



Дисципліна

**«Конструювання програмного
забезпечення»**

Структура дисципліни

Зміст лекцій

Модуль 1.

Використання основних конструкцій мови Java. Масиви
Робота з текстовим типом даних
Застосування принципів ООП: інкапсуляція
Призначення блоків ініціалізації. Пакети
Застосування принципів ООП: успадкування
Застосування принципів ООП: поліморфізм. Абстрактні класи та інтерфейси
Застосування механізму обробки виключень
Застосування вкладених класів. Enum
Застосування універсальних типів даних

Структура дисципліни

Зміст лекцій

Модуль 2.

Введення у фреймворк колекцій Java.
Класи-оболонки
Особливості реалізації колекції типу Map
Потокове введення/виведення даних у мові Java
Серіалізація об'єктів в Java
Процес завантаження класів в Java. Рефлексія
Класи ядра Java: Properties, System, BigInteger, BigDecimal, Locale
Логування роботи програмного забезпечення
Модульне тестування. Фреймворк JUnit

Рекомендуемая литература

1. Г.ШИЛДТ Java 8. Полное руководство, 9-е изд. – М.: ООО «И.Д. Вильямс», 2015. – 1376 с.
2. БРЮС ЭККЕЛЬ Философия Java, 4-е изд. – СПб.: Питер, 2015. – 1168 с.
3. Блинов И.Н., Романчик В.С. Java. Методы программирования. – Минск: издательство «Четыре четверти», 2013. - 896 с.
4. Электронный ресурс:
<https://docs.oracle.com/javase/tutorial/>.
5. Электронный ресурс: <https://maven.apache.org/>.

Рекомендуемая литература

6. К.С. ХОРСТМАНН, Г.КОРНЕЛЛ. Java 2. Библиотека профессионала, том 1. Основы, 7-е изд. - М: Изд. дом «Вильямс», 2007. – 896 с.
7. К.С. ХОРСТМАНН, Г.КОРНЕЛЛ. Java 2. Библиотека профессионала, том 2. Тонкости программирования, 7-е изд. - М: Изд. дом «Вильямс», 2007. – 1168 с.



Базовые конструкции языка Java.

Отличительные особенности

Базовые конструкции языка Java. Отличительные особенности

Состав:

1. *Спецификация языка Java* (Java language specification, JLS) – лексика, типы данных, основные конструкции;
2. *Спецификация Java-машины* (Java Virtual Machine, JVM);
3. *Средство разработчика* (Java Development Kit, JDK) – утилиты, стандартные библиотеки классов, демонстрационные примеры.

Ссылка на официальный сайт компании Oracle:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Интегрированные среды разработки

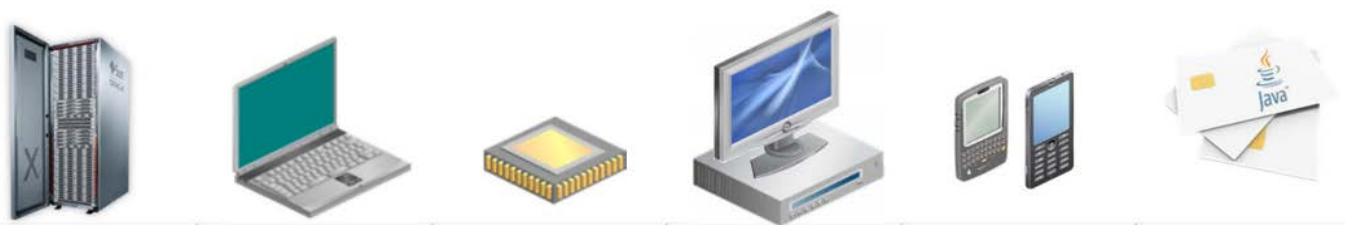
❑ **Eclipse** : <http://www.eclipse.org/downloads/>

❑ **NetBeans** :
<https://netbeans.org/downloads/index.html>

❑ **IntelliJ IDEA** :
<http://www.jetbrains.com/idea/download/>

Базовые конструкции языка Java. Отличительные особенности

Технологии и группы продуктов



Группы продуктов	Сервера	Настольные	Встроенные	TV	Мобильные	Карточки
Надстройки (API)	Java EE	JavaFX		BD-J Java TV	MSA	
Платформы	Java SE		Java ME			Java Card

Инструменты JDK и утилиты

- ❑ **Basic Tools** (appletviewer, apt, extcheck, jar, java, javac, javadoc, javah, javap, jdb)
- ❑ **Security Tools** (keytool, jarsigner, policytool, kinit, klist, ktab)
- ❑ **Internationalization Tools** (native2ascii)
- ❑ **Remote Method Invocation (RMI) Tools** (rmic, rmiregistry, rmid, serialver)
- ❑ **Java IDL and RMI-IIOP Tools** (tnameserv, idlj, orbd, servertool)
- ❑ **Java Deployment Tools** (javafxpackager, pack200, unpack200)
- ❑ **Java Web Start Tools** (javaws)
- ❑ **Java Troubleshooting, Profiling, Monitoring and Management Tools** (jcmd, jconsole, jmc, jvisualvm)
- ❑ **Java Web Services Tools** (schemagen, wsgen, wsimport, xjc)

Структура простой программы

package com.knu.lesson1; —————> Объявление пакета

/**
 * *Created by Student on 08.06.2018.* —————> Комментарий
 */

```
public class MyFirstClass { —————> Объявление класса
    public static void main(String[] arg) {
        System.out.println("Hello Student!");
    }
} —————> Тело метода main
```

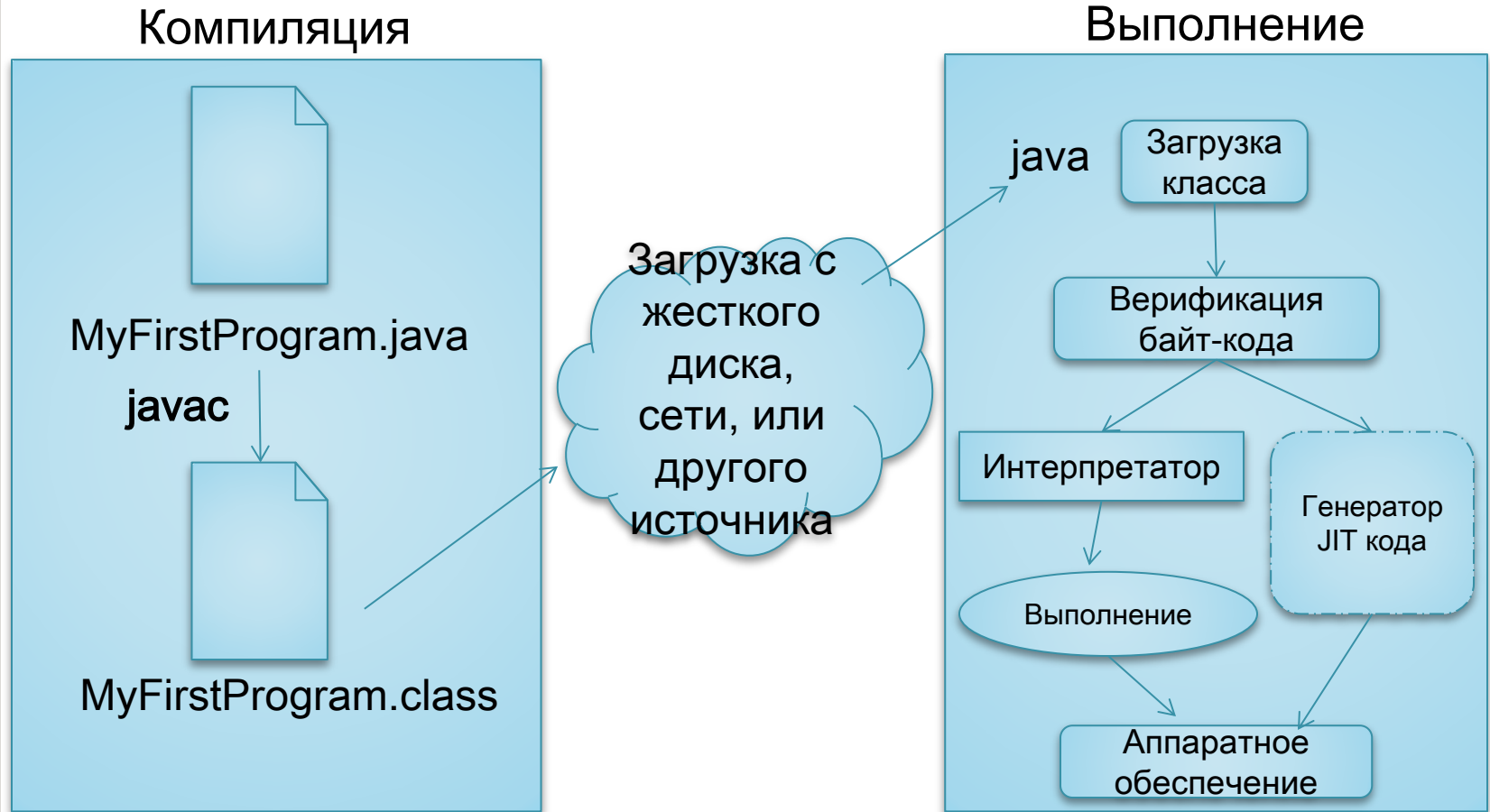
Один исходный файл может содержать описание нескольких классов, но только один из них должен быть **public**!
Имя исходного файла и имя **public** класса должны совпадать!

Базовые конструкции языка Java. Отличительные особенности

Исходная программа – это один или несколько текстовых файлов с расширением *.java*

Исполняемая программа – это наборы классов (файлы с расширением *.class*) на байт-коде → совокупность инструкций для абстрактного процессора в виде байтовых последовательностей команд этого процессора и данных к ним.

Базовые конструкции языка Java. Отличительные особенности



СТАНДАРТНЫЕ ПОТОКИ ВВОДА/ВЫВОДА

- ❑ стандартный вывод: доступ через *System.out* (объект типа **PrintStream**);
- ❑ стандартный вывод ошибки: доступ через *System.err* (объект типа **PrintStream**);
- ❑ стандартный ввод: доступ через *System.in* (объект типа **InputStream**);

Базовые конструкции языка Java. Отличительные особенности

Пример 1 (вывод строк):

```
package com.knu.lesson11;  
public class Main {  
    public static void main(String[] args){  
        System.err.print("Мустанг ");  
        System.out.println("уже здесь!");  
    }  
}
```

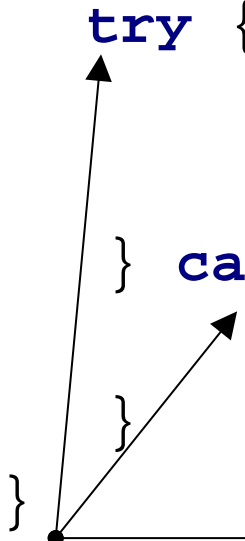
Результат

Мустанг уже здесь!

Базовые конструкции языка Java. Отличительные особенности

Пример 2 (ввод данных):

```
package com.knu.lesson12;  
  
public class Main {  
    public static void main(String[] args) {  
        try {  
            int value = System.in.read();  
            System.out.println(value);  
        } catch (IOException e) {  
            System.err.print("Ошибка: " + e);  
        }  
    }  
}
```



Результат

a

97

Во время операции ввода возможно возникновение ошибки (исключения), поэтому требуется обработка с помощью блока **try-catch**

Базовые конструкции языка Java. Отличительные особенности

Типы данных языка Java

<i>Индикатор типа</i>	<i>Размер (в байтах)</i>	<i>Значение по умолчанию</i>	<i>Диапазон или максимальное значение</i>
boolean	1	false	true, false
byte	1	0	-128 ÷ 127
char	2	'\u0000'	0 ÷ 65535
short	2	0	-32768 ÷ 32767
int	4	0	-2147483648 ÷ 2147483647
long	8	0	9223372036854775807 ($2^{63}-1$)
float	4	0.0	-3.4e+38 ÷ 3.4e+38
double	8	0.0	-1.7977e+308 ÷ 1.7977e+308

Базовые конструкции языка Java. Отличительные особенности

Описание (объявление) переменной:

<тип> <идентификатор>;

Например, **int value;**

Инициализация при описании:

<тип> <идентификатор> = <литерал> ;

Например, **double value1 = -5.7;**

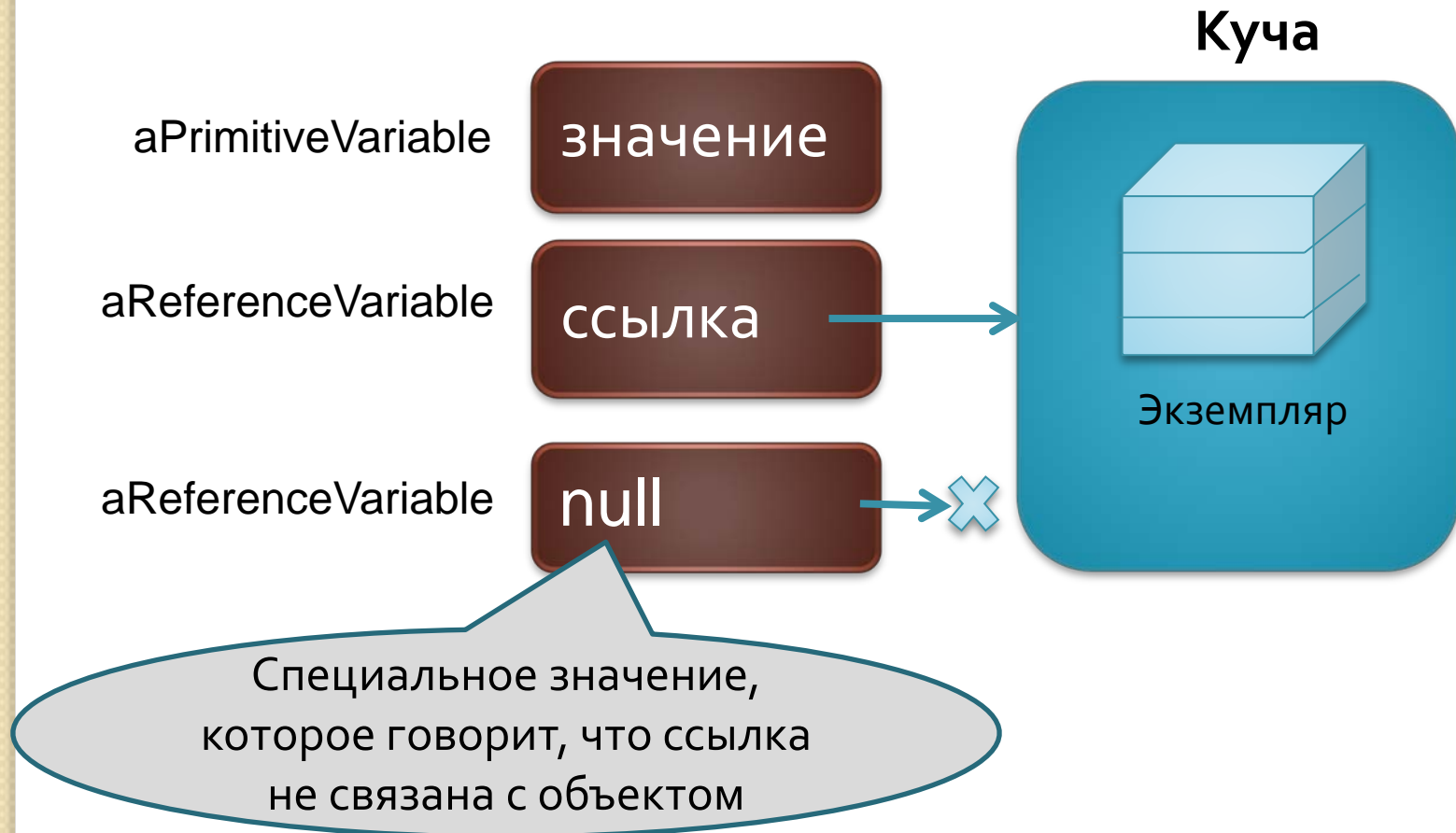
Инициализация:

<идентификатор> = <литерал> ;

Например, **value = 100;**

Переменные

- Прimitives
- Ссылочные



Переменные

- ❑ В именах переменных не могут использоваться символы арифметических и логических операторов, а также символ '#'.
#
- ❑ Применение символов '\$' и '_' допустимо, в том числе и в первой позиции имени.

Правильно:

- my\$money
- _flag
- new_string

Неправильно:

- field#
- open^flag
- 1searchIndex

ЛИТЕРАЛЫ

Литералы в Java имеют тип!

1) Целочисленные – **int**;

Например,

byte b1 = -80;

short s2 = 0b1011;

int i3 = 0x2f;

В десятичной
системе
счисления

В двоичной
системе
счисления

В шестнадцатиричной
системе счисления

Если необходимо задать литерал, значение которого превышает тип **int**, т.е. **long**, тогда в конце укажите букву L.

long k4 = 2345678923456L;

2) Вещественные – **double**.

Например,

double d6 = 77.234;

С фиксированной
точкой

double d7 = -1.5E-6;

С плавающей
точкой

Если необходим литерал для инициализации переменной типа **float**, тогда в конце необходимо указать букву F.

float f5 = 18.456F;

Базовые конструкции языка Java. Отличительные особенности

- ❑ С Java 7 в литералах можно использовать *символ подчеркивания* для повышения читабельности больших чисел.

Например,

```
long creditCardNumber = 1234_5678_9012_3456L;
```

```
long hexBytes = 0xFF_EC_DE_5E;
```

```
byte nybbles = 0b0010_0101;
```

```
float pi = 3.14_15F;
```

3) Символьный литерал – **char**

Например,

char ch1 = '3';

char ch2 = 'g';

char ch3 = '(';

char ch4 = '\u0034';

char ch5 = '\u002e';

В виде изображения
символа

В виде кода символа в
шестнадцатеричной системе
счисления

ВИДЫ ОПЕРАТОРОВ

Арифметические операторы

+	Сложение	/	Деление
+=	Сложение (с присваиванием)	/=	Деление (с присваиванием)
—	Бинарное вычитание и унарное изменение знака	%	Деление по модулю
-=	Вычитание (с присваиванием)	%=	Деление по модулю (с присваиванием)
*	Умножение	++	Инкремент
*=	Умножение (с присваиванием)	--	Декремент

Битовые операторы

	Или	>>	Сдвиг вправо
=	Или (с присваиванием)	>>=	Сдвиг вправо (с присваиванием)
&	И	>>>	Сдвиг вправо с появлением нулей
&=	И (с присваиванием)	>>>=	Сдвиг вправо с появлением нулей и присваиванием
^	Исключающее или	<<	Сдвиг влево
^=	Исключающее или (с присваиванием)	<<=	Сдвиг влево с присваиванием
~	Унарное отрицание		

Операторы отношения

➤ Применяются для сравнения символов, целых и вещественных чисел, а также для сравнения ссылок при работе с объектами.

<	Меньше	>	Больше
<=	Меньше либо равно	>=	Больше либо равно
==	Равно	!=	Не равно

Логические операторы

	Или	&&	И
!	Унарное отрицание		

- ❑ К операторам относится также оператор определения принадлежности типу **instanceof**, оператор **[]** и тернарный оператор **?:** (if-then-else).
- ❑ Логические операции выполняются над значениями типа **boolean** (**true** или **false**).
- ❑ Оператор **instanceof** возвращает значение **true**, если объект является экземпляром данного класса.

ОСОБЕННОСТИ ОПЕРАТОРОВ

1) Арифметическое деление

Результат зависит от типов операндов:

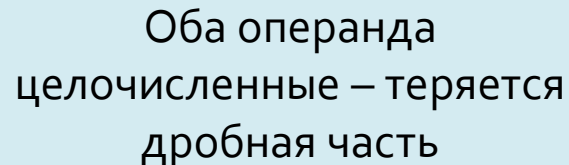
- Если оба операнды целочисленные, то результат – целочисленное значение;
- Если один из операндов вещественный, то тогда результат будет вещественный.

Например,

```
int ii = 13;
```

```
double dd = ii / 2;
```

Оба операнда
целочисленные – теряется
дробная часть



Результат: 6.0

Базовые конструкции языка Java. Отличительные особенности

2) *Получение остатка от деления*

- Может выполняться как с целочисленными значениями, так и с вещественными.

Например,

```
int a = 57;
```

```
int b = 6;
```

```
int res = a % b;
```

Результат:
res = 3

```
double d1 = 4.5;
```

```
double dd = d1 % 2.3;
```

Результат:
dd = 2.2

Базовые конструкции языка Java. Отличительные особенности

3) Логические операторы

Первая форма

!

&

|

^

Вторая форма

not

And

Or

Xor

&&

||

- Вторую форму имеют только операторы **And** и **Or** (операции «короткого замыкания»): если результат оператора можно вычислить по значению первого операнда, то значение второго операнда не оценивается.

Например,

```
int a = 57;
```

```
int b = 7;
```

```
double d = 4.5;
```

```
boolean flag = (a > 0) || (b < 5) && (dd > 1);
```

Результат: flag = true

Так как первый операнд дает **true**, то второй не высчитывается

Базовые конструкции языка Java. Отличительные особенности

4) Тернарный оператор

Любое выражение, которое вычисляет логическое значение.

`boolean_expression ? expression_1 : expression_2`

Если **true**, тогда результат значение этого выражения.

Если **false**, тогда результат значение этого выражения.

Например,

```
int x = i < 0 ? 25 : 17;
```

Пример 3:

```
public class Operators {  
    public static void main(String[] args) {  
        System.out.println("5%1=" + 5%1 + "   5%2=" + 5%2);  
        int b1 = 0xe; //14 или 1110  
        int b2 = 0x9; //9  или 1001  
        int i = 0;  
  
        System.out.println(b1 + "|" + b2 + " = " + (b1|b2));  
        System.out.println(b1 + "&" + b2 + " = " + (b1&b2));  
        System.out.println(b1 + "^" + b2 + " = " + (b1^b2));  
        System.out.println(" ~" + b2 + " = " + ~b2);  
        System.out.println(b1 + ">>" + ++i + " = " + (b1>>i));  
        System.out.println(b1 + "<<" + i + " = " + (b1<<i++));  
        System.out.println(b1 + ">>>" + i + " = " + (b1>>>i));  
    }  
}
```

Результат

выполнения:

5%1=0 5%2=1
14|9 = 15
14&9 = 8
14^9 = 7
~9 = -10
14>>1 = 7
14<<1 = 28
14>>>2 = 3

ПРИВЕДЕНИЕ (ПРЕОБРАЗОВАНИЕ) ТИПОВ

Расширяющиеся преобразования типов выполняются автоматически:

byte → short → int → long → float → double
char ↗

Сужающие преобразования типов выполняются с использованием явного приведения типов:

(<тип>) объект;

Базовые конструкции языка Java. Отличительные особенности

Например,

```
byte b = (byte) 35;
```

- ❑ Преобразование литерала выполняется автоматически при инициализации во время описания, если значение литерала входит в диапазон описываемого типа, т.е. **byte b = 35;**
- ❑ Нельзя присваивать переменной значение литерала или другой переменной более длинного типа без явного приведения типов;
- ❑ При использовании операций инкремент (++) и декремент (--), а также сокращенной формы операторов: (+=, /=, */ и т.д.) приведение типов выполняется неявно;
- ❑ При инициализации переменных с использованием арифметических операций автоматически выполняется приведение результата к объявленному типу, если его значение находится в допустимых пределах.

Базовые конструкции языка Java. Отличительные особенности

Пример 4:

```
class TypeByte {  
    public static void main(String[] args) {  
        int i = 3;  
        byte b = 1,    b1 = 1+2;  
        // short s = 304111;           // ошибка приведения типов  
        short s = (short)304111;  
        // b = b1 + 1;                 // ошибка приведения типов  
        b = (byte) (b1 + 1);  
        // b = -b;                     // ошибка приведения типов  
        b = (byte) -b;  
        b1 *=2;  
        b1++;  
        // b = i;                      // ошибка приведения типов  
        b = (byte) i;  
        b += i++;  
        float f = 1.1f;  
        b /= f;  
    }  
}
```



Операторы

ОПЕРАТОРЫ УПРАВЛЕНИЯ

➤ Оператор **if**:

Позволяет условный выбор двух операторов, выполняя один или другой, но не оба сразу.

```
if (boolexpr) { /*операторы*/ }  
else { /*операторы*/ } //может отсутствовать
```

- ✓ При отсутствии ветки **else** операторы, расположенные после окончания оператора **if**, выполняются вне зависимости от значения булевского выражения оператора **if**;
- ✓ Переменные можно объявлять внутри блока оператора – такая переменная действует только в пределах этого оператора.

➤ Циклы:

```
1. while (boolexpr) { /*операторы*/ }
2. do { /*операторы*/ }
   while (boolexpr);
3. for(expr1; boolexpr; expr3){ /*операторы*/ }
```

- ✓ Циклы выполняются, пока булевское выражение *boolexpr* равно **true**.

Особенности циклов

- ❑ проверка условия для всех циклов выполняется только один раз за одну итерацию, для **for** и **while** – перед итерацией, для **do/while** – по окончании итерации;
- ❑ цикл **for** следует использовать при необходимости выполнения алгоритма строго определенное количество раз. Цикл **while** используется в случае, когда неизвестно число итераций для достижения необходимого результата (*например*, поиск необходимого значения в массиве);

Особенности циклов

- ✓ этот цикл применяется для организации бесконечных циклов в виде `while(true);`;
- ✓ для цикла `for` не рекомендуется в цикле изменять индекс цикла;
- ✓ условие завершения цикла должно быть очевидным, чтобы цикл не «сорвался» в бесконечный цикл;
- ✓ для индексов следует применять осмысленные имена;
- ✓ циклы не должны быть слишком длинными. Такой цикл претендует на выделение в отдельный метод;
- ✓ вложенность циклов не должна превышать трех.

Базовые конструкции языка Java. Отличительные особенности

➤ Оператор **switch**

Оператор **switch** передает управление одному из нескольких операторов в зависимости от значения выражения.

```
switch( exp ) {  
    case exp1: /*операторы, если  
                exp==exp1*/  
        break;  
    case exp2: /*операторы, если  
                exp==exp2*/  
        break;  
    default: /* операторы Java */  
}
```

✓ Значения **exp1**, ..., **expN** должны быть константами и могут иметь значения типа **int**, **byte**, **short**, **char**, **enum** или **Strings**

Базовые конструкции языка Java. Отличительные особенности

Пример 4:

```
int month = 2, year = 2000, numDays = 0;
boolean hYear = true;
switch (month) {
    case 1: case 3: case 5:
    case 7: case 8: case 10:
    case 12:
        numDays = 31; break;
    case 4: case 6:
    case 9: case 11:
        numDays = 30; break;
    case 2:
        if (hYear) numDays = 29;
        else numDays = 28;
        break;
    default:
        System.out.println("Invalid month.");
        break;
}
System.out.println("Number of Days = “ + numDays);
```


Базовые конструкции языка Java. Отличительные особенности

➤ Операторы переходов

break – применяется для выхода из цикла или оператора **switch**

continue – применяется для перехода к следующей итерации цикла.

- ✓ В языке Java расширились возможности оператора **break** и **continue**, которые можно использовать с меткой. Например,

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

```
    outer: //метка цикла
```

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

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

```
            if (j == 1)
```

```
                break outer;
```

```
            System.out.println(" value of j = " + j);
```

```
        }
```

```
    } //завершение цикла с меткой outer
```

```
}
```

Выход из внешнего цикла
(цикла, отмеченного
меткой **outer**, а не из
ближайшего)