясь на изменениях состояния программы.

Управляющие операторы делятся на категории: **ветвление** и **повторение(итерация)**.

Разработал: Цымбалюк А.Н.

Оператор if

Общий вид :

```
if (Логическое условие) {  
    Блок операторов которые выполняются в случае  
    когда логическое условие истинно  
}  
else {  
    Блок операторов которые выполняются в случае  
    когда логическое условие ложно  
}
```

! Блок else - необязательный

Логическое условие — один или более операторов сравнения (для переменных числового типа или символьного) или использования переменной типа `boolean`.

Запись оператора	Описание
<code>==</code>	Равно
<code>!=</code>	Не равно
<code><</code>	Меньше
<code>></code>	Больше
<code><=</code>	Меньше или равно
<code>>=</code>	Больше или равно

Диаграммы условных операторов

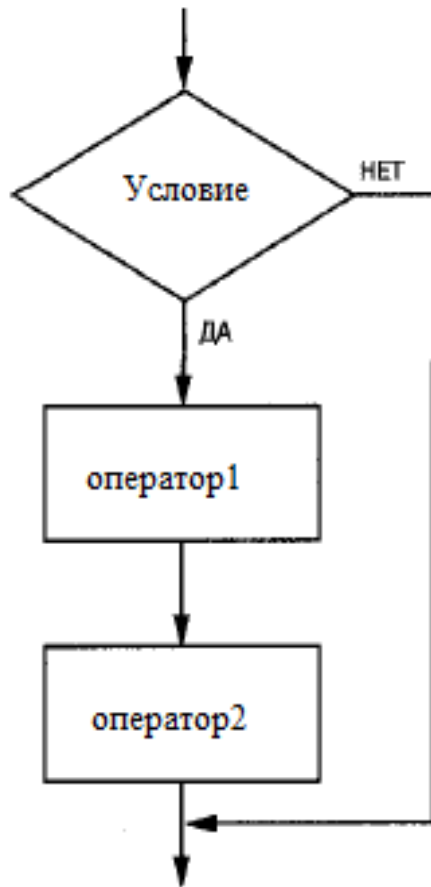


Диаграмма оператора if

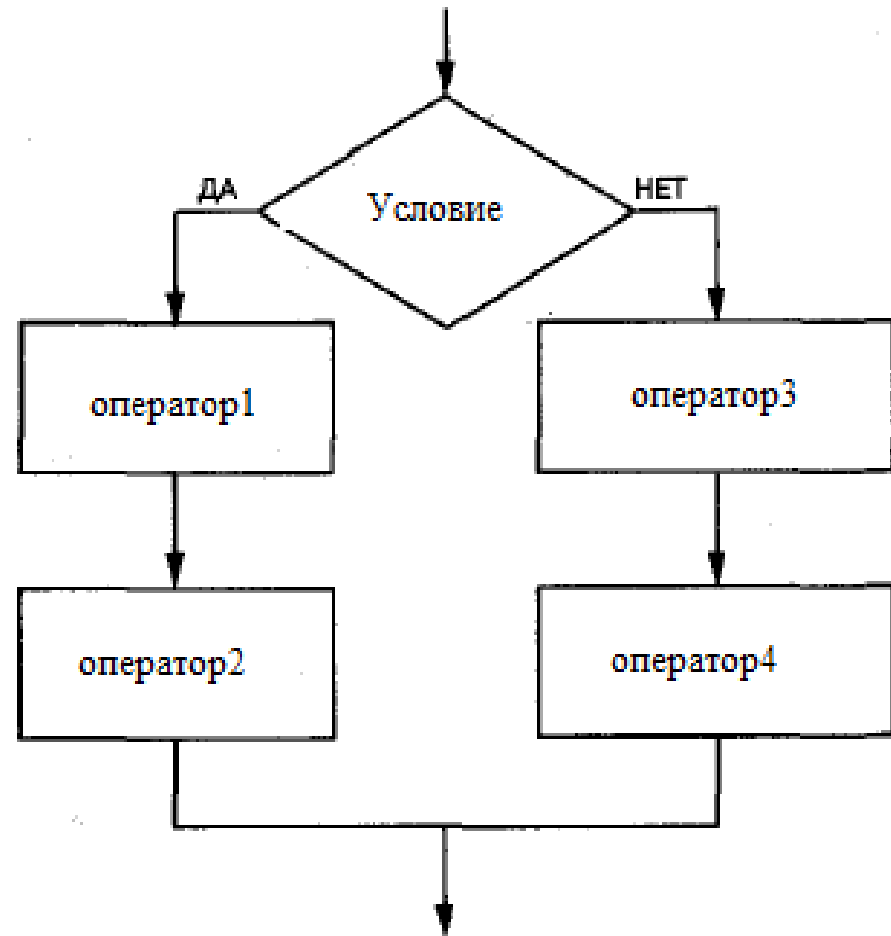


Диаграмма оператора if - else

Вложенный операторы if

Вложенные if

Вложенный *if* – это оператор *if*, который размещен внутри другого *if* – или *else*-оператора. Вложение *if* – обычный прием в программировании. Когда вы вкладываете *if*, нужно помнить, что оператор *else* всегда относится к самому близкому оператору *if*, который находится в том же блоке, что и *else*.

```
if (ЛУ){  
  Блок операторов  
}  
else {  
  if(ЛУ 1){  
    Блок операторов  
  }  
  else {  
    Блок операторов  
  }  
}
```

Вложенный оператор if

Пара if - else

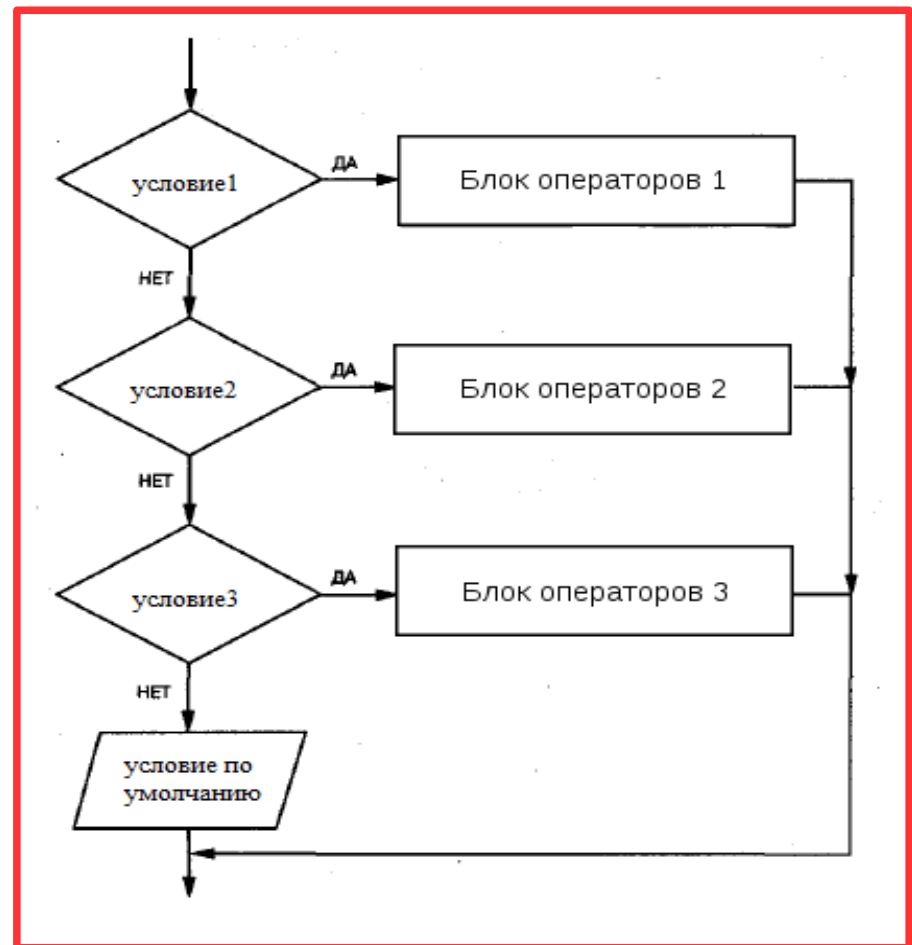
Многозвенный if-else-if

Общую программную конструкцию, которая основана на последовательности вложенных if, называют многозвенным (ladder) if-else-if.

Структура многозвенного if

```
if(ЛУ 1) {  
    блок операторов 1  
}  
else if(ЛУ 2) {  
    блок операторов 2  
}  
else if(ЛУ 3) {  
    блок операторов 3  
}  
....  
else {  
    блок операторов N  
}
```

Диаграмма (блок схема) многозвенного if



Примеры на использование оператора if

```
package lesson3;
import java.util.Scanner;
public class Lesson1 {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int a;
        int b;
        System.out.println("Input a");
        a=sc.nextInt();
        System.out.println("Input b");
        b=sc.nextInt();

        if (a<b){
            System.out.println("a<b");
        }
        else {
            if (a>b){
                System.out.println("a>b");
            }
            else {
                System.out.println("a=b");
            }
        }
    }
}
```

Логическое условие 1-го оператора if

Пара if- else содержит вложенный if-else

Логическое условие вложенного оператора if

Пара if -else является вложенной для внешнего else

Логические (булевы) операторы

&& - «AND» - «И»

(ЛУ 1)&&(ЛУ 2) = Истинно, если истинны ЛУ 1 и ЛУ 2

|| - «OR» - «ИЛИ»

(ЛУ 1)|| (ЛУ 2) = Истинно, если истинно хотя бы одно ЛУ.

! - «NOT» - «НЕ»

Инвертирует логическое условие

Тернарный оператор ?:

Логическое Условие ? выражение1 : выражение2

Если Логическое Условие равно **true**, то вычисляется **выражение1** и его результат становится результатом выполнения всего оператора. Если же Логическое Условие равно **false**, то вычисляется **выражение2**, и его значение становится результатом работы оператора. Оба операнда выражение1 и выражение2 должны возвращать значение одинакового (или совместимого) типа.

```
import java.util.Scanner;

public class Lesson1 {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int a;
        int b;
        int c;
        System.out.println("Input a");
        a=sc.nextInt();
        System.out.println("Input b");
        b=sc.nextInt();
        c = (a<b)?a:b;
        System.out.println(c);
    }
}
```

Тернарный оператор: если $a < b$ то $c = a$, в противном случае $c = b$

Оператор switch

Оператор *switch* —это *java*-оператор множественного ветвления.

Общая форма оператора *switch*:

```
switch (expression) {  
    case value1:  
        //последовательность операторов1  
        break;  
    case value2:  
        //последовательность операторов2  
        break;  
    ...  
    case valueN:  
        //последовательность операторов N  
        break;  
    default:  
        //последовательность операторов по умолчанию  
}  
}
```

! expression должно иметь тип `byte`, `short`, `int` или `char`; каждое `value`, указанное в операторах `case`, должно иметь тип, совместимый с типом выражения. Каждое значение `case` должно быть уникальной константой. Дублирование значений `case` недопустимо.

Работа оператора `switch`

Оператор `switch` работает следующим образом. Значение выражения сравнивается с каждым из указанных значений в `case`-операторах. Если соответствие найдено, то выполняется кодовая последовательность, следующая после этого оператора `case`. Если ни одна из `case`-констант не соответствует значению выражения, то выполняется оператор `default`. Однако оператор `default` необязателен. Если согласующихся `case` нет, и `default` не присутствует, то никаких дальнейших действий не выполняется.

Оператор `break` используется внутри `switch`, чтобы закончить последовательность операторов. Когда встречается оператор `break`, выполнение передается к первой строке кода, которая следует за полным оператором `switch`. Он создает эффект досрочного выхода из `switch`.

Пример использования switch

```
package lesson3;

import java.util.Scanner;

public class Lesson1 {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int month=sc.nextInt();
        String season;
        switch(month){
            case 12:
            case 1:
            case 2:
                season="Зима";
                break;
            case 3:
            case 4:
            case 5:
                season="Весна";
                break;
            case 6:
            case 7:
            case 8:
                season="Лето";
                break;
            case 9:
            case 10:
            case 11:
                season="Осень";
                break;
            default:
                season="непонятный месяц";
        }
        System.out.println(month+" месяц это "+season+".");
    }
}
```

Выражение которое
ищется в операторах case

Блок оператора switch

Действие по умолчанию

Операторы цикла.

Цикл — разновидность управляющей конструкции в высокоуровневых языках программирования, предназначенная для организации многократного исполнения набора инструкций. Также циклом может называться любая многократно исполняемая последовательность инструкций, организованная любым способом (например, с помощью условного перехода).

Последовательность инструкций, предназначенная для многократного исполнения, называется **телом цикла**. Единичное выполнение тела цикла называется **итерацией**. **Выражение** определяющее, будет в очередной раз выполняться итерация, или цикл завершится, называется условием выхода или условием окончания цикла (либо условием продолжения в зависимости от того, как интерпретируется его истинность — как признак необходимости завершения или продолжения цикла). Переменная, хранящая текущий номер итерации, называется **счётчиком итераций цикла** или просто **счётчиком** цикла. Цикл не обязательно содержит счётчик, счётчик не обязан быть один — условие выхода из цикла может зависеть от нескольких изменяемых в цикле переменных, а может определяться внешними условиями (например, наступлением определённого времени), в последнем случае счётчик может вообще не понадобиться.

Виды циклов

Цикл с предусловием — цикл, который выполняется пока истинно некоторое условие, указанное перед его началом. Это условие проверяется до выполнения тела цикла, поэтому тело может быть не выполнено ни разу (если условие с самого начала ложно).

Цикл с постусловием — цикл, в котором условие проверяется после выполнения тела цикла. Отсюда следует, что тело всегда выполняется хотя бы один раз.

Цикл со счётчиком — цикл, в котором некоторая переменная изменяет своё значение от заданного начального значения до конечного значения с некоторым шагом, и для каждого значения этой переменной тело цикла выполняется один раз.

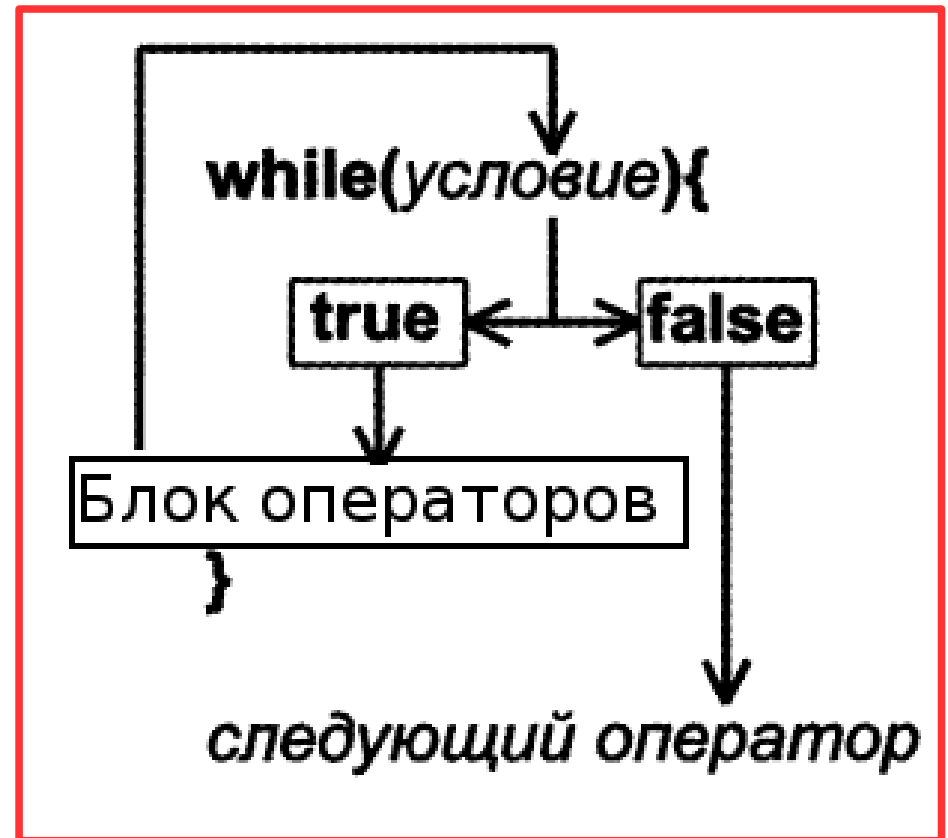
Цикл с предусловием - while

сначала проверяется условие потом выполняется блок операторов

Структура записи в Java

```
while (Логическое условие){  
    Блок операторов  
}
```

Алгоритм выполнения цикла while



Пример цикла while

```
package lesson3;

import java.util.Scanner;

public class Lesson1 {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();

        while(n>0){
            System.out.println(n);
            n=n-1;
        }

    }
}
```

Логическое условие
Продолжения цикла

Тело цикла

Цикл while

! Переменная связанная с Логическим Условием обязательно должна изменяться в теле цикла. В противном случае выполнение тела цикла будет бесконечным.

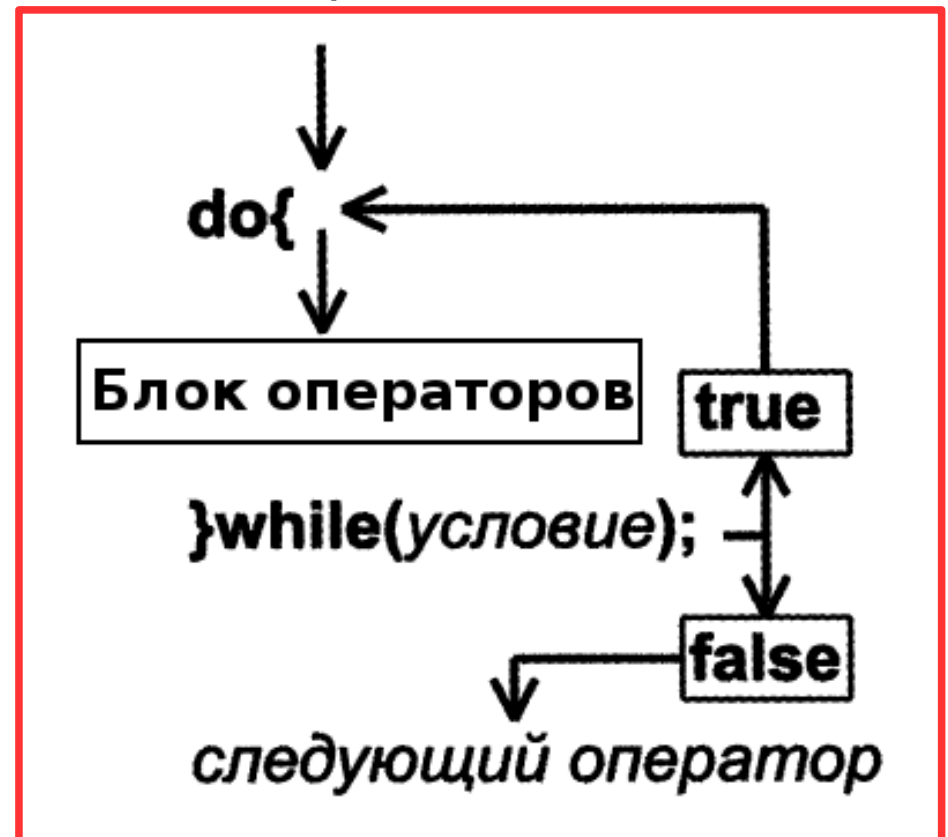
Цикл с постусловием **do — while**

сначала выполняется блок операторов потом проверяется условие

Структура записи в Java

```
do{  
  Блок операторов  
}while (Логическое условие)
```

Алгоритм выполнения
цикла do-while



Пример цикла do - while

```
package lesson3;  
  
import java.util.Scanner;  
  
public class Lesson1 {  
  
    public static void main(String[] args) {  
        Scanner sc=new Scanner(System.in);  
        int n=sc.nextInt();
```

```
        do{  
            System.out.println(n);  
            n=n-1;  
        }while(n>0);  
    }  
}
```

Тело цикла

Цикл do- while

Логическое условие
Продолжения цикла

! Переменная связанная с Логическим Условием обязательно должна изменяться в теле цикла. В противном случае выполнение тела цикла будет бесконечным.

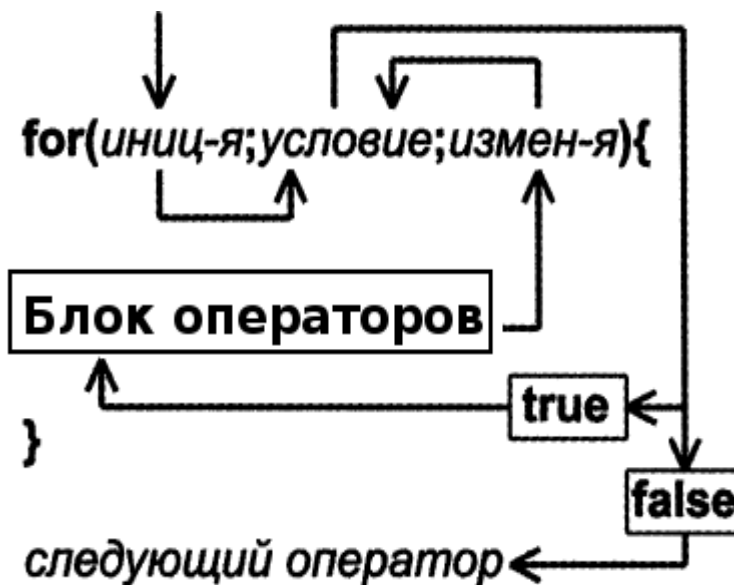
Цикл со счётчиком for

количество итераций заведомо известно

Структура записи в Java

```
for(инициализация; Логическое Условие; приращение){  
    блок операторов  
}
```

Алгоритм выполнения
цикла for



Пример цикла for

```
package lesson3;  
  
import java.util.Scanner;  
  
public class Lesson1 {  
  
    public static void main(String[] args) {  
        Scanner sc=new Scanner(System.in);  
        int n=sc.nextInt();  
  
        for(int i=0;i<n;i++){  
            System.out.println(i);  
        }  
    }  
}
```

Инициализация

Логическое условие
окончания цикла

Приращение переменной
цикла

Тело цикла

Цикл for

Цикл for — более подробный взгляд.

Не обязательно объявлять
переменную цикла

Приращения может и не быть.
Также может выполняться 2 и более
действия

for(; ;)

Условие не обязательно связано
с переменной цикла. Или вообще
отсутствует.

! Строго говоря цикл for — может легко заменить
цикл while и do-while

Еще один пример использования цикла for

```
package lesson3;

import java.util.Scanner;

public class Lesson1 {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int i=0;
        for( ;i<n; ){
            System.out.println(i);
            i++;
        }
    }
}
```

Вложенные циклы

```
for(инициализация; Логическое Условие; приращение){  
операторы
```

```
....
```

```
    for(инициализация; Логическое Условие; приращение){  
        блок операторов  
    }
```

```
....
```

```
}
```



Внутренний цикл



Внешний цикл

Пример вложенных циклов for (рисуем прямоугольник)

```
package lesson3;

import java.util.Scanner;

public class Lesson1 {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

← Внутренний цикл

↑ Внешний цикл

Операторы прерывания цикла

continue – прекращает итерацию цикла

break – завершает цикл

! Внимание оператор **break** прерывает только цикл из которого он был вызван.

Пример использования операторов continue и break

```
package lesson3;  
public class Lesson1 {  
  
    public static void main(String[] args) {  
        int n=10;  
        for(int i=0;i<n;i++){  
  
            if (i==3) continue;  
  
            if(i==7) break;  
            System.out.println(i);  
        }  
    }  
}
```

3-я итерация пропускается

На 7-й цикл прекратиться

0
1
2
4
5
6

Результат работы программы

Область действия переменных объявленных в цикле

```
for(инициализация; Логическое Условие; приращение){  
  int i=0;  
}
```

! Переменная объявленная внутри цикла видна только в теле цикла

Домашнее задание — Уровень 1

- 1) Написать программу которая считает 4 числа с клавиатуры и выведет на экран самое большое из них.
- 2) Есть девятиэтажный дом, в котором 4 подъезда. Номер подъезда начинается с единицы. На одном этаже 4 квартиры. Напишите программу которая получит номер квартиры с клавиатуры, и выведет на экран на каком этаже, какого подъезда расположена эта квартира. Если такой квартиры нет в этом доме то нужно сообщить об этом пользователю.
- 3) С помощью циклов нарисовать «обои». Причем количество полос должно вводиться с клавиатуры. В примере 7 полос.

++++++***+++***

++++++***+++***

++++++***+++***

++++++***+++***

++++++***+++***

- 4) Вычислить с помощью цикла факториал числа - n введенного с клавиатуры ($4 < n < 16$). Факториал числа это произведение всех чисел от этого числа до 1. Например $5! = 5 * 4 * 3 * 2 * 1 = 120$

Домашнее задание - Уровень 2

- 1) Есть круг с центром в начале координат и радиусом 4. Пользователь вводит с клавиатуры координаты точки x и y . Написать программу которая определит лежит ли эта точка внутри круга или нет.
- 2) Дан треугольник координаты вершин $A(0,0)$, $B(4,4)$, $C(6,1)$. Пользователь вводит с клавиатуры координаты точки x и y . Написать программу которая определит лежит ли эта точка внутри треугольника или нет.
- 3) С помощью цикла (только одного) нарисовать такую фигуру. Причем максимальная высота этой фигуры вводится с клавиатуры (в примере максимальная высота - 4)

*

**

**

*

- 4) С помощью циклов вывести на экран все простые числа от 1 до 100. Простое число — число которое делиться нацело только на единицу или само на себя. Первые простые числа это — 2,3,5,7...

Основы программирования

Лекция 4 (Строки и массивы)

Массив — это конечная последовательность упорядоченных элементов одного типа, доступ к каждому элементу в которой осуществляется по его индексу.

Размер или длина массива — это общее количество элементов в массиве. Размер массива задается при создании массива и не может быть изменен в дальнейшем, т. е. нельзя убрать элементы из массива или добавить их туда, но можно в существующие элементы присвоить новые значения.

Разработал: Цымбалюк А.Н.

Массивы в Java

В Java **массивы** являются **объектами**. Это значит, что имя, которое дается каждому массиву, лишь указывает на адрес какого-то фрагмента данных в памяти. Кроме адреса в этой переменной ничего не хранится. Индекс массива, фактически, указывает на то, насколько надо отступить от начального элемента массива в памяти, чтоб добраться до нужного элемента.

Индекс начального элемента — 0, следующего за ним — 1 и т. д. Индекс последнего элемента в массиве — на единицу меньше, чем размер массива.

Минимально возможное количество индексов необходимое для точного указания на элемент массива называется размерностью.

Объявление и инициализация массивов

Объявление

Тип данных которые хранятся в массиве **имя** []

Тип данных которые хранятся в массиве [] **имя**

Инициализация

имя=new Тип данных которые хранятся в массиве[размер]

имя={значение 1, значение 2,...}

! Объявление можно объединить с инициализацией

О внутреннем устройстве массивов

Массив состоит из двух частей — ссылки и данных хранящихся в ОЗУ

`int [] array` ← ссылка на целочисленный массив

`{1,2,3,4,5}` ← Данные массива



Т.е. ссылка только указывает на тип данных массива и где они лежат, но при этом сама она данными массива не является.

Примеры

Массив из 10 целых чисел .

```
int[] a=new int[10];
```

Объявление и инициализация массива вещественных чисел.

```
double [] b={1,2,3,4,5.5};
```

! Для определения длины массива используется функция length (сначала указывается имя массива потом «.» и length. Например - a.length

! Если при создании массива используется оператор new то массив заполняется значениями по умолчанию. Для числовых типов — 0. Для логического — false . Для символьного - '\u0000'. Для строкового — null.

Как обратиться к отдельному элементу массива

Чтобы обратиться к какому-то из элементов массива для того, чтобы прочитать или изменить его значение, нужно указать имя массива и за ним индекс элемента в квадратных скобках. Элемент массива с конкретным индексом ведет себя также, как переменная.

Например:

```
System.out.println(b[1]);  
System.out.println(b[b.length-1]);
```

Результат работы программы:

2

5.5



Внимание если обратиться к несуществующему элементу массива возникнет ошибка выполнения.

Циклы основной инструмент при работе с массивами

```
package pr3;  
  
public class Pr3 {  
  
    public static void main(String[] args) {  
        int[] mas=new int[10];  
        for(int i=0;i<mas.length;i++)  
        {  
            mas[i]=i;  
            System.out.print(mas[i]+" ");  
        }  
    }  
}
```

Объявление массива 10 целых чисел

Присвоение значения элементу массива

Вывод элемента массива на экран

Копирование массивов

Копировать массивы можно с помощью поэлементного копирования

```
package pr3;

public class Pr3 {

    public static void main(String[] args) {

        int[] mas={3,7,8,1,2,78,33,45,8,10};
        int[] b=new int[10];

        for(int i=0;i<mas.length;i++){
            b[i]=mas[i];
        }
    }
}
```

Либо с помощью встроенной функции arraycopy

Формат вызова

System.arraycopy(какой массив копируется, начальный индекс с которого начинается копирование, в какой массив копируется, начальный индекс вставки, длина скопированного участка массива)

```
int[] mas={3,7,8,1,2,78,33,45,8,10};  
int[] b=new int[10];  
System.arraycopy(mas, 0, b, 0, mas.length);
```

Длина скопированного
Участка равна длине mas

Какой массив копируется

С какого индекса начинается вставка

В какой массив копируется

С какого индекса начинается копирование

Дополнительные функции для работы с массивами

Для вызова необходимо импортировать — `import java.util.Arrays`

Копирование:

`Arrays.copyOfRange(какой массив копируется, начальный индекс, конечный индекс)`

Заполнение:

`Arrays.fill(какой массив заполняется, чем заполняется)`

Сортировка:

`Arrays.sort(какой массив сортируют)`

`Arrays.sort(какой массив сортируют, начальный индекс, конечный индекс)`

Преобразование:

`Arrays.toString(какой массив преобразуется)`

Специальный вид цикла для работы с массивами

Цикл - foreach

Объявление:

```
for( переменная: массив)
```

Переменная
принимает
последовательно
значение элементов
массива

Пример

```
for(int k:b){  
System.out.print(k+" ");  
}
```

! Может использоваться только для чтения элементов массива

Двухмерные массивы

Объявление:

Тип [][] имя = new Тип [строки][столбцы]

Примером двухмерного массива является матрица
(или морской бой ☺)

Инициализация

```
int [][] a={{1,2,3},{4,5,6},{7,8,9}}
```

Пример объявления, заполнения и вывода на экран 2-х мерного массива

```
package pr3;
```

```
public class Pr3 {
```

```
public static void main(String[] args) {
```

```
int [][] b=new int[4][5];
```

Объявление

```
for(int i=0;i<b.length;i++){
```

Цикл по строкам

```
for(int j=0;j<b[0].length;j++){
```

Цикл по столбцам

```
b[i][j]=i+j;
```

Заполнение значениями

```
System.out.print(b[i][j]+" ");
```

Вывод на экран

```
}  
System.out.println();  
}  
}  
}
```

Как устроены многомерные массивы в Java

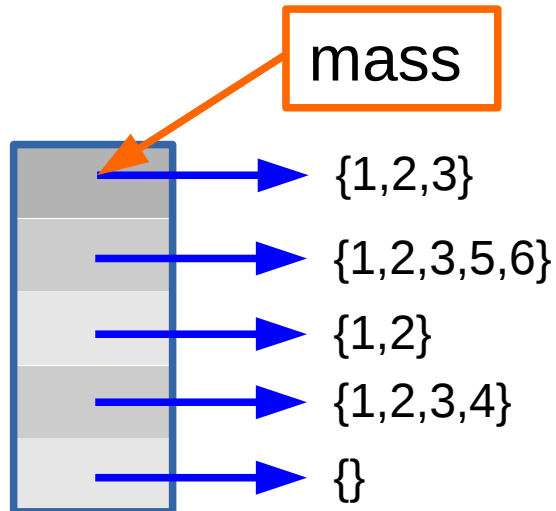
```
int [][] mass
```

Объявление «двухмерного» массива типа int.

Объявление ссылки на одномерный массив

Тип данных хранящихся в массиве — ссылки на одномерные массивы типа int.

Т.е. двухмерный массив это массив — это массив массивов.



Рваный двухмерный массив типа int

Создание «рваных» массивов

При создании 2-х мерных массивов при создании достаточно объявить первый размер, а второй размер объявлять динамически

```
package pr3;

public class Pr3 {

    public static void main(String[] args){
        int [][] b=new int[8][];
        for(int i=0;i<b.length;i++){
            b[i]=new int[i];
            for(int j=0;j<b[i].length;j++){
                b[i][j]=i+j;
                System.out.print(b[i][j]+" ");
            }
            System.out.println();
        }
    }
}
```

Вывод программы

```
1
2 3
3 4 5
4 5 6 7
5 6 7 8 9
6 7 8 9 10 11
7 8 9 10 11 12 13
```

Строки

Объявление строк
`String имя = "Текст";`

Конкатенация строк — объединение строк с помощью операции `+`. Каждая строка дописывается вправо.

Пример

```
String s=" Java"+" the"+" best"
```

! Конкатенация строк возможна с числовыми типами переменных.

Методы для работы со строками

Выделение символов

getChars(int начало источника, int конец, char[] цель, int начало цели)
charAt(индекс символа)
toCharArray()

Сравнение строк

String1.equals(String2)
String1.equalsIgnoreCase(String2) — без учета регистра
String1.compareTo(String2) — возвращает положительное или отрицательное число в зависимости от результата сравнения строк

Поиск

indexOf(символ) — ищет индекс первого вхождения символа
lastIndexOf(подстрока) — ищет индекс последнего вхождения подстроки

Выделение части строк

substring(начальный индекс, конечный индекс)

Примеры работы со строками

```
package com.gmail.tsa;
```

```
import java.util.Arrays;
```

```
public class Main {  
    public static void main(String[] args) {
```

```
        String text = "Hello";  
        String text1 = "World";
```



Создаем 2 строки

```
        char[] t = text.toCharArray();  
        System.out.println(Arrays.toString(t));
```



Получаем массив символов из строки

```
        char l = text.charAt(0);  
        System.out.println(l);
```



Получаем 0-символ из строки

```
        System.out.println(text.equals(text1));
```

```
        int i = text.indexOf("ll");
```



Ищем строку «ll» в строке и получаем индекс

```
        String subtext = text.substring(0, 3);  
        System.out.println(subtext);  
    }
```



Вырезаем кусок строки

```
}
```


Метод split для работы со строками

Используется для разбиения строки на части на основе разделителей. В результате получается массив строк.

```
package com.gmail.tsa;
```

```
import java.util.Arrays;
```

```
public class Main {  
    public static void main(String[] args) {
```

```
        String text = "123,345,657";
```

Исходная строка

```
        String[] t = text.split(",");
```

Разбиваем ее на части на основе
Разделителя - «,»

```
        for (String temp : t) {  
            System.out.println(temp);  
        }  
    }
```

Выводим получившийся массив на экран

Домашнее задание -Уровень 1

- 1) Дан массив {0,5,2,4,7,1,3,19} — написать программу для подсчета нечетных цифр в нем.
- 2) Написать код для возможности создания массива целых чисел (размер вводится с клавиатуры) и возможности заполнения каждого его элемента вручную. Выведите этот массив на экран.
- 3) Создать массив случайных чисел (размером 15 элементов). Создайте второй массив в два раза больше, первые 15 элементов должны быть равны элементам первого массива, а остальные элементы заполнить удвоенных значением начальных. Например
Было → {1,4,7,2}
Стало → {1,4,7,2,2,8,14,4}
- 4) Введите строку текста с клавиатуры — реализуйте программу для возможности подсчета количества символа — 'b' в этой строке, с выводом результат на экран.

Домашнее задание — Уровень 2

- 1) «Перевернуть массив» (При выполнении задания использовать дополнительный массив нельзя) - (3 часа)

Было

```
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
```

Стало

```
[1, 1, 1, 1, 1, 1]
[2, 2, 2, 2, 2, 2]
[3, 3, 3, 3, 3, 3]
[4, 4, 4, 4, 4, 4]
[5, 5, 5, 5, 5, 5]
[6, 6, 6, 6, 6, 6]
```

- 2) Написать код для зеркального переворота массива (7,2,9,4) -> (4,9,2,7). - массив может быть произвольной длины. (При выполнении задания использовать дополнительный массив нельзя)(1 час)

- 3) Ввести с клавиатуры число (до миллиарда) которое обозначает количество долларов и центов пользователя. Вывести это количество прописью. (4 часа)

Например:

How much money do you have?

123,34

You have: one hundred twenty three dollars thirty four cents

Основы программирования

Лекция 5 (Методы)

Метод — это именованный обособленный блок кода выполняющий последовательность каких либо операций.

Разработал: Цымбалюк А.Н.

Применение методов

- Когда нужно разбить задачу на более мелкие подзадачи
- Когда нужно многократно провести один набор операций над сходными данными
- Когда нужно написать и протестировать метод для дальнейшего использования в ваших проектах

Объявление методов в Java

```
Тип_возвращаемого_значения имя (тип параметр 1, тип параметр 2, ...){  
Тело метода  
}
```

Тип_возвращаемого_значения - любой тип переменных в Java, в теле метода обязательно должно быть ключевое слово **return** после которого идти переменная или значение такого же типа.

! Метод может и не возвращать значения тогда его тип возвращаемого значения - **void**

Имя - имя метода придуманное пользователем (внимание слова зарезервированные в Java использовать нельзя). Желательно что бы имя метода начиналось с маленькой буквы.

Параметр — любой тип переменных Java

Тело метода — последовательность операций

Объявление метода

Перед типом переменной нужно указать слово **static** - это только пока не начнете использовать классы.

Методы объявляются вне главного метода , а используются внутри его.

Пример объявления метода

Метод возвращает целочисленную переменную

```
int sum(int a, int b) {  
    return a+b;  
}
```

Имя метода

Два целочисленных значения - параметры

Оператор return
вернет целочисленное значение

Пример объявления и использования метода

```
package pr3;
```

```
public class Pr3 {
```

```
public static void main(String[] args) {
```

```
int d=5;
```

```
int c=7;
```

Использование метода

Фактические параметры

```
System.out.println(sum(d,c));
```

```
}
```

Формальные параметры

```
static int sum(int a,int b){  
return a+b;  
}
```

Объявление и описание
метода

Формальные параметры позиционно приравниваются
фактическим и получают их значение.

Пример объявления и использования метода

```
package pr3;  
  
public class Pr3 {  
  
    public static void main(String[] args) {  
  
        int d=5;  
        int c=7;  
  
        sum(d,c);  
  
    }  
  
    static void sum(int a,int b){  
        System.out.println(a+b);  
    }  
  
}
```

Использование метода

Объявление и описание
метода

Как работают методы

Блок операторов основного метода

.....

Вызов метода

.....

.....

Тело метода

Операторы метода

.....

.....

Возврат значения



Передача параметров в методы

Все примитивные типы передаются по значению — т.е. при передаче создается локальная копия переменной и в методе используется копия. Т.е. при изменении значение переменной в методе в главном методе ее значение не меняется.

Все объектные типы передаются в методы по ссылке — т.е. при передаче передается ссылка на объект, и его изменения в теле метода оказывают влияние на сам объект.

Пример метода с вызовом по значению

```
package pr3;

public class Pr3 {

    public static void main(String[] args) {

        int a=5;
        int b=3;
        System.out.println("Значение переменной в главном методе до вызова  
sum - "+a);
        System.out.println("Результат вызова метода sum - "+sum(a,b));
        System.out.println("Значение переменной в главном методе после  
вызова sum - "+a);

    }

    static int sum(int a,int b){
        a=a+3;
        System.out.println("Значение a в методе sum - "+a);
        return a+b;
    }
}
```

Пример метода с вызовом по ссылке

```
package pr3;

public class Pr3 {

    public static void main(String[] args) {

        int[] a={1,2,3,4,5,67,8};
        System.out.println("Значение переменной в главном методе до вызова sum - ");
        for(int k:a){System.out.print(k+" ");}
        System.out.println();
        sum(a);
        System.out.println();
        System.out.println("Значение переменной в главном методе после вызова sum - ");
        for(int k:a){System.out.print(k+" ");}

    }

    static void sum(int[]a){
        a[3]=a[3]+3;
        System.out.println("Значение a в методе sum - ");
        for(int k:a){System.out.print(k+" ");}

    }
}
```

Пример метода с вызовом по ссылке приводящий к изменению данных- (сортировка выборкой)

```
package pr3;
import java.util.Random;
public class Pr3 {

    public static void main(String[] args)
    {

        Random rn=new Random();
        int[] a=new int[15];
        int[] b=new int[15];
        for(int i=0;i<a.length;i++){
            a[i]=rn.nextInt(20);
        }
        print(a);
        b=sort(a);
        print(b);
        print(a);
    }
}
```

← **Главный метод**

Методы →

```
static void print(int[]a){
    System.out.println();
    for(int k:a){System.out.print(k+" ");}
}

static int[] sort(int[]a){
    for(int i=0;i<a.length;i++){
        for(int j=i;j<a.length;j++){
            int temp;
            if(a[j]<a[i]){
                temp=a[j];
                a[j]=a[i];
                a[i]=temp;
            }
        }
    }
    return a;
}
```

Пример метода с вызовом по ссылке не приводящий к изменению данных- (сортировка пузырьком)

```
package pr3;
import java.util.Random;
public class Pr3 {

    public static void main(String[] args) {

        Random rn=new Random();
        int[] a=new int[15];
        int[] b=new int[15];
        for(int i=0;i<a.length;i++){
            a[i]=rn.nextInt(20);
        }
        print(a);
        b=sort(a);
        print(b);
        print(a);

    }
```

← **Главный метод**

```
static void print(int[]a){
    System.out.println();
    for(int k:a){System.out.print(k+" ");}
}

static int[] sort(int[]a){
    int[]b=new int[a.length];
    System.arraycopy(a, 0, b, 0, a.length);
    for(;;){
        int sum=0;
        for(int i=0;i<b.length-1;i++){
            int temp;
            if(b[i]>b[i+1]){
                temp=b[i];
                b[i]=b[i+1];
                b[i+1]=temp;
                sum++;
            }
        }
        if (sum==0) break;
    }
    return b;
}
```

Методы →

Некоторые замечания о методах

В теле метода могут вызываться другие методы

Пример: вычисление объема цилиндра

```
package pr3;

public class Pr3 {

    public static void main(String[] args) {

        System.out.println(calculateVolume(2.1,3));

    }

    static double calculateArea(double r){
        return Math.PI*r*r;
    }

    static double calculateVolume(double r,double h){
        return h*calculateArea(r);
    }

}
```

Вызов другого метода

Некоторые замечания о методах

Параметром метода может быть результат другого метода

Пример: вычисление объема цилиндра

```
package pr3;

public class Pr3 {

    public static void main(String[] args) {

        System.out.println(calculateVolume(calculateArea(2.1),3));

    }

    static double calculateArea(double r){
        return Math.PI*r*r;
    }

    static double calculateVolume(double S,double h){
        return h*S;
    }

}
```



Результат метода как параметр

Домашнее задание — Уровень 1

- 1) Напишите метод который вернет максимальное число из массива целых чисел.
- 2) Реализуйте метод параметрами которого являются - целое число, вещественное число и строка. Возвращает он конкатенацию строки с суммой вещественного и целого числа.
- 3) Реализуйте метод рисующий на экране прямоугольник из звездочек «*» — его параметрами будут целые числа которые описывают длину и ширину такого прямоугольника.
- 4) Напишите метод который реализует линейный поиск элемента в массиве целых чисел. Если такой элемент в массиве есть то верните его индекс, если нет то метод должен возвращать число - «-1»
- 5) Напишите метод который вернет количество слов в строке текста.

Домашнее задание — Уровень 2

1) Существуют такие последовательности чисел

0,2,4,6,8,10,12

1,4,7,10,13

1,2,4,8,16,32

1,3,9,27

1,4,9,16,25

1,8,27,64,125

Реализуйте программу которая выведет следующий член этой последовательности (либо подобной им). Например пользователь вводит строку 0,2,4,6,8,10,12 ответом программы должно быть число 14. (6 часов)

2) Число-палиндром с обеих сторон (справа налево и слева направо) читается одинаково. Самое большое число-палиндром, полученное умножением двух двузначных чисел – $9009 = 91 \times 99$.

Найдите самый большой палиндром, полученный умножением двух трехзначных чисел. (4 часа)

3) Существует массив $\{1,2,3,4,5\}$ — размер массива может быть произвольным. Напишите программу которая выведет на экран все возможные комбинации из этих цифр. Внимание повторений быть не должно. (2 часа)

Основы программирования

Лекция 6 (Дата и Время)

Единственная причина для существования времени — чтобы все не случилось одновременно.
А.Эйнштейн

Разработал: Цымбалюк А.Н.

Получение даты в Java

Для работы необходимо подключить дополнительный пакет

```
import java.util.Date;
```

Потом нужно создать объект типа Date

```
Date dd=new Date();
```



Имя которое вы дали объекту

! Это создаст объект Date с текущим временем и датой

Другой способ создания объектов типа Date

Date **имя** = new Date(long **t**)

Имя — имя которое вы даете объекту
t- переменная с количеством миллисекунд прошедших с 1 января 1970 года

Пример

```
long t=403231213974L;  
Date dd=new Date(t);  
System.out.println(dd);
```

Форматированный вывод даты и времени

Для работы необходимо подключить дополнительные пакеты

```
import java.text.SimpleDateFormat;
```

Потом нужно создать объект типа SimpleDateFormat на основе специального шаблона

```
SimpleDateFormat stf=new SimpleDateFormat("yy:MM:dd");
```



Имя которое вы дали объекту



Пример шаблона

Создать форматированную строку на основе созданного объекта

```
String res=stf.format(dd);
```

Шаблон даты и времени

Символ	Что означает
G	Эра
y	Год(4 числа)
yy	Год(последние 2 цифры)
yyyy	Год(4 числа)
M	номер месяца без лидирующих нулей
MM	номер месяца (с лидирующими нулями если номер месяца < 10)
MMM	четырёх буквенное сокращение месяца
MMMM	полное название месяца
w	неделя в году без лидирующих нулей
ww	неделя в году с лидирующими нулями
W	неделя в месяце без лидирующих нулей
WW	неделя в месяце с лидирующим нулем
D	день в году
d	день месяца без лидирующих нулей
dd	день месяца с лидирующими нулями

Символ	Что означает
F	день недели в месяце без лидирующих нулей
FF	день недели в месяце с лидирующими нулями
E	день недели (сокращение)
EEEE	день недели (полностью)
a	AM/PM указатель
H	часы в 24-часовом формате без лидирующих нулей
HH	часы в 24-часовом формате с лидирующим нулем
k	количество часов в 24-часовом формате
K	количество часов в 12-часовом формате
h	время в 12-часовом формате без лидирующих нулей
hh	время в 12-часовом формате с лидирующим нулем
m	минуты без лидирующих нулей
mm	минуты с лидирующим нулем
s	секунды без лидирующих нулей
ss	секунды с лидирующим нулем
Z	часовой пояс
Z	часовой пояс в формате RFC 822

Пример

```
package lesson6;  
import java.util.Date;  
import java.text.SimpleDateFormat;  
public class L6 {
```

```
public static void main(String[] args) {  
    long t;  
    t=System.currentTimeMillis();
```

```
Date dd=new Date(t);
```



Создание нового объекта Date

```
SimpleDateFormat stf=new SimpleDateFormat("'Сегодня ' d MMMM yyyy 'год'");
```



Создание объекта SimpleDateFormat



Описание шаблона

```
String res=stf.format(dd);  
System.out.println(res);  
}
```



Создание форматированной строки

```
}
```

Организация удобного ввода даты

Нужно подключить дополнительный пакет

```
import java.text.ParseException;
```

Потом создать шаблон

```
SimpleDateFormat stf=new SimpleDateFormat(" d MMMM yyyy ");
```

Считать строку с клавиатуры и поместить ее в обработчик

```
try{  
Date dd=stf.parse(str);  
System.out.println(dd);  
}  
catch(ParseException e){  
System.out.println("Wrong input");  
}
```

Пример

```
package com.gmail.tsa;
import java.util.Date;
import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.Scanner;
public class Main {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        Date dd=new Date();
        SimpleDateFormat stf=new SimpleDateFormat("d MM yyyy");

        System.out.println("Input Date");
        String str=sc.nextLine();

        try{
            dd=stf.parse(str);
            System.out.println(dd);
        }
        catch(ParseException e){
            System.out.println("Wrong input");
        }
    }
}
```

Методы объекта Date

Метод	Описание
<code>boolean after(Data name)</code>	Вернет - «истина» если вызывающий объект старше name
<code>boolean before(Data name)</code>	Вернет - «истина» если вызывающий объект младше name
<code>int compareTo(Data name)</code>	Сравнивает даты, если они равны то вернется 0. Отрицательное значение если name старше, положительное если наоборот
<code>long getTime()</code>	Вернет кол-во милисекунд с 1 апреля 1970 года
<code>void setTime(long time)</code>	Устанавливает время

Использование класса Calendar

Для работы необходимо импортировать дополнительный пакет

```
import java.util.Calendar;
```

Создать объект класса Calendar

```
Calendar cl=Calendar.getInstance();
```

Для класса существуют следующие методы (список не полный)

Получение данных —

```
get(Calendar.Date )  
get(Calendar.MONTH)  
get(Calendar.YEAR)  
get(Calendar.HOUR)  
get(Calendar.MINUTE)  
get(Calendar.SECOND)
```

Установка данных -

```
set(Calendar.HOUR, значение)  
set(Calendar.MINUTE, значение)  
set(Calendar.SECOND, значение)
```

Использование классов `StringBuilder` и `StringBuffer`

Эти классы используются для комфортной работы со строками

Для работы необходимо объявить объект нужного типа

```
StringBuffer sb=new StringBuffer("HELLO WORD");
```

! Также можно не указывать параметров или вместо них указать размер.

```
StringBuffer sb=new StringBuffer();
```

```
StringBuffer sb=new StringBuffer(10);
```

Методы классов StringBuilder и StringBuffer

Метод	Что означает
setLength(int размер)	Устанавливается длина. Все символы не попавшие в диапазон теряются.
charAt(int где)	Извлекает символ по индексу
setCharAt(int где, char символ)	Установить символ по индексу
getChars(int нач.,int кон., char [] куда, int с какого индекса)	Копирует подстроку символов
append(что добавить)	Добавляет значение переменных к существующему классу
insert(int индекс, что)	Вставляет одну строку в другую начиная с индекса
reverse()	Меняет порядок символов
delete(int нач,int конец)	Удаляет кусок текста
deleteCharAt(int индекс)	Удаляет символ по индексу
replace(int нач., int конец., String строка)	Заменяет один набор символов другим
substring(int нач,int кон)	Вырезает часть строки
int indexOf(String строк ,int нач)	Поиск подстроки (если найдет то первый индекс, если нет то -1)
void trimToSize()	Уменьшает размер для соответствия содержимому

В чем же разница ?

StringBuilder — более быстрый, но не синхронизированный.

StringBuffer — более безопасный, но и более медленный.

```
package com.gmail.tsa;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        StringBuffer sb=new StringBuffer("HELLO WORD");  
        sb.replace(0, 5, "BABAY");  
  
        System.out.println(sb);  
    }  
}
```

Использование класса Formatter

Предназначен для создания форматированных строк

Для работы необходимо подключить дополнительный пакет

```
import java.util.Formatter;
```

Создать объект Formatter

```
Formatter fr=new Formatter();
```

Пример

```
package lesson6;
```

```
import java.util.Formatter;
```

```
public class L6 {
```

```
public static void main(String[] args) {
```

```
    Formatter fr=new Formatter();
```

```
    }
```

```
}
```

Методы класса Formatter

Метод	Описание
<code>void close()</code>	Закрывает объект Formatter
<code>void flush()</code>	Сбрасывает буфер формата
<code>Formatter format (String формстрока, аргументы)</code>	Форматирует аргументы в соответствии с строкой форм. строки
<code>String toString()</code>	Возвращает отформатированную строку

Основы форматирования

Строка форматирования представляет собой строку где нужно вставить отформатированные символы на определенные места указанные с помощью спецификаторов формата.

Спецификатор формата	Тип аргумента
%a, %A	Шеснадцетичное с плавающей точкой
%b, %B	Булево
%c	Символ
%d	Десятичное целое
%e, %E	Научная нотация
%f	Десятичное с плавающей точкой
%o	Восьмеричное целое
%n	Вставляет символ перевода строки
%s, %S	Строка
%t, %T	Время и дата
%x, %X	Шеснадцетичное целое
%%	Вставляет символ %

Пример

```
package com.gmail.tsa;
```

```
import java.util.Formatter;
```

```
public class Main {
```

```
public static void main(String[] args) {
```

```
Formatter fr=new Formatter();
```

```
fr.format("Форматировать в %s очень просто. %d год",  
"Java", 2014);
```

```
String str=fr.toString();
```

```
System.out.println(str);
```

```
}  
}
```

Спецификатор формата



%s очень просто. %d год",



Строка форматирования

Параметры



Использование форматирования

Указание точности(используется со спецификатором %f, %e, %g, %s)

Для числового типа : кол-во десятичных разрядов. кол-во чисел после запятой.

Для символьного типа: мин длинна . max длинна

Спецификаторы **формата даты времени**
(работает только с объектами класса **Calendar**)

Суффикс	Заменяется на
a	Сокращенное название дня недели
A	Полное название дня недели
b	Сокращенное название месяца
B	Полное название месяца
c	День:Месяц:Год
C	Первые два знака года
D	Месяц/день/год
h	Сокращенное название месяца
N	Наносекунды

Побитовые операции (можно применять к целочисленным переменным)

Побитовые операции Java

Операция	Описание
	OR
&	AND
^	XOR
~	NOT

Операторы сдвига Java

Операция	Описание
<<	Сдвиг влево
>>	Сдвиг вправо
>>>	Беззнаковый сдвиг вправо

Операция |

Вернет 1 если хоть один из битов равен 1.

```
0101011
0010000
-----
0111011
```

Операция &

Вернет 1 если оба бита равны 1.

```
0101011
0010000
-----
0000000
```


Операция \wedge

Вернет 1 если хотя бы один из битов равен 1. Однако если оба бита равны 1 то результатом будет 0.

```
1101011
1010000
-----
0111011
```

Операция \sim

Инвертирует биты

```
1101011
-----
0010100
```

Операция сдвига

Значение << количество

Сдвигает биты в значение на указанное количество раз

$000001110 \ll 2 = 000111000$

Значение >> количество

Сдвигает биты в значение на указанное количество раз

$000001110 \gg 2 = 000000011$

Значение >>> количество

Сдвигает биты в значение на указанное количество раз без учета знака

$000001110 \ggg 2 = 000000011$

Пример

```
public class BitLogik {  
  
    public static void main(String[] args) {  
        String  
        binary[]={ "0000", "0001", "0010", "0011", "0100", "0101", "0110", "0111", "1  
000", "1001", "1010", "1011", "1100", "1101", "1110", "1111"};  
        int a=3;  
        int b=6;  
        int c=a | b;  
        int d=a & b;  
        int e=a ^ b;  
        int f=(~a&b)|(a& ~b);  
        System.out.println("a="+binary[a]);  
        System.out.println("b="+binary[b]);  
        System.out.println("a|b="+binary[c]);  
        System.out.println("a&b="+binary[d]);  
        System.out.println("a^b="+binary[e]);  
        System.out.println("(~a&b)|(a& ~b)"+binary[f]);  
  
    }  
  
}
```

Домашнее задание — Уровень 1

- 1) Написать программу которая вернет количество миллисекунд прошедших от такого же числа, но в прошлом месяце до сегодняшней даты. Например если сегодня 3 августа, то узнать сколько миллисекунд прошло с 3 июля.
- 2) Написать свой вариант метода `Arrays.toString()` для `int[]`.
- 3) Ввести с консоли число в бинарном формате. Перевести его в `int` и вывести на экран ("10" -> 2).
- 4) Выведите на экран 10 строк со значением числа Пи. Причем в первой строке должно быть 2 знака после запятой, во второй 3, в третьей 4 и т.д.

Домашнее задание — Уровень 2

- 1) Ввести с консоли дату. Сравнить ее с текущей датой в системе. Вывести отличающиеся части (год, месяц) на экран.
- 2) Расстояние Хэмминга между двумя двоичными числами - это число позиций, в которых биты различаются .

Для примера:

117 = 0 1 1 1 0 1 0 1

17 = 0 0 0 1 0 0 0 1

$H = 0+1+1+0+0+1+0+0 = 3$ - расстояние Хэмминга между (117,17)
Даны два положительных целых числа (N, M) в десятичном виде.
Вам необходимо подсчитать расстояние Хэмминга между этими двумя числами.

- 3) Вовочка сидя на уроке писал подряд одинаковые числа. Когда Мария Ивановна забрала у него тетрадь там было несколько рядов чисел. Напишите программу которая определит минимальное число которое писал Вовочка например:

11111111=>1

12121212=>12

121121121=>121

Основы программирования

Лекция 7 (Работа с файлами)

Лучший метод уменьшения размеров файлов: "Del *.*" - 100% сжатие.

Неизвестный автор.

Разработал: Цымбалюк А.Н.

Базовые сведения о использовании `PrintWriter`

`PrintWriter` - представляет собой более удобный способ вывода информации.

Для создания объекта `PrintWriter` требуется выполнить операторы

Подключение дополнительного пакета

```
import java.io.*;
```

Создать объект этого класса

```
PrintWriter pw=new PrintWriter(System.out,true);
```

Далее можно использовать созданный объект используя его методы

```
pw.println("HELLO");
```

Использование класса `PrintWriter` считается более предпочтительным в консольных приложениях

Пример

```
package fw;  
import java.io.*;  
public class Fw {  
  
    public static void main(String[] args) {  
        PrintWriter pw=new PrintWriter(System.out,true);  
  
        pw.println("HELLO");  
    }  
}
```

← Импорт нужного пакета

↑ Создание объекта

↑ Использование созданного объекта

Основы работы с файловой системой — класс File

Подключение дополнительного пакета

```
import java.io.*;
```

Создать объект этого класса

```
File f1=new File("a.txt");
```



Файл ассоциированный с объектом

Используем полученный объект

Пример использования

```
package fw;  
import java.io.*;  
public class Fw {  
  
    public static void main(String[] args) {  
        PrintWriter pw=new PrintWriter(System.out,true);  
  
        File f1=new File("a.txt");  
        pw.println(f1.getAbsolutePath());  
    }  
}
```

Создание объект этого класса

Вывод полного пути к ассоциированному файлу

Полезные методы класса File

Имя метода	Описание
<code>getName()</code>	Вернет имя файла
<code>getParent()</code>	Вернет имя каталога файла
<code>getAbsolutePath()</code>	Вернет путь к файлу
<code>exist()</code>	Вернет true если файл существует
<code>renameTo(File новое имя)</code>	Переименовывает файл
<code>delete()</code>	Удаляет файл
<code>isDirectory()</code>	Проверяет является ли объект каталогом
<code>mkdir()</code>	Создает каталог
<code>createNewFile()</code>	Создать новый файл
<code>mkdirs()</code>	Создает каталог путь для которого еще не создан
<code>list()</code>	Только для каталогов. Список остальных объектов внутри него

```
package com.gmail.tsa;
```

```
import java.io.*;
```

Пример использования File

```
public class FileOperation {
```

```
    public static void main(String[] args) {  
        PrintWriter pw = new PrintWriter(System.out, true);
```

```
        File file1 = new File("a.txt");
```

```
        try {  
            file1.createNewFile();  
        } catch (IOException e) {  
            pw.println("ERROR!");  
        }  
    }
```

Создается пустой файл

```
    File folder = new File("A");  
    folder.mkdirs();
```

Создается пустой каталог

```
    file1.renameTo(new File("b.txt"));
```

Переименовываем файл

```
    File file2 = new File("b.txt");  
    file2.delete();
```

Удаляется файл

```
    File f1 = new File(".");  
    if (f1.isDirectory()) {  
        String[] filenames = f1.list();  
        for (String filename : filenames) {  
            pw.println(filename);  
        }  
    }  
}
```

Получаем содержание каталога
и выводим на экран

Процедура для записи информации в файл

Создать объект File

Создать объект PrintWriter

Указать File или просто имя файла как назначение PrintWriter

Вывод информации по аналогии с консолью


Запись должна проводиться только в пределах блока try..catch

Пример записи информации в файл

```
package com.gmail.tsa;
```

```
import java.io.IOException;  
import java.io.PrintWriter;
```

Подключение
дополнительных пакетов



```
public class Main {
```

Подключение PrintWriter к файлу




```
    public static void main(String[] args) {
```

```
        try (PrintWriter pw = new PrintWriter("a.txt")) {
```

```
            pw.println("Hello world");
```

Запись строки в файл



```
        } catch (IOException e) {  
            System.out.println("Error");  
        }
```

```
    }
```

```
}
```

Процедура для считывания информации из файла

Создать объект File

Создать объект BufferedReader

Создать объект FileReader

Считать данные

Закрыть файл

Пример считывания данных из файла

```
package com.gmail.tsa;
```

```
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        try (BufferedReader br = new BufferedReader(new  
FileReader("a.txt"))) {
```

```
            String temp = "";
```

```
            for (; (temp = br.readLine()) != null;) {
```

```
                System.out.println(temp);
```

```
            }
```

```
        } catch (IOException e) {
```

```
            System.out.println("Error");
```

```
        }
```

```
    }
```

```
}
```

Создание
необходимых
объектов



Построчное считывание

Домашнее задание — Уровень 1

- 1) Создайте консольный «текстовый редактор» с возможностью сохранения набранного текста в файл.
- 2) Напишите метод для сохранения в текстовый файл двумерного массива целых чисел.
- 3) Реализуйте метод который выведет на экран список всех каталогов которые «лежат» в каталоге который будет параметром этого метода.

Домашнее задание — Уровень 2

- 1) Напишите метод для считывания двумерного массива из файла (размер массива заранее неизвестен, определите его сами на основе данных в файле).
- 2) Считайте текст на английском языке и выведите статистику по частоте использования букв в тексте (т. е. буква — количество использований), причем первыми должны выводиться буквы используемы чаще всего.
- 3) Напишите метод для создания в файле ASCII — арта, т. е. фигуры размером 40x40 символов заполненных символами образующими узор.

Основы программирования

Лекция 8 (Debugging и отладка)

Debugging (отладка) – это этап разработки программы, в ходе которого обнаруживают, локализуют и исправляют баги (ошибки).

Разработал: Цымбалюк А.Н.

Чтобы определить, где находится ошибка, нужно:

- Узнать текущее значение переменных
- Выявить, по какому пути выполнялась программа

Есть два взаимодополняющих метода отладки:

Использование отладчиков – специализированного ПО, которое включает в себя интерфейс пользователя для пошагового выполнения программы с остановками на некоторых строках кода, либо при достижении конкретного условия.

Вывод текущего состояния программы с помощью операторов вывода, расположенных в критических точках. Вывод производится на экран, в файл или на принтер. Вывод в файл еще называют журналированием.

Отладчик – это специализированный инструмент, который позволяет программисту наблюдать за выполнением исследуемой программы, перезапускать ее и останавливать, прогонять в замедленном темпе и т. д.

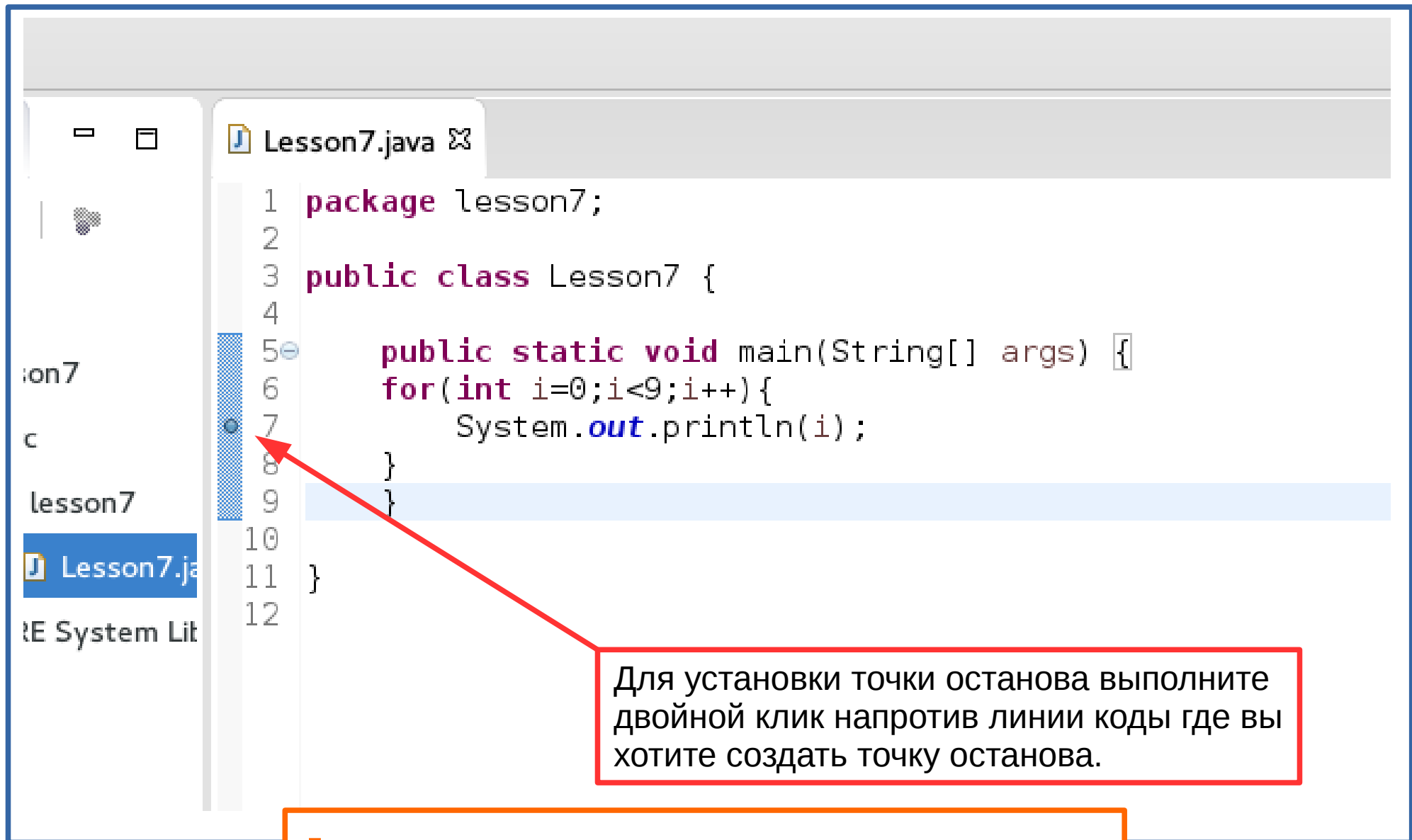
Использование встроенного отладчика в Eclipse

Для использования отладчика запустите установите сначала точки останова в программе.

В программировании, точка останова (англ. breakpoint) — это преднамеренное прерывание выполнения программы, при котором выполняется вызов отладчика (одновременно с этим, программа сама может использовать точки останова для своих нужд). После перехода к отладчику, программист может исследовать состояние программы (логи, состояние памяти, регистров процессора, стека и т. п.), с тем чтобы определить, правильно ли ведет себя программа. После остановки в отладчике, программа может быть завершена либо продолжена с того же места где произошел останов.

На практике, точка останова определяется как одно или несколько условий, при которых происходит прерывание программы. Наиболее часто используется условие останова при переходе управления к указанной инструкции программы (instruction breakpoint). Другое условие останова — операция чтения, записи или изменения указанной ячейки или диапазона ячеек памяти (data breakpoint или watchpoint).

Создание точек останова



The screenshot shows an IDE window titled "Lesson7.java". The code is as follows:

```
1 package lesson7;
2
3 public class Lesson7 {
4
5     public static void main(String[] args) {
6         for(int i=0;i<9;i++){
7             System.out.println(i);
8         }
9     }
10
11 }
12
```

A blue vertical bar on the left side of the editor indicates a breakpoint is set on line 7. A red arrow points from a text box to this breakpoint.

Для установки точки останова выполните двойной клик напротив линии кода где вы хотите создать точку останова.

! Внимание точек останова может быть несколько

Для запуска выполнения в режиме отладчика нажмите на иконку или выберите Run-> Debug As -> Java application

tor Navigate Search Project Run Window Help

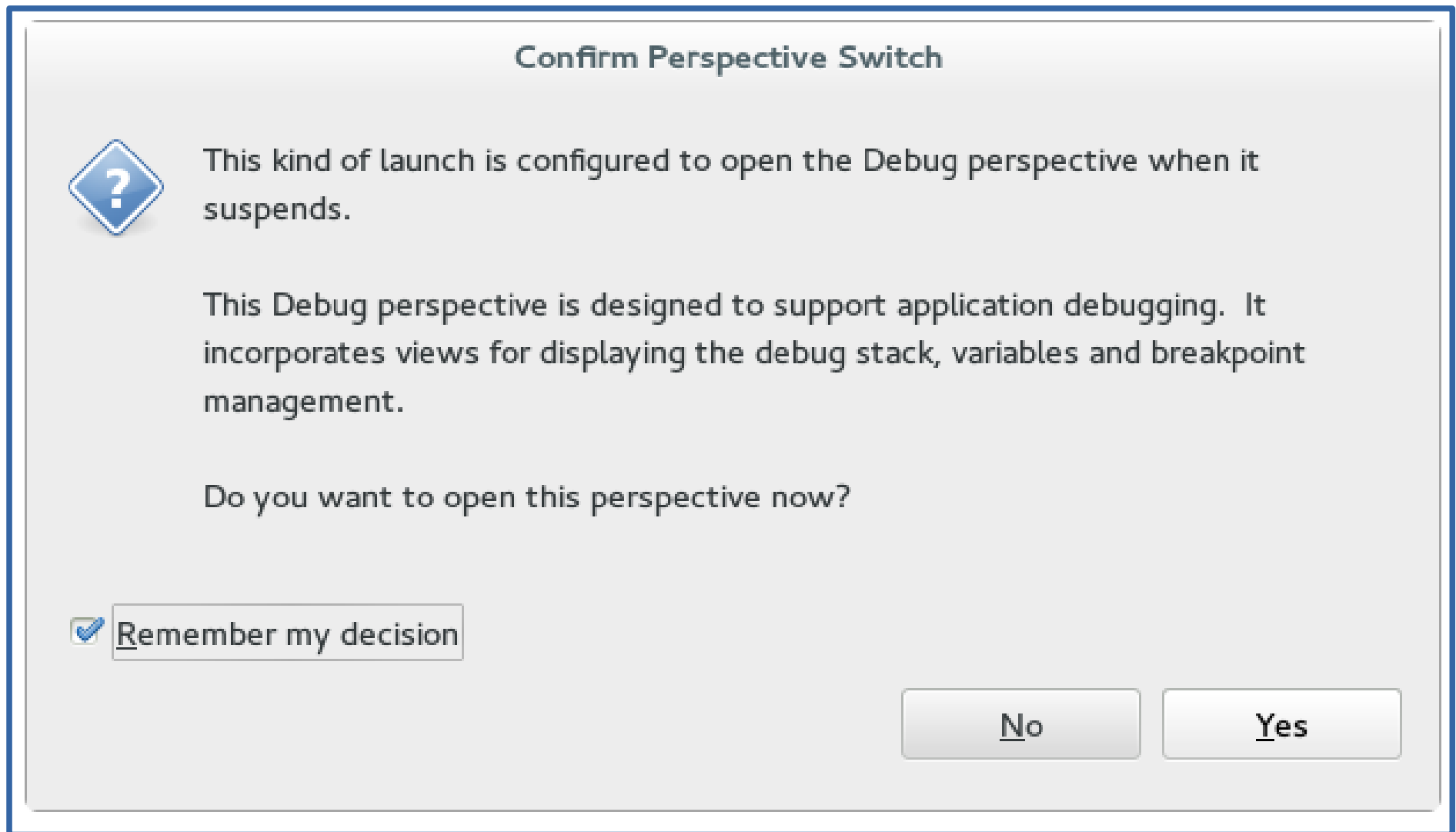


Иконка старта режима отладки

Lesson7.java

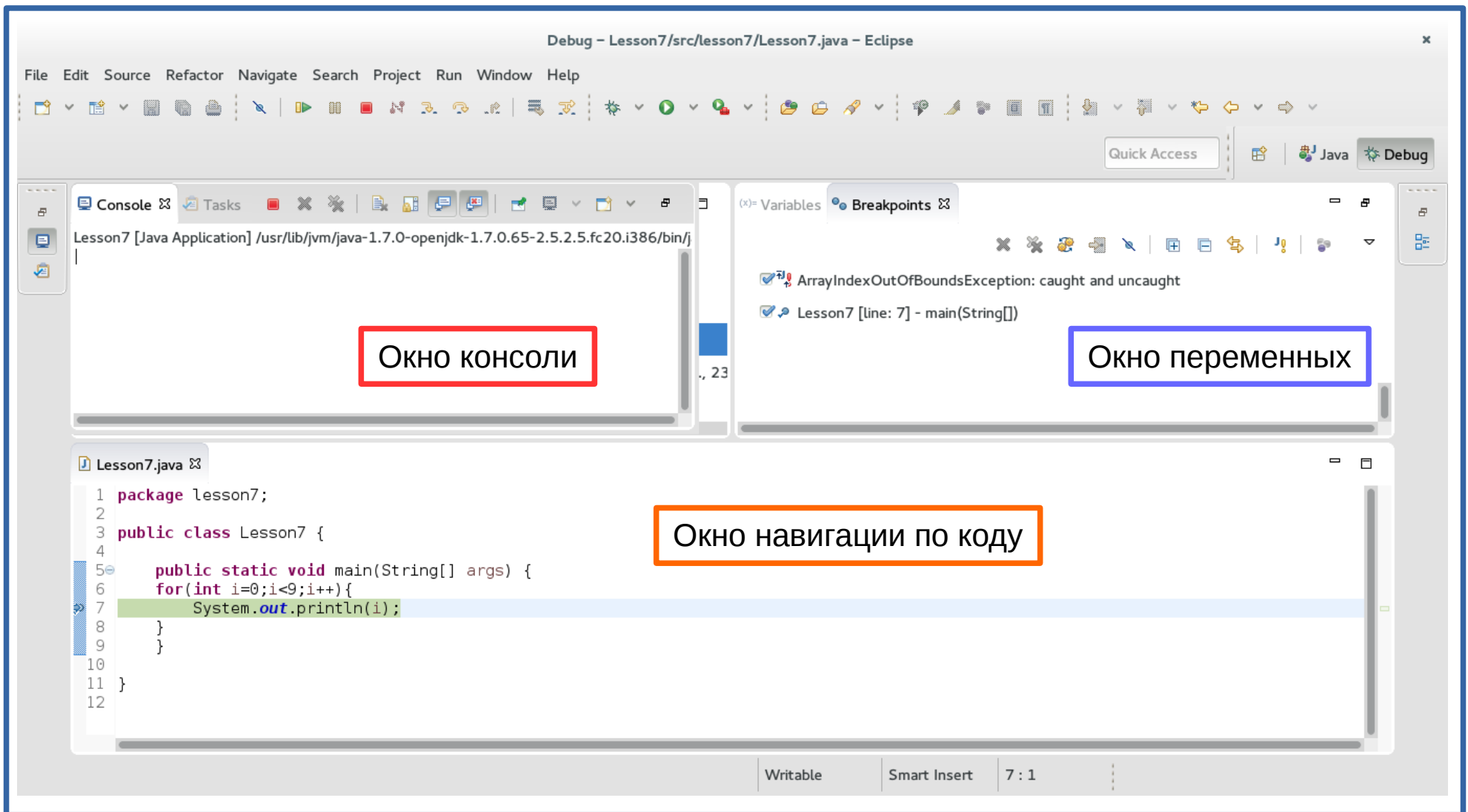
```
1 package lesson7;
2
3 public class Lesson7 {
4
5     public static void main(String[] args) {
6         for(int i=0;i<9;i++){
7             System.out.println(i);
8         }
9     }
10
11 }
12
```

Потом возникнет диалоговое окно которое предложит перевести Eclipse в режиме отладки.

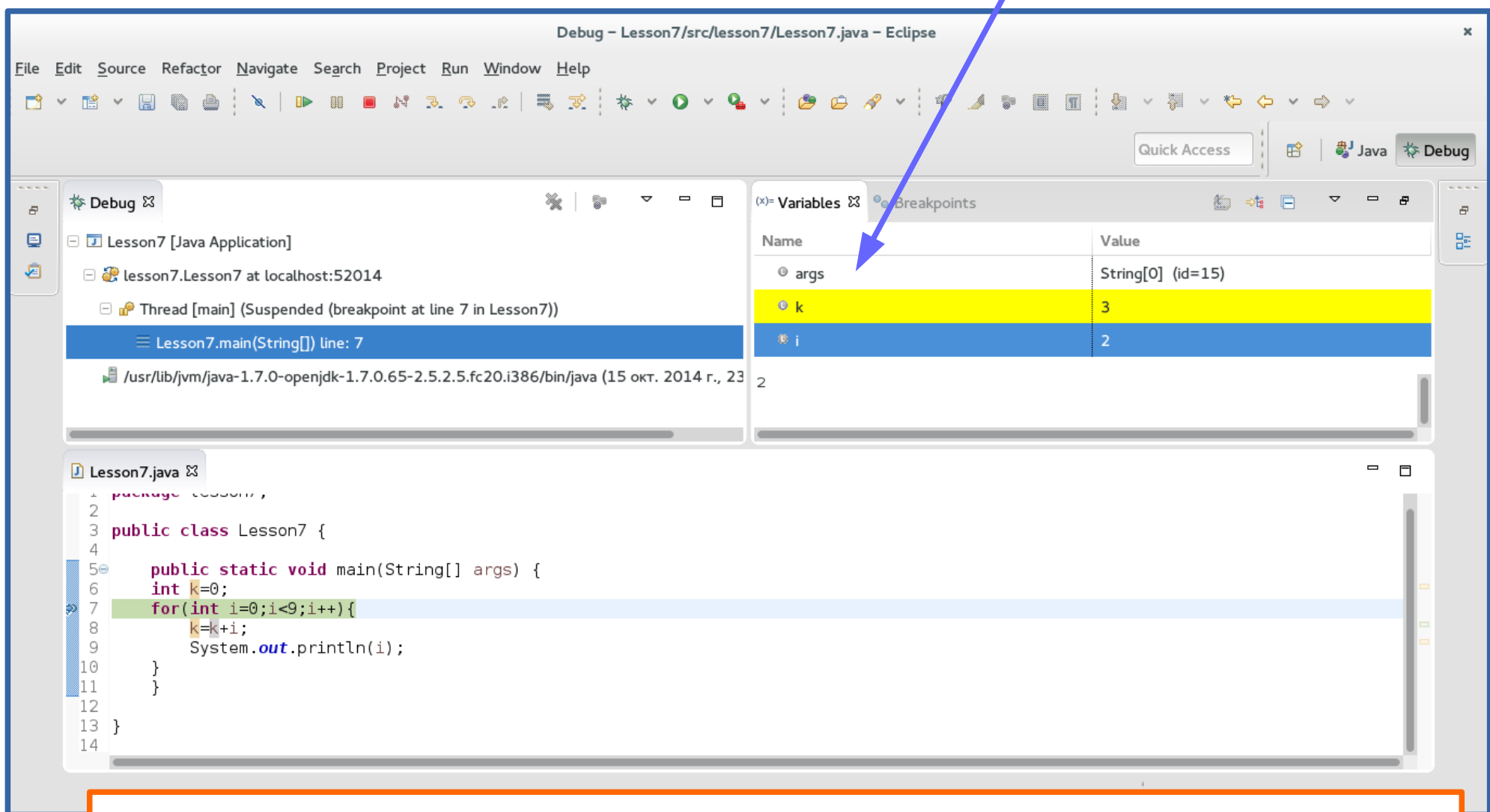


Выбираем YES и попадаем в режим отладки

Вид окна отладки Eclipse



При старте отладки есть возможность просматривать значения переменных



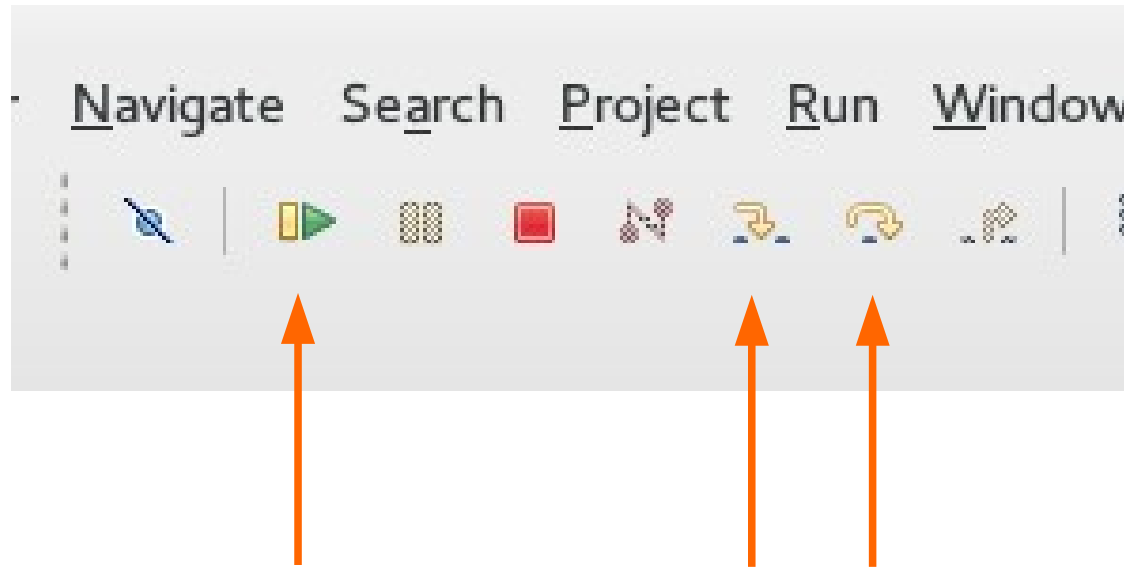
The screenshot shows the Eclipse IDE in a debug state. The top toolbar includes icons for running and debugging. The left sidebar shows the project structure with 'Lesson7 [Java Application]' and 'Thread [main] (Suspended (breakpoint at line 7 in Lesson7))'. The main editor displays the source code of 'Lesson7.java' with a breakpoint at line 7. The right sidebar shows the 'Variables' view, which lists the current state of variables:

Name	Value
args	String[0] (id=15)
k	3
i	2

A blue arrow points to the 'k' variable in the Variables view.

! Выполнив щелчок правой кнопкой мыши по переменной и выбрав пункт меню Change Value.. можно изменить значение некоторых переменных.

Пошаговое выполнение кода



Иконки для пошагового выполнения операторов

Посмотреть значение переменных в отладчике

```
package lesson7;

public class Lesson7 {

    public static void main(String[] args) {
        int k=12;
        for(int i=0;i<22;i++){
            k=k+i;
            System.out.println(k);
        }
    }
}
```

Использование директивы assert

Директива assert — вызывает преднамеренное завершение программы, при выполнении определенного логического условия.

Применяется для поиска логических ошибок на этапе проектирования ПО.

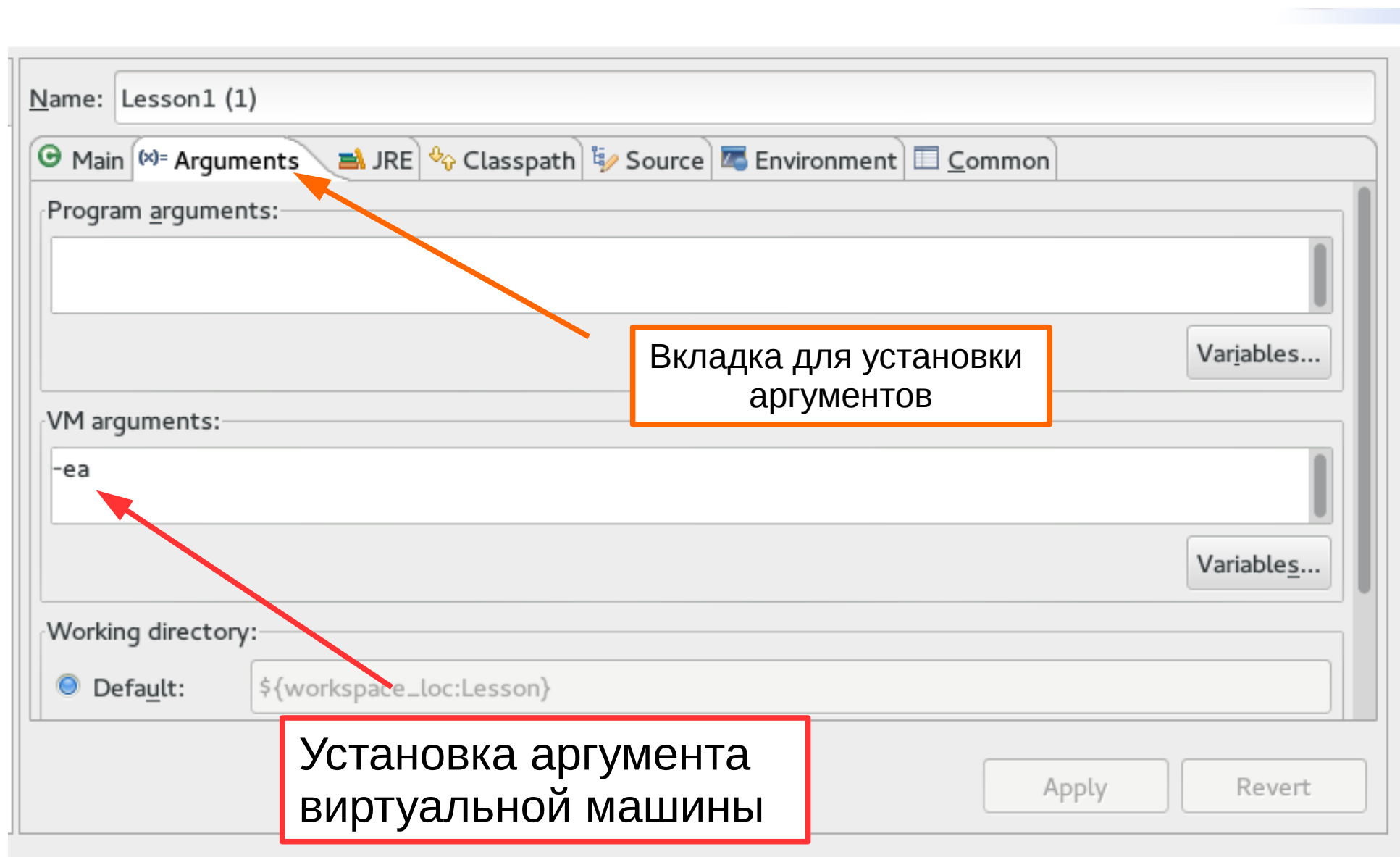
! Для применения этой директивы сначала нужно установить параметр -ea для виртуальной машины Java.

Добавление параметров VM в Eclipse

Для установки конфигурации виртуальной машины выполните такую последовательность действий
Run - > Run configuration

В появившемся диалоговом окне перейдите на вкладку (x)=Arguments в поле VM arguments введите -ea

Установка ургументов виртуальной машины



Объявление директивы assert

assert логическое условие

assert логическое условие:"Текст сообщения"

Как только логическое условие станет false — программа экстренно завершит свою работу

package lesson7; Пример использования директивы assert

```
public class Lesson7 {
```

```
public static void main(String[] args) {
```

```
for(int i=0;i<10;i++){
```

```
assert i<=5:"ERROR";
```

```
System.out.println(i);
```

```
}
```

```
}
```

```
}
```


Домашнее задание

- 1) Протестировать с помощью отладчика произвольную программу написанную вами.

Спасибо за ваше внимание.

Надеюсь данный курс поможет вам в дальнейшем продвижении в сферу программирования на одном из интереснейших языков Java.

Если курс вам понравился можете оставить отзыв, или наоборот если есть замечания выскажите их нам мы всегда открыты для диалога.

Желаем вам удачи и надеемся на дальнейшее сотрудничество. С уважением коллектив
PROG.KIEV.UA