МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра "ПОВТ и АС"

Базовые типы Java

Методические указания к лабораторной работе
по дисциплинам "Программирование", "Объектно-ориентированное программирование"

Ростов-на-Дону 20 г.

Составитель: к.ф.-м.н., доц. Габрельян Б.В.
УДК 512.3

Базовые типы Java: методические указания – Ростов н/Д: Издательский центр ДГТУ, 20 . – 8 с.

Рассмотрены базовые типы Java, целочисленные, вещественные, строки, массивы. Даны задания по выполнению лабораторной работы. Методические указания предназначены для студентов направлений 090304 "Программная инженерия", 020303 "Математическое обеспечение и администрирование информационных систем".

Ответственный редактор:

Издательский центр ДГТУ, 20

1. Общая структура программы на Java

Java - чисто объектно-ориентированный язык программирования, поэтому, в частности, в нем отсутствуют внешние функции (в отличие, например, от C++ или Python). Все коды должны размещаться в методах каких-нибудь классов. Метод может быть вызван только для какого-то экземпляра (объекта) этого класса. Но когда наша программа только запускается на выполнение объектов класса еще нет. Нужен такой метод, который вел бы себя как внешняя функция, в том смысле, что его вызов не требовал бы указания объекта класса. Такими являются методы класса в целом - статические методы, поэтому точка входа в программу - функция, которая первой вызывается на выполнение при запуске desktop-приложения Java, должна быть статическим методом.

Общие правила таковы, нужно создать хотя бы один класс и объявить в нем открытый статический метод с именем main, не возвращающий значения и получающий один аргумент - массив строк. Через этот массив Java-приложение может получить значения аргументов, переданных программе при её вызове через командую строку. Например,

```
public class Lab2 {
    public static void main(String[] args) {
        // программа на Java
    }
}
```

Класс объявлен открытым в Java это означает, что исходный текст программы с объявлением этого класса обязательно должен размещаться в файле с именем, совпадающим с именем класса и расширением java, то есть в данном примере в файле Lab2.java. Таким образом, если программный проект содержит несколько открытых классов в этом проекте объявление каждого

открытого класса должно размещаться в собственном файле с соответствующим именем и расширением. Если все эти файлы расположены в одном каталоге, откомпилировать весь проект в командной строке (терминале) находясь в этом каталоге можно так

javac -cp . *.java

Здесь ключ -cp (class path) указывает на каталог (каталоги) в котором нужно искать файлы с определениями классов программы, точка означает текущий каталог. Если при установке SDK путь к текущему каталогу прописан в переменной окружения процесса CLASSPATH ключ -cp можно опустить.

2. Стандартные типы данных Java

В Java различают типы-значения и ссылочные типы. Простые встроенные типы, числовые, логический (**boolean**), символьный (**char**) относятся к типам-значениям. Переменные таких типов можно создавать статически, и они могут размещаться в программном стеке. Напротив, объекты классов можно создавать лишь динамически, с помощью операции new, и они размещаются в куче. В общем случае доступ к стеку выполняется быстрее, чем доступ к куче, поэтому типы-значения выгодно использовать в расчетах.

2.1. Числовые типы

К стандартным числовым типам относятся целочисленные: **byte**, **short**, **int**, **long** и вещественные **float**, **double** типы. Переменные соответствующих типов объявляются так же как в Си/C++:

```
int a;
float x, y;
short s1 = 5, s2 = 10;
byte b = 12;
```

```
double d1 = s1 * 2., d2;
```

Но использование неинициализированных переменных в Java контролируется строже чем в Си/C++, и инициализация считается хорошим тоном.

```
int a = 0;
float x = 0., y = .0;
double d1 = s1 * 2., d2 = 0.0;
```

Простые числовые типы позволяют выполнять вычисления быстро, но в некоторых случаях требуются не типы-значения, а ссылочные типы. Например, стандартные контейнеры Java - динамические массивы, множества, ассоциативные массивы, не могут содержать значения примитивных типов, они работают только со ссылочными типами, обобщенные типы также можно параметризовать только по ссылочным типам, но не по типам-значениям. Тогда приходится «заворачивать» значения простых типов в объекты классов.

Для каждого простого встроенного типа в Java существует соответствующий класс, позволяющий создавать объекты, содержащие в себе значение этого типа. Такие классы дают возможность преобразовать, например, целочисленную переменную в объект класса Integer, или, напротив, записать в целочисленную переменную значение объекта класса Integer. Преобразование переменной или константы в объект называют упаковкой (или обёртыванием), а соответствующие классы классами-обёртками или оболочками (wrapper-classes).

2.2. Строки

В Java реализован класс String для работы со строками символов в формате Unicode. Таким образом стандартный тип String в Java это ссылочный тип. Поэтому объявление

String str;

это объявление ссылки на объект класса String, но не сам объект. Данное объявление не создает строку (даже пустую), а создает переменную () для доступа к строке, которая будет создана или уже была создана. Причем в данном примере это неинициализированная ссылка. Обычная практика состоит в инициализации ссылки специальным значением - пустой ссылкой null.

String str = null;

Причем такое значение, показывающее, что на данный момент ссылка не связана ни с каким объектом, может использоваться для ссылок на объекты любых классов, не только класса String.

Можно создать новый объект класса String с помощью операции new. Эта операция возвращает ссылку на созданный объект и эту ссылку можно сохранить в переменной str. Например,

str = new String("Hello");

или в конструкции инициализации

String str2 = new String("World");

Можно связать ссылку с уже существующим объектом класса String

str = str2;

String str3 = str2;

В классе String реализованы различные методы, в том числе метод length(), возвращающий целое значение - число символов в строке.

int len = str3.lenght();

Но если ссылка на объект пустая, попытка вызвать через эту ссылку метод приведет к возникновению исключения во время выполнения программы

String s = null;

len = s. length(); // Исключение NullPointerException

Константная строка (литерал) - это объект класса String. Поэтому для такой строки можно вызывать методы класса String, например

```
len = "Hello".length();
```

Для класса String определена операция конкатенации (объединения) строк. Например, результатом выполнения операции "abcd" + "efgh" будет новая строка "abcdefgh". Кроме того, вторым аргументом операции могут быть константы или переменные простых типов (на самом деле и объекты любых классов). Они неявно преобразуются в строки. Например, если значение переменной x=5, то "x=" + x формирует строку "x=5". Таким образом, используя System.out.print() (без перехода на новую строку после вывода значения на экран) или System.out.println() можно выводить на стандартное устройство вывода результаты выполненных программой вычислений. Например,

```
public class Lab2 {
	public static void main(String[] args) {
		int x = 10;
		System.out.println("x = " + x + "\tx^2 = " + x*x);
	}
}
```

Выводит на экран:

```
x = 10		x^2 = 100
```

Объекты класса String в Java неизменяемые, можно получить символ строки указав его индекс (метод charAt(int index)), но нельзя изменить этот символ. Поэтому и операция конкатенации не изменяет строку, но создает новую - результат конкатенации. Существует класс StringBuffer представляющий динамически изменяемый набор символов. В этом классе есть методы setCharAt(int index, char ch), append, insert, delete и т.п. Компилятор Java

выполняет оптимизацию конкатенации строк, если в выражении эта операция встречается неоднократно, для этого он неявно использует объект класса StringBuffer.

2.3. Классы-оболочки

Для числовых типов определены следующие классы-оболочки:

Тип	**Класс**
byte	*Byte*
short	*Short*
int	*Integer*
long	*Long*
float	*Float*
double	*Double*

До Java 9 стандартным приемом упаковки значения в объект класса-оболочки было использование конструктора этого класса. Например,

```
double d = 1.73;
Double dObj = new Double(d);
Integer iObj = new Integer(-10);
```

Начиная с Java 9 рекомендуется использовать статический метод valueOf,

```
Double dObj = Double.valueOf(d);
Integer iObj = Integer.valueOf(-10);
```

Кроме того, Java поддерживает автоматическую упаковку значения простого типа в объект соответствующего класса-оболочки и автоматическую распаковку из объекта в простое значение, например

```
Doube dObj = 1.73;
double d = dObj;
```

Классы-оболочки содержат статические поля-константы для максимального (MAX_VALUE) и минимального (MIN_VALUE) возможного значений переменных соответствующего типа и, кроме того, набор методов для преобразования чисел в строки и наоборот. Для преобразования значений числовых типов в строки используются методы классов-оболочек с именем *type*Value, где вместо *type* задается имя соответствующего типа. Например,

```
Byte bObj = 65;
byte b = bObj.byteValue();
…
b = Byte.MAX_VALUE;          // b = 127
// или (менее предпочтительное)
b = bObj.MAX_VALUE;
```

Объект класса-оболочки можно преобразовать в строку с помощью метода toString, например,

```
String str = bObj.toString();      // s = "65"
```

А переменную простого типа с помощью статического метода toString(), например,

```
str = Byte.toString(b);
```

В каждом классе-оболочке предусмотрены также перегруженные (статические в отличие от *type*Value) методы parse*Type* для обратного преобразования строкового представления числа в число. Например, int i = Integer.**parseInt**(str); Статический метод toString() в классах Integer и Long имеет также форму с двумя параметрами: первый - значение, преобразуемое в строку и второй, це лочисленный, основание системы счисления (значение из диапазона 2 ÷ 32) в которой будет представлено это значение. Например,

```
String sBin = Integer.toString((int)b,2);
```

Все классы-оболочки для числовых типов являются наследниками класса **Number**.

Для символьного типа-значения char существует класс-оболочка **Character**, для логического типа boolean - класс-оболочка **Boolean**.

В классе Character определены методы, позволяющие определить, является ли символ цифрой, буквой, задана ли буква в верхнем или нижнем регистре (см. документацию по SDK). Существуют версии с аргументом типа char или int. В Unicode четко прописаны диапазоны для разных символов, например, для символов различных естественных языков, математических символов и т.д. При этом для каждого набора символов могут задаваться как основной, так и дополнительные диапазоны. Варианты методов класса Character с аргументом типа char распознают только символы основных наборов, а методы с аргументом типа int корректно работают с любыми символами Unicode, как из основных, так и из дополнительных диапазонов.

Классы-оболочки не содержат методов для изменения своих объектов, то есть их объекты являются неизменяемыми.

3. Массивы

Массивы в Java - это объекты, поэтому, работа с массивом в Java проводится через ссылку на него. Таким образом, для работы с массивом нужно первым делом создать ссылку, затем, с помощью операции new можно создать новый массив и связать с ним уже имеющуюся ссылку. Далее ссылка и индекс элемента используются для доступа к элементам массива подобно тому, как это делается в Си/C++ или Python. Индекс всегда целое значение. Например,

```
int[] a; // неинициализированная ссылка на массив целых переменных
```

```
        a = new int[5];       // создаем массив из 5 целых переменных и связываем
                              // с ним  ссылку a
// заносим значение в 0-й элемент массива (индексация всегда с нуля)
        a[0] = 125;
```

Для двумерного массива:

```
        int[][] matrix = null;        // ссылка
        matrix = new int[3][4];    // матрица: 3 строки x 4 столбца
        matrix[0][0] = 0;
        matrix[0][1] = 1;
```

Можно проводить инициализацию массивов так:

```
// создаются массив из 5 целочисленных переменных и ссылка
        int[] a = { 1, 2, 3, 4, 5 };
// двумерный массив (массив массивов одинакового размера)
        int[][] b = {
                { 1, 2, 3 },   // 0-я строка
                { 4, 5, 6 },   // 1-я строка
                { 7, 8, 9 }    // 2-я строка
        };
// более сложная структура данных (массив массивов)
        double[][] c = {
                { 1.1, 2.2, 3.3 },      // 0-я строка - массив из 3-х элементов
                { 4.4, 5.5 },           // 1-я строка - массив из 2-х элементов
                { 6.6 }                 // 2-я строка - массив из одного элемента
        };
```

Работа с массивом объектов немного сложнее, чем с массивом простых (примитивных) типов. Следующий пример демонстрирует создание массива строк:

```
String[] sArray = null;     // создать ссылку на массив строк
sArray = new String[3];  // создать массив из трех ссылок на String
aArray[0] = new String("Строка 1"); // создать первую строку
aArray[1] = new String("Строка 2"); // создать вторую строку
aArray[2] = new String("Строка 3"); // создать третью строку
```

Правда, для строк, но не любых объектов, возможна более простая запись, использующая конструкцию инициализации:

```
String[] s = { "abc", "de", "fghijk" };
```

Все классы массивов поддерживают открытое поле length, содержащее целое значение - количество элементов массива. Например,

s[0].length возвращает 3.

4. Операторы (statements) Java

Синтаксис операторов Java подобен синтаксису операторов Си/C++.

Например, нахождение наибольшего из x, y

```
int x = 5, y = 10, z = 0;
if(x > y) z = x; else z = y;
```

Оператор цикла for для вывода на экран всех элементов массива a

```
int[] a = {1,2,3,4,5};
for(int i = 0; i < a.length; ++i) System.out.println(a[i]);
```

Существует также версия for для прохода по контейнеру (foreach)

```
for(int x : a) System.out.println(x);
```

На каждой итерации x связывается с очередным значением из массива a.

Начиная с Java 10 тип локальной переменной можно не указывать явно, если компилятор может вывести этот тип из контекста (например, есть инициализирующее выражение, тогда тип выражения и будет типом переменной)

```
for(var x : a) System.out.println(x);
```

Так как на каждой итерации x связывается с элементом массива a и компилятору известно, что элементы массива имеют тип int, компилятор выводит для x тип int.

5. Аргументы командной строки

Параметр функции public static void main(String[] args) - args это ссылка на массив строк, аргументов, передаваемых программе при ее запуске через командную строку. Т.е., если программа запускается так

```
java Lab2 a b 5 3.4
```

то ей передаются 4 параметра. Эти параметры доступны (в виде строк) как элементы массива args: args[0] = "a", args[1] = "b", args[2] = "5", args[3] = "3.4". Чтобы преобразовать значение аргумента из строки в число можно использовать метод соответствующего класса-оболочки. Например,

```
int x = 0;
x = Integer.parseInt(args[2]);
double y = 0.;
y = Double.parseDouble(args[3]);
```

6. Методы класса Math.

Стандартные математические функции в Java определены в классе Math. Они реализованы как статические методы и поэтому могут вызываться даже в случае, если не создан ни один экземпляр класса Math. Статические методы класса Math представлены в таблице ниже. Кроме того, в этом же классе определены функции min(x,y), max(x,y) и константы PI (число π), E (основание экспоненты).

Функция	*Описание*	*Тип возвр. значения*

sin(x)	x - в радианах	double
cos(x)	x - в радианах	double
tan(x)	x - в радианах	double
asin(x)	арксинус от x, x из [-1.0 .. 1.0]	double
acos(x)	арккосинус от x, x из [-1.0 .. 1.0]	double
atan(x)	арктангенс от x, x из [-π/2 .. π/2]	double
exp(x)	e в степени x	double
log(x)	натуральный логарифм от x	double
sqrt(x)	корень квадратный из x	double
pow(x,y)	x в степени y, x - положительное	double
ceil(x)	наименьшее целое >= x	double
floor(x)	наибольшее целое <= x	double
round(x)	floor(x+0.5)	для x типа float - int, для x типа double - long
abs(x)	абсолютная величина x	соответствует типу x

Значения углов для тригонометрических функций указываются в радианах, а в Math определены методы преобразования значений в радианах в значения в градусах (toDegrees(x), x - в радианах) и наоборот (toRadians(x), x - в градусах).

Например, значение y=2*cos(x)+3*tn(2*x) для x = e и x = 30º:

```
double x = Math.E, y1 = 0., y2 = 0.;
y1 = 2*Math.cos(x)+3*Math.tan(2*x);
x = Math.toRadians(30);
y2 = 2* Math.cos(x)+3*Math.tan(2*x);
```

ЗАДАНИЕ 1. Напишите программу, выводящую на экран таблицу, содержащую минимальные и максимальные значения для всех простых числовых типов.

ЗАДАНИЕ 2. Дан массив из 20 целых переменных. Конкретные значения для элементов задаются в конструкции инициализации. Найти среднее геометрическое отрицательных элементов.

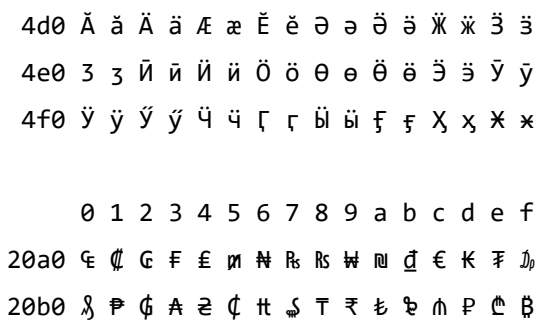
ЗАДАНИЕ 3. Радар посылает сигнал, ловит его отражение и по времени распространения сигнала определяет расстояние до предмета. Есть некоторое расстояние R. Сигнал от предметов, расположенных на больших расстояниях слишком слабый и радар не может обнаружить их обнаружить. Радар используется для охраны некоторого объекта. Определено расстояние r меньшее R. Если какой-то предмет оказывается слишком близко к объекту охраны, на расстоянии меньшем r, объявляется тревога. Даны (заданы в самой функции main) координаты (два вещественных значения) некоторого предмета в окрестности объекта (в Декартовой прямоугольной системе координат). Определить и вывести на экран сообщение: если предмет далеко - "Не обнаружен", если на расстоянии большем r, но меньшем или равном R - "Обнаружен", если на расстоянии меньшем или равном r - "Тревога".

ЗАДАНИЕ 4. Преобразуйте предыдущую программу так, чтобы можно было задавать расстояния R и r, координаты предмета через командную строку. Протестируйте программу для всех возможных вариантов попадания точки.

ЗАДАНИЕ 5. Напишите программу, получающую через командную строку целое десятичное число и отображающую на экране само это число и его представление в двоичной, восьмеричной и шестнадцатеричной системах счисления.

ЗАДАНИЕ 6. В стандарте Unicode-16 под знаки Кириллицы (включая символы украинского, сербского, македонского и т.д. языков, а также старославянского) отведен диапазон кодов 0x0400 - 0x04FF. Создайте метод для отображения в виде таблицы непрерывного набора Unicode-символов, которому через аргументы передаются стартовый код, число выводимых строк и столбцов таблицы. Напишите программу, которая использует метод для вывода символов Кириллицы (стартовый код 0x0400, 16 строк, 16 столбцов, а, затем, для вывода некоторых знаков денежных единиц (например, бразильского крузейро, индийской рупии, евро, гривны, турецкой лиры, рубля, биткоина), стартовый код 0x20a0, 2 строки, 16 столбцов. Нужно получить что-то подобное следующему

```
    0 1 2 3 4 5 6 7 8 9 a b c d e f
400 Ѐ Ё Ђ Ѓ Є Ѕ І Ї Ј Љ Њ Ћ Ќ Ѝ Ў Џ
410 А Б В Г Д Е Ж З И Й К Л М Н О П
420 Р С Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я
430 а б в г д е ж з и й к л м н о п
440 р с т у ф х ц ч ш щ ъ ы ь э ю я
450 ѐ ё ђ ѓ є ѕ і ї ј љ њ ћ ќ ѝ ў џ
460 Ѡ ѡ Ѣ ѣ Ѥ ѥ Ѧ ѧ Ѩ ѩ Ѫ ѫ Ѭ ѭ Ѯ ѯ
470 Ѱ ѱ Ѳ ѳ Ѵ ѵ Ѷ ѷ Ѹ ѹ Ѻ ѻ Ѽ ѽ Ѿ ѿ
480 Ҁ ҁ ҂ ҃ ҄ ҅ ҆ ҇ ҈ ҉ Ҋ ҋ Ҍ ҍ Ҏ ҏ
490 Ґ ґ Ғ ғ Ҕ ҕ Җ ж Ҙ ҙ Қ қ Ҝ к Ҟ ҟ
4a0 Ҡ ҡ Ң ң Ҥ ҥ Ҧ ҧ Ҩ ҩ Ҫ ҫ Ҭ ҭ Ү ү
4b0 Ұ ұ Ҳ ҳ Ҵ ҵ Ҷ ҷ Ҹ ҹ Һ һ Ҽ ҽ Ҿ ҿ
4c0 І Ӂ ӂ Ӄ ӄ Ӆ ӆ Ӈ ӈ Ӊ ӊ Ӌ ӌ Ӎ ӎ І
```



Чтобы символы отображались в консольном окне, нужно, чтобы кодировка символов и в текстовом редакторе, в котором набирается текст программы и кодировка символов в терминале (консольном окне) использовали стандарт Unicode. В MS Visual Studio Code и IntelliJ Idea это настройки по умолчанию. Чтобы добиться этого в Eclipse нужно выполнить следующее: Window - Preferences - General - Workspace - Text file encoding (установить UTF-8)

ЗАДАНИЕ 7. Написать приложение для реализации следующего алгоритма: дана строка, посчитать и вывести на экран следующие значения: количество букв, сколько из них строчных, сколько прописных, количество цифр, сколько из них арабских, сколько не арабских, количество других символов и общее количество всех символов.

ЗАДАНИЕ 8. Используйте справочную систему Java для того, чтобы разобраться с тем, как использовать методы indexOf класса String.

ЗАДАНИЕ 9. Используйте методы indexOf класса String для решения и программной реализации следующей задачи. Посчитать, сколько раз заданная подстрока встречается в указанной строке.

ЗАДАНИЕ 10. Напишите программу для решения следующей задачи. Дана строка. Найти все ее циклические перестановки. Например, для строки “abcd” это строки “abcd”, “bcda”, “cdab”, “dabc”.

Литература

1. К.Арнольд, Дж.Гослинг, Д.Холмс "Язык программирования Java. 3-е изд.". – М.: Вильямс. – 2001, 624с.
2. М.Холл, Л.Браун "Программирование для Web. Библиотека профессионала". – М.: Вильямс. – 2002, 1264с.
3. Б.Эккель "Философия Java. Библиотека программиста". – СПб.: Питер. – 2001, 880с.
4. П.Ноутон, Г.Шилдт "Java 2". – СПб.:BHV-Петербург. – 2001, 1072с.
5. Д.Бишоп "Эффективная работа: Java 2". – СПб.:Питер; К.:BHV. – 2002, 592с.

Редактор А.А. Литвинова

ЛР № 04779 от 18.05.01. В набор В печать

Объем 0,5 усл.п.л., уч.-изд.л. Офсет. Формат 60x84/16.

Бумага тип №3. Заказ № Тираж 120. Цена

Издательский центр ДГТУ

Адрес университета и полиграфического предприятия:

344010, г. Ростов-на-Дону, пл. Гагарина, 1.