ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Кафедра ЮНЕСКО по новым информационным технологиям

С.Н. Карабцев

ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ НА JAVA

Лабораторный практикум

Математический факультет

Специальность 010503 – математическое обеспечение и администрирование информационных систем

Кемерово, 2009

ЛАБОРАТОРНАЯ РАБОТА №1.
СОЗДАНИЕ ПРОГРАММ НА ЯЗЫКЕ JAVA

1. Цель работы

Получить общее представление о создании программ на языке Java и познакомиться с его основными понятиями. Изучить синтаксические единицы, основные операторы и структуру кода программы. Освоить способы компиляции исходного кода и запуска программы.

2. Методические указания

Лабораторная работа направлена на приобретение навыка написания программ на языке Java, а также умения выполнять компиляцию и запуск программы как из среды разработки (Eclipse, http://eclipse.org), так и из командной строки.

Требования к результатам выполнения лабораторного практикума:

- при выполнении задания необходимо сопровождать все реализованные процедуры и функции набором тестовых входных и выходных данных и описаниями к ним;
- компиляцию, запуск программ выполнять различными способами;
- по завершении выполнения задания составить отчет о проделанной работе.

При составлении и оформлении отчета следует придерживаться рекомендаций, представленных на следующей странице: http://unesco.kemsu.ru/student/rule/rule.html.

3. Теоретический материал

Язык Java ворвался в Интернет в конце 1995 года и немедленно завоевал популярность. Он обещал стать универсальным средством, обеспечивающим

связь пользователей с любыми источниками информации, независимо от того, где она расположена – на Web-сервере, в базе данных, хранилище данных и т.д. Этот хорошо разработанный объектно-ориентированный язык программирования поддержали все производители программного обеспечения. Он имеет встроенные средства, позволяющие решать задачи повышенной сложности такие как: работа с сетевыми ресурсами, управление базами данных, динамическое наполнение web-страниц, многопоточность приложений.

Инсталляция набора инструментальных средств Java Software Development Kit

JDK долгое время был базовым средством разработки приложений. Он не содержит никаких текстовых редакторов, а оперирует только с уже существующими java-файлами. Компилятор представлен утилитой javac (java compiler), виртуальная машина реализована программой java. Для тестовых запусков аплетов есть специальная утилита appletviewer.

Пакет Java Software Development Kit можно загрузить с web-страницы http://java.sun.com/javase/downloads/index.jsp. Способы инсталляции на разных платформах (Solaris, Windows, Linux) отличаются друг от друга. После инсталляции пакета JSDK нужно добавить имя каталога jdk\bin в список путей, по которым операционная система может найти выполняемые файлы. Правильность установки пакета можно проверить, набрав команду **java –version.** На экране должно появиться, примерно, следующее:

```
java version "1.5.0_01"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_01-b08)
Java HotSpot(TM) Client VM (build 1.5.0_01-b08, mixed mode, sharing)
```

Среда разработки программ

Для написания программ на языке Java достаточно использовать самый простой текстовый редактор, однако применение специализированных

средств разработки (Eclipse, Java WorkShop, Java Studio и др.) предоставляет большой набор полезных и удобных функций. Существует несколько способов компиляции и запуска на выполнение программ, написанных на языке Java: из командной строки или из другой программы, например, интегрированной среды разработки. Для компиляции программы из командной строки необходимо вызвать компилятор, набрав команду **javac** и указав через пробел имена компилируемых файлов:

```
javac file1.java file2.java file3.java
```

При успешном выполнении этапа компиляции в директории с исходными кодами появятся файлы с расширением .class, которые являются java байт-кодом. Виртуальная Java-машина (JVM) интерпретирует байт-код и выполняет программу. Для запуска программы необходимо в JVM загрузить основной класс, т.е. класс, который содержит функцию **main(String s[])**. Например, если в файле file1.java есть функция main(), которая располагается в классе file1, то для запуска программы после этапа компиляции необходимо набрать следующее:

```
java file1
```

Компиляцию и запуск программ из интегрированных сред разработки необходимо осуществлять в соответствии с документацией на программный продукт.

Анализ программы

Технология Java, как платформа, изначально спроектированная для Глобальной сети Internet, должна быть многоязыковой, а значит, обычный набор символов ASCII (American Standard Code for Information Interchange, Американский стандартный код обмена информацией), включающий в себя лишь латинский алфавит, цифры и простейшие специальные знаки (скобки, знаки препинания, арифметические операции и т.д.), недостаточен. Поэтому для записи текста программы применяется более универсальная кодировка Unicode. Например, если в программу нужно вставить знак с кодом 6917,

необходимо его представить в шестнадцатеричном формате (1B05) и записать: **\u1B05**.

Компилятор, анализируя программу, сразу разделяет ее на:

- пробелы (white spaces);
- комментарии (comments);
- основные лексемы (tokens).

Пробелами в данном случае называют все символы, разбивающие текст программы на лексемы. Это как сам символ пробела (space, \u0020, десятичный код 32), так и знаки табуляции и перевода строки. Они используются для разделения лексем, а также для оформления кода, чтобы его было легче читать. Например, следующую часть программы (вычисление корней квадратного уравнения):

```
double a = 1, b = 1, c = 6;
double D = b * b - 4 * a * c;

if (D >= 0) {
  double x1 = (-b + Math.sqrt (D)) / (2 * a);
  double x2 = (-b - Math.sqrt (D)) / (2 * a);
}
```

можно записать и в таком виде:

```
double a=1,b=1,c=6;double D=b*b-4*a*c;if(D>=0)
{double x1=(-b+Math.sqrt(D))/(2*a);double
x2=(-b-Math.sqrt(D))/(2*a);}
```

В обоих случаях компилятор сгенерирует абсолютно одинаковый код. Единственное соображение, которым должен руководствоваться разработчик, - легкость чтения и дальнейшей поддержки такого кода.

Комментарии не влияют на результирующий бинарный код и используются только для ввода пояснений к программе. В Java комментарии бывают двух видов: строчные и блочные. Строчные комментарии начинаются с ASCII-символов // и длятся до конца текущей строки, например:

int y=1970; // год рождения

Блочные комментарии располагаются между ASCII-символами /* и */, могут занимать произвольное количество строк. Кроме этого, существует особый вид блочного комментария – комментарий разработчика (/** комментарии*/). Он применяется для автоматического создания документации кода [1].

Лексика языка

Лексика описывает, из чего состоит текст программы, каким образом он записывается и на какие простейшие слова (лексемы) компилятор разбивает программу при анализе. Лексемы (или tokens в английском варианте) – это основные "кирпичики", из которых строится любая программа на языке Java [1]. Ниже перечислены все виды лексем в Java:

- идентификаторы (identifiers);
- ключевые слова (key words);
- литералы (literals);
- разделители (separators);
- операторы (operators).

Идентификаторы - это имена, которые даются различным элементам языка для упрощения доступа к ним. Имена имеют пакеты, классы, интерфейсы, поля, методы, аргументы и локальные переменные. Длина имени не ограничена. Идентификатор состоит из букв и цифр. Имя не может начинаться с цифры

Ключевые слова – специальные лексемы, зарегистрированные в системе для внутреннего использования, такие как abstract, default, if, private, this, boolean, implements, protected, static, try, void, native и др.

Литералы позволяют задать в программе значения для числовых, символьных и строковых выражений, а также null-литералов. В Java определены следующие виды литералов:

- целочисленный (integer);

– дробный (floating-point);
– булевский (boolean);
– символьный (character);
– строковый (string);
– null-литерал (null-literal).

Целочисленные (тип int занимает 4 байта, тип long – 8) литералы позволяют задавать целочисленные значения в десятеричном, восьмеричном и шестнадцатеричном виде. Запись нуля можно осуществить следующими способами:

– 0 (10-ричная система)
– 00 (8-ричная)
– 0x0 (16-ричная)

Если в конце литерала не стоит указателя на тип, то литерал по умолчанию имеет тип int.

Дробные литералы (тип float занимает 4 байта, тип double – 8) представляют собой числа с плавающей десятичной точкой. Дробный литерал состоит из следующих составных частей (по умолчанию имеет тип double):

– целая часть;
– десятичная точка (используется ASCII-символ точка);
– дробная часть;
– показатель степени (состоит из латинской ASCII-буквы «E» в произвольном регистре и целого числа с опциональным знаком «+» или «–»);
– окончание-указатель типа (D или F).

Символьные литералы. Представляют собой один символ и заключаются в одинарные кавычки ‘s’, ‘a’. Допускается запись через Unicode ‘\u0041’ – латинская буква “A”.

Строковые литералы состоят из набора символов и записываются в двойных кавычках: “символьный литерал”.

Null литерал может принимать всего одно значение: null. Это литерал ссылочного типа, причем эта ссылка никуда не ссылается.

Разделители – специальные символы, используемые в конструкциях языка “()”, “[]”, “{}”, “;”, “,”, “.”.

Операторы используются в различных операциях – арифметических, логических, битовых, операциях сравнения и присваивания.

Пример простой программы “Hello, world!” выглядит следующим образом:

```
public class Test {
/**
  * Основной метод, с которого начинается
  * выполнение любой Java программы.
  */
    public static void main (String args[])
    {
      System.out.println("Hello, world!");
    }
}
```

Типы данных

Java является строго типизированным языком. Это означает, что любая переменная и любое выражение имеют известный тип еще на момент компиляции. Такое строгое правило позволяет выявлять многие ошибки уже во время компиляции. Компилятор, найдя ошибку, указывает точную строку и причину ее возникновения, а динамические "баги" необходимо сначала выявить тестированием, а затем найти место в коде, которое их породило. Все типы данных разделяются на две группы. Первую составляют 8 простых или примитивных (от английского primitive) типов данных [1-3]. Они подразделяются на три подгруппы:

- целочисленные: byte, short, int, long, char;
- дробные: float, double;
- булевский: boolean.

Булевский тип представлен всего одним типом boolean, который может хранить всего два возможных значения - true и false. Величины именно этого типа получаются в результате операций сравнения.

Вторую группу составляют объектные или ссылочные (от английского reference) типы данных. Это все классы, интерфейсы и массивы.

Переменные

Переменные используются в программе для хранения данных. Любая переменная имеет три базовых характеристики: имя, тип, значение. Имя уникально идентифицирует переменную и позволяет к ней обращаться в программе. Тип описывает, какие величины может хранить переменная. Значение - текущая величина, хранящаяся в переменной на данный момент. Значение может быть указано сразу (инициализация), а в большинстве случаев задание начальной величины можно и отложить.

```
int a;
int b=0, c=2+3;
double d=e=5.5;
```

Ниже в таблице сведены данные по всем целым и дробным типам:

Название типа	Длина (байт)	Область значений
Целые типы		
byte	1	-128..127
short	2	-32768..32767
int	4	-2147483648..2147483647
long	8	-9223372036854775808.. 9223372036854775807
char	2	0..65535
Дробные типы		
float	4	3.40282347e+38f; 1.40239846e-45f
double	8	1.79769313486231570e+308; 4.94065645841246544e-324

Поток управления

В языке Java, как и в любом другом языке программирования, есть условные операторы и циклы для управления потоком. Блок, или составной оператор, произвольное количество простых операторов языка Java, заключенных в фигурные скобки. Блоки определяют область видимости своих переменных. Блоки могут быть вложенными один в другой. Однако невозможно объявить одинаково названные переменные в двух вложенных блоках.

```
public static void main(String [] args)
{  int n;
   …
   {
    int k;
    int n; // Ошибка – невозможно переопределить переменную n во вложенном цикле
   }       //переменная k определена только в этом блоке
}
```

Условные операторы

Условный оператор в языке Java имеет вид:

```
if (условие) оператор
// или
if (условие) {
оператор1;
оператор2;  }
```

Все операторы, заключенные в фигурные скобки, будут выполнены, если значение условия истинно. Общий случай условного оператора выглядит так:

```
if  (условие) оператор1 else оператор2
if (yourSale >= target)
{ performance=”Удовлетворительно”;
  Bonus = 100 + 0.01*( yourSale - target);
}
else
{ performance=”Неудовлетворительно”;
  Bonus =0;
}
```

Многовариантное ветвление представлено в виде повторяющихся операторов if … else if…

```
if (sale >=2*target)
{ performance=”Отлично”;
}
else if (sale >=1.5*target)
{ performance=”Удовлетворительно”;
}
else {System.out.println(“Вы уволены”);}
```

Неопределенные циклы

Существует два вида повторяющихся циклов, которые лучше всего подходят, если вы точно не знаете, сколько повторений должно быть выполнено. Первый из них, цикл while, выполняет тело цикла, только пока выполняется его условие.

while (условие) {операторы;}

Условие цикла while проверяется в самом начале. Следовательно, возможна ситуация, когда код, содержащийся в блоке, не будет выполнен ни разу. Если необходимо, чтобы блок выполнялся хотя бы один раз, проверку условия нужно перенести в конец. Это можно сделать с помощью цикла do/while.

do оператор while (условие);

Определенные циклы

Цикл for – распространенная конструкция для выполнения повторений, количество которых контролируется счетчиком, обновляемым на каждой итерации.

```
for (int i = 1; i < = 10; i++){
System.out.println(i);
}
```

Первый элемент оператора for обычно выполняет инициализацию счетчика, второй формулирует условие выполнения тела цикла, а третий определяет способ обновления счетчика.

```
for (int i = 10; i > 0; - i){
System.out.println("Обратный отсчет …" + i);
}
```

Многовариантное ветвление – оператор switch

Конструкция if/else может оказаться неудобной, если необходимо сделать выбор из многих вариантов. Например, создавая систему меню из трех альтернатив, можно использовать следующий код.

```
String input = JOptionPane.showInputDialog ("Выберите вариант (1, 2, 3)");
int choice = Integer.parseInt (input);
switch (choice){
case 1:
  …
  break;
case 2:
  …
  break;
case 3:
  …
  break;
default: // неверный выбор
 …
 break; }
```

Выполнение начинается с метки case, соответствующей значению переменной choice, и продолжается до следующего оператора break или конца оператора switch. Если ни одна метка не совпадает со значением переменной, выполняется раздел default. Метка case должна быть целочисленной!

Прерывание потока управления

Для выхода из цикла можно применять тот же оператор, что использовался для выхода из тела оператора switch.

```
while (balance <=100){
 balance += payments;
```

```
  if (balance .= goal) break; // выход из цикла
}
```

Пакеты

Программа на Java представляет собой набор пакетов (***packages***). Каждый пакет может включать вложенные пакеты, а так же может содержать классы и интерфейсы. Каждый пакет имеет свое пространство имен, что позволяет создавать одноименные классы в различных пакетах.

Имена бывают простыми (simple), состоящими из одного идентификатора, и составными (qualified), состоящими из последовательности идентификаторов, разделенных точкой. Составное имя любого элемента пакета составляется из составного имени этого пакета и простого имени элемента.

Простейшим способом организации пакетов и типов является обычная файловая структура. Например, исходный код класса

space.sunsystem.Moon хранится в файле ***space\sunsystem\Moon.java***

Запуск программы на JAVA стоит производить из директории, в которой содержатся пакеты. Было бы ошибкой запускать Java прямо из папки space\sunsystem и пытаться обращаться к классу Moon, несмотря на то, что файл-описание лежит именно в ней. Необходимо подняться на два уровня директорий выше, чтобы Java, построив путь из имени пакета, смогла обнаружить нужный файл.

Модуль компиляции

Модуль компиляции (complication unit)-хранится в текстовом .java-файле и является единичной порцией входных данных для компилятора. Состоит из трех частей:

- Объявление пакета;
- Import-выражения;
- Объявления верхнего уровня;

Объявление пакета указывает, какому пакету будут принадлежать все объявляемые ниже типы. Используется ключевое слово ***package***, после которого указывается полное имя пакета. Например, в файле java/lang/Object.java идет: package java.lang; что служит одновременно объявлением пакета lang, вложенного в пакет java, и указанием, что объявляемый ниже класс Object, находится в этом пакете. Так складывается полное имя класса java.lang.Object.

Область видимости типа - пакет, в котором он располагается. Внутри этого пакета допускается обращение к типу по его простому имени. Из всех других пакетов необходимо обращаться по составному имени.

Для решения этой проблемы вводятся ***import***-выражения, позволяющие импортировать типы в модуль компиляции и далее обращаться к ним по простым именам. Существует два вида таких выражений:

- импорт одного типа: import java.net.URL;
- импорт пакета: import java.awt.*;

4. Порядок выполнения работы

- изучить предлагаемый теоретический материал;
- создайте следующие программы:

1. Программа выдает на экран все аргументы, переданные ей через командную строку. Основной класс программы (с функцией main()) находится в пакете **test.first**. Привести команды компиляции и запуска программы с точным указанием места на жестком диске, откуда выполнялись эти команды.
2. Программа, в которой перебираются числа от 1 до 500 и выводятся на экран. Если число делится на 5, то вместо него выводится слово fizz, если на 7, то buzz. Если число делится на 5 и на 7, то выводить

слово fizzbuzz. Примечание*: остаток от деления в Java обозначается через символ %.

3. Программа, в которой все переданные во входную строку аргументы выводятся на экран в обратной порядке. Например, если было передано 2 аргумента – make install, то на экран должно вывестись llatsni ekam. Примечание*: для разбора слова по буквам необходимо использовать функцию charAt(). Например, str.charAt(i) вернет символ с позиции i в слове, записанном в строковую переменную str. Команда str.length() возвращает длину слова str.

5. Содержание отчета

В отчете следует указать:

1. цель работы;
2. введение;
3. программно-аппаратные средства, используемые при выполнении работы;
4. основную часть (описание самой работы), выполненную согласно требованиям к результатам выполнения лабораторного практикума;
5. заключение (описание результатов и выводы);
6. список используемой литературы.

6. Литература

1. Вязовик, Н.А. Программирование на Java. [электронный ресурс] http://www.intuit.ru/department/pl/javapl (7.09.2009).
2. Хорстманн К.С., Корнелл Г. Библиотека профессионала. JAVA 2. Том 1. Основы. 8-е издание. Пер. с англ. – М.: ООО Издательский дом “Вильямс”, 2008. – 816 с.
3. Эккель Б. Философия JAVA. – СПб.: Питер, 2003. – 971с.