# Java 2000, РТФ УПИ преподаватель А.А.Шабаршин

## 1. Введение в Java

### 1.1 Технология Java

Сегодня одной из самых быстро развивающихся технологий компьютерной индустрии становится технология Java. Технология Java, вначале своего существования, позиционировалась лишь для встраиваемых систем, но на волне огромного интереса к глобальной сети Интернет она получила новый толчок своего развитития - как средство для динамического наполнения веб-страниц. Эта технология разрабатывается в стенах фирмы Sun Microsystems с 1995 года и опирается на новый язык объектно-ориентированного программирования Java.

В основе Java лежат два принципа - независимость от программно-аппаратной платформы (т.е. от операционной системы и процессора) и безопасность создаваемых приложений.

Первый принцип реализуется через механизм виртуальной машины Java (Virtual Java Machine - JVM). Второй - прежде всего за счёт самого языка, который включает строгую типизацию, запрет адресной арифметики, автоматическую сборку "мусора", а так же запрет файловых операций при работе с Интернетом.

Java как систему программирования отличает отсутствие указателей (вместо них экземпляры класса), глобальных переменных, функций работы с памятью (автоматическая сборка мусора), строгая типизация, встроенная поддержка многозадачности, поддержка интерфейсов взамен множественного наследования. Java-код, получаемый в результате компиляции, перед выполнением проверяется специальной программой (верификатором). Это означает, что вы не сможете выполнить <искусственно> созданный Java-код, который исчерпает стек операндов или каким-либо еще способом нарушит работу Java-машины. Кстати говоря, байт-код не обязательно должен получаться в результате компиляции Java-программы! Существуют другие языки, которые можно использовать в этих целях.

На языке Java могут создаваться как автономные программы, так и так называемые апплеты - небольшие блоки кода, которые загружаются из Интернета в Web-браузер и исполняются. Java поддерживают Netscape Navigator начиная с версии 2.0 и Microsoft Internet Explorer начиная с версии 3.0. Современные браузеры (Netscape, Explorer) имеют <встроенную> Java-машину и позволяют выполнять <безопасный> Java-код.
Вызовы отдельных методов при этом запрещены. К примеру, Java-программы, выполняемые в браузере, не могут обратиться к файлам на диске. Основные проблемы, с которыми сталкивается программист при работе с <Явой>, это неэффективное исполнение Java-кода, неполная или неточная поддержка спецификаций стандартных классов, большая нагрузка на ресурсы компьютера, плохие алгоритмы сбора мусора и, как следствие, утечки памяти. Стандартная библиотека, окружающая Java-программу, постоянно изменяется. Sun ведет большую работу, направленную на ее улучшение, но на практике это означает, что Java бывает разной! Как правило, можно выделить <Яву> до версии 1.1, версию 1.1 (наиболее распространенную и встроенную в браузеры Netscape Communicator и Microsoft Explorer) и версию 1.2 (так называемая Java 2), поддержка которой в браузере возможна, но сопряжена с определенными трудностями (надо устанавливать дополнительный модуль). К тому же Microsoft приложила усилия для компрометации <Явы>, создав на платформе Windows Java-машину, не полностью соответствующую спецификациям от Sun.

По умолчанию в Microsoft Internet Explorer версии 4 и Netscape Communicator 4.7 выполнение апплетов разрешено. В стандартную поставку 5-й версии Internet Explorer Java-машина уже не входит, хотя ее можно скачать позднее из Интернета (поиграились и хватит!). Но и в 5-й версии Java-апплетам изначально выполняться разрешается. Этот факт говорит о безграничном доверии производителей браузеров к Java-технологии.

Синтаксис языка Java во многом похож на C++, но идеология языка совершенно другая. В отличие от многих других языков, компилятор Java компилирует исходные тексты не в двоичный код конкретного процессора, а в код виртуальной машины Java (JVM). При исполнении этого кода происходит либо интерпретация этого кода, как подразумевалось с самого начала и используется в бесплатном комплекте разработчика Java фирмы Sun Microsystems, либо компиляция байт-кода JVM в код конкретного процессора в момент вызова класса на исполнение (just-in-time), как сделано во многих коммерческих приложениях и в новых Web-браузерах.
Во втором случае Java программы работают существенно быстрее - но увеличивается время загрузки, так как в процессе загрузки осуществляется компиляция программы в платформозависимый код.
К тому же в языке Java реализована многозадачность (точнее многопоточность).

## 1.2 Комплект разработчика JDK

Бесплатный комплект разработчика Java (Java Development Kit - JDK) фирмы Sun Microsystems распространяется через Интернет http://java.sun.com, также его можно найти на различных компакт-дисках. JDK версии 1.0 появился в январе 1996 года, то есть официально языку Java сейчас пять лет.
У нас на кафедре проинсталирован JDK версии 1.0.2.
Sun Microsystems предлагает варианты JDK для различных платформ - Windows, Macintosh, различных UNIX-систем. Получается, что если вы напишите программу на Java, то она будет адекватно работать на любой из этих платформ. Вариант JDK для Windows-95 распространяется в самораспакующемся zip-файле, который содержит длинные имена файлов. Размер JDK 1.0 составлял 3.7 Мбайт, а версии 1.0.1 - уже 4.3 Мбайт. В JDK входят основные средства разработки приложений - компилятор, отладчик, интерпретатор автономных программ, просмотрщик апплетов и др. Также там есть стандартная библиотека классов и много примеров Java-программ с комментариями. К сожалению в JDK нет документации и средств помощи.

При распаковке JDK создаётся каталог JAVA, а в нём несколько подкаталогов. В подкаталоге DEMO располагаются примеры программ, в подкаталоге LIB - библиотека классов в файле classes.zip. Если заглянуть в него, то можно увидеть следующую иерархию классов:

```
java ─┬─ applet ── ...
      ├─ awt ──────────┬── Font.class
      ├─ io ─────── ... ├── Graphics.class
      ├─ lang ──── ... └...
      ├─ net ──── ...
      └─ util ──── ...
sun ──┬─── ...
      └ ...
```

В подкаталоге BIN располагаются все исполняемые файлы, в том числе:

javac.exe - компилятор Java,
java.exe - интерпретатор автономных программ,
appletviewer.exe - просмотрщик апплетов.

Все исполняемые файлы принимают параметры из командной строки и выводят информацию на консоль, поэтому ими удобно пользоваться в окне сессии DOS. Исходный текст программы на языке Java должен быть сохранён в файле с расширением *.java. В одном файле могут быть описаны несколько классов. Для компиляции нужно запустить компилятор с одним параметром - именем файла: javac test.java. При этом каждый класс будет откомпилирован в свой файл кодов с таким же именем, что и имя класса. Файл кодов класса имеет расширение *.class.

Как я уже говорил в Java возможно два варианта создания программ: автономная программа и апплет. Автономная программа запускается Java-интерпретатором java.exe. А апплет может быть помещён в Интернет и запущен любым браузером, поддерживающим Java. В JDK для просмотра апплетов есть программа appletviewer.exe.

## 1.3 Знакомство с WWW

World Wide Web сегодня является самым популярным ресурсом Интернета, поэтому создатели Java, для популяризации языка, ориентировались именно на него. Для представления гипертекстовой информации в WWW используется язык гипертекстовой разметки HTML. В качестве средств разметки документа в HTML используются теги - слова, заключенные в треугольные кавычки. Теги бывают открывающие и закрывающие.

```
<HTML>
<HEAD>
<TITLE>First Page</TITLE>
</HEAD>
<BODY>
   Текст, текст ...
   <A HREF="http://www.uicde.ru">Ссылка в Интернете</A>
   <IMG SRC="test.gif"> Рисунок
   Поддержка Java-апплетов в HTML появилась лишь в 1996 году
   <APPLET CODE="HelloJava.class" WIDTH=400 HEIGHT=300>
   <PARAM NAME=aa VALUE="aaa">
   <PARAM NAME=bb VALUE="bbb">
   </APPLET>
</BODY>
```

```
</HTML>
```

Для тестирования апплета с помощью appletviewer.exe требуется создать минимальный HTML файл:

```
<APPLET CODE="HelloJava.class" WIDTH=400 HEIGHT=300>
</APPLET>
```

Назовём его например example.htm. Тогда для запуска апплета HelloJava требуется в командной строке записать appletviewer example.htm и исполнить.

## 2. Программирование на Java

### 2.1 Сравнение Java и C++

### 2.2 Отличия апплетов и автономных программ

Чтобы показать отличие апплетов от автономных программ я приведу два примера:

```
import java.awt.*;
import java.applet.*;

public class HelloJava extends Applet
{
   public void paint (Graphics g)
   {
      g.drawString ("Hello, Java !",5,10);
   }
}
```

и

```
class HelloJava
{
   public static void main (String args[])
   {
      System.out.println ("Hello, Java !");
   }
}
```

Итак апплет - это класс, расширяющий класс Applet, а автономная программа - это класс, имеющий метод main.

Разберём текст примеров. Первые две строки - это команды import. С помощью них вы сообщаете компилятору о том, какие классы вы будете использовать в своей программе. Например, если в программе вы используете объекты классов Image и Graphics, можно написать:

```
        import java.awt.Font;
        import java.awt.Graphics;
```

или проще:

```
        import java.awt.*;
```

или можно поставить полное имя класса в тексте программы:

```
public class HelloJava extends java.applet.Applet
{
   public void paint (java.awt.Graphics g)
   {
      g.drawString ("Hello, Java !",5,10);
   }
}
```

С помощью ключевого слова extends реализуется наследование. В Java нет множественного наследования - у класса может быть только один родительский класс. Родительский класс в Java называется суперклассом, а дочерний класс - подклассом. Функции описываются внутри класса, функция paint является стандартным методом апплета и вызывается, когда требуется перерисовать окно апплета. Ключевое слово public означает, что, в одном случае, класс является общедоступным, а в другом - метод класса является общедоступным.

### 2.3 Типы, классы и объекты

Класс представляет собой шаблон, по которому в программе могут быть созданы объекты данного класса. Он состоит из данных (члены класса) и функций (методы класса).

Абстрактным методом называется функция класса, для которой класс не предоставляет исходного кода. Вместо этого в классе просто указывается ее имя и тип.

Абстрактный класс - это класс, который содержит один или более абстрактных методов.

Интерфейс - это абстрактный класс, все методы которого являются абстрактными. Посредством интерфейсов в языке Java реализуется нечто пожобное множественному наследованию, кторое в Jave в явном виде отсутствует.

Чтобы прдемонстрировать то, как можно создавать объект какого-либо класса, изменим код рассмотренного ранее апплета :

```
import java.awt.*;
import java.applet.*;

public class HelloJava2 extends Applet
{
   public void paint (Graphics g)
   {
      Font font = new Font("TimesRoman", Font.BOLD, 24);
      g.setFont (font);
      g.drawString ("Hello, Java !",5,10);
   }
}
```

Здесь можно увидеть создаётся объект класса Font. В классе должен быть описан соответствующий конструктор - это метод, имеющий имя, идентичное имени класса. Он вызывается в момент создания объекта этого класса. Обратите внимание на константу Font.BOLD - в Java константы класса также описываются внутри класса. Например константа BOLD опсывается в классе Font следующим образом:

```
public static final int BOLD = 1;
```

Слово public означает, что BOLD является доступным. Слово static означает, что BOLD единственный для всех объектов класса. Слово final означает, что BOLD не будет меняться. И int - тип, BOLD - имя.

В Java отсутствуют указатели и адресная арифметика. Однако запись  Graphics g  означает, что объект передаётся по ссылке. По ссылке в Java передаются объекты классов и массивы. По значению передаются только переменные стандартных типов.

В Java стандартными являются те же типы, что и в C++, но есть некоторые отличия. Всего определено 8 базовых типов :

- boolean - 8 бит (true,false)
- byte  - 8 бит (значения от -128 до 127)
- char  - 16 бит [unicode]
- short - 16 бит
- int   - 32 бита
- long  - 64 бита
- float - 32 бита
- double - 64 бита

Массивы в Java выделяются также как и объекты - с помощью слова new:

```
int a[] = new int[100];
int[] a = new int[100];
```

Следует отметить, что выделенную таким способом память освобождать не нужно - после того, как на эту область памяти перестанет что-либо ссылаться, она будет освобождена сборщиком "мусора".

Набор операций над базовыми типами почти такой же как в C++ :

```
.   - разделитель (объект.имя_члена_класса)
[]  - индекс (имя_массива[индекс_элемента])
()  - вызов функции (имя(параметры)
++  - постфиксный инкремент (переменная++)
++  - префиксный инкремент (++переменная)
```

```
--  - постфиксный декремент (переменная--)
--  - префиксный декремент (--переменная)
~   - побитовая инверсия (~выражение)
!   - логический оператор отрицания (!выражение)
instanceof - проверка на принадлежность классу (объект instanceof имя_класса)
new - оператор резервирования памяти (new имя_класса)
*   - умножение (выражение*выражение)
/   - деление (выражение/выражение)
%   - остаток от целочисленного деления (выражение%выражение)
+   - сложение (выражение+выражение)
-   - вычитание (выражение-выражение)
<<  - побитовый сдвиг влево (выражение<<выражение)
>>  - побитовый сдвиг вправо (выражение>>выражение)
>>> - побитовый сдвиг вправо с заполнением нулями (выражение>>>выражение)
<   - меньше (выражение<выражение)
>   - больше (выражение>выражение)
<=  - меньше или равно (выражение<=выражение)
>=  - больше или равно (выражение>=выражение)
==  - равенство (выражение==выражение)
!=  - неравенство (выражение!=выражение)
&   - побитовое И [AND] (выражение&выражение)
^   - побитовое исключчабщее ИЛИ [XOR] [сложение по модулю 2] (выражение^выражение)
|   - побитовое ИЛИ [OR] (выражение|выражение)
&&  - логическое И (выражение&&выражение)
||  - логическое ИЛИ (выражение||выражение)
?:  - если-то (логическое_выражение?то_выражение:иначе_выражение)
=   - присвоение (оператор=выражение)
```

## 2.4 Строки

Из библиотечных классов наиболее употребимым является класс String. Строки в Java можно складывать:

```
String s1 = "aaa";
String s2 = "bbb " + s1 + " ccc";
```

Интересно, что в выражение со строками можно записывать переменные других типов:

```
int i = 10;
double d = 3.14;
String s3 = "i=" + i + ",d=" + d + ",100=" + 100;
```

Это вызвано тем, что каждому простому типу данных соответствует замещающий класс :

- boolean - Boolean
- char - Character
- int - Integer
- long - Long
- float - Float
- double - Double

и в каждом из этих классов есть метод toString(), приводящий тип к строке. Вообще метод toString() есть в каждом библиотечном классе - в сложных объектах в строку выводятся только основные параметры.

В языке Java неявным предком всех объектов считается класс Object, в котором есть простейшая реализация метода toString().

Опишем свой класс MyClass следующим образом :

```
class MyClass
{
   int x,y;

   public MyClass(int xx,int yy)
   {
      x = xx;
      y = yy;
   }
}
```

Его неявным предком считается класс Object, в котором есть простейшая реализация метода toString(). Мы можем написать следующую программу :

```
import MyClass;

class Prog1
```

```
   {
      public static void main (String args[])
      {
         MyClass my = new MyClass(10,2);
         System.out.println ( "my = " + my );
      }
   }
```

Наша программа выведет на консоль следующую строку :

```
   my = MyClass@1393758
```

А теперь добавим в наш класс MyClass еще один метод - переопределение метода toString() :

```
   public String toString() { return "(" + x + ";" + y + ")"; }
```

Запустим программу Prog1 и увидим, что сейчас она выводит

```
   my = (10,2)
```

Если заглянуть в исходные тексты описания класса Object и посмотреть на то, как реализован метод toString(), то мы увидим следующее :

```
   public String toString() {
     return getClass().getName()+"@"+Integer.toString(hashCode()<<1>>>1,16);
   }
```

Используя такой подход вставим в нашу автономную программу Prog1 метод, выводящий на консоль имя класса любого объекта

```
   public static void printClassName ( Object obj )
   {
      System.out.println ( "The class of " + obj + " is " +
                           obj.getClass().getName() );
   }
```

А в метод main следующие три строки :

```
         printClassName("string");
         printClassName(my);
         printClassName(args);
```

Что из этого получится - попробуете на практике или дома.

## 2.5 Класс Graphics

## 2.6 Апплеты

java.applet.Applet выведен из следующих классов :
java.awt.Panel - панель для контейнера
java.awt.Container - контейнер для нескольких компонентов
java.awt.Component - клавитаура, мышь и все, что связано с окном (paint,repaint)
interface java.awt.ImageObserver - проверка при загрузке изображения

Программа апплета должна начинаться так:

```
   import java.applet.*;

   public class TestApplet extends Applet
   {
```

Название файла программы должно точно совпадать с названием класса апплете (здесь TestApplet.java).

```
    // методы, которые можно переопределить
    public void init(); // инициализация
    public void start(); // выполнение действий после создания апплета
    public void paint(Graphics g); // вызывается, когда нужно перерисовать
    public void stop(); // завершающие операции
    public void destroy(); // финальные операции
    String getAppletInfo(); // информация об авторе и версии апплета
    // другие полезные методы
    AppletContext getAppletContext(); // AppletContext общается с браузером
    AudioClip getAudioClip(URL url); // загружает файл аудиоклипа
    AudioClip getAudioClip(URL url,String clipname);
```

```
 URL getCodeBase(); // возвращает URL-адрес апплета (без имени апплета)
 URL getDocumentBase(); // возвращает URL-адрес страницы с апплетом
 Image getImage(URL url); // загружает файл изображения
 Image getImage(URL url,String imagename);
 String getParameter(String name); // считывает параметр с именем name
 String[][] getParameterInfo(); // возвращает все параметры
 boolean isActive(); // возвращает true, если апплет выполянется
 void play(URL url); // воспризводит аудиоклип
 void play(URL url, String clipname);
 void showStatus(String msg); // выводит сообщение в строку состояния
 // методы классов-предков класса Applet, которые так же используют
 Dimension size(); // возвращает размер (Component)
 void repaint(); // перерисовывает окно апплета (Component)
```

Если требуется принять численные параметры апплета, то
это можно сделать следующим образом :

```
int i = Integer.valueOf(getParameter(name)).intValue();
float f = Float.valueOf(getParameter(name)).floatValue();
```

## 2.7 Интерфейс Runnable и Потоки (Threads)

```
public class TestApplet extends Applet implements Runnable
{
   public void run(); // добавляет метод run
}
```

Для создания многопоточности применяется класс Thread

Первый способ создания потока

```
     class PrimeThread extends Thread {
         public void run() {
             // compute primes...
         }
     }
```

чтобы запустить такой поток, нужно

```
     PrimeThread p = new PrimeThread();
     p.start();
     ...
```

Второй способ создания потока

```
     class Primes implements Runnable {
         public void run() {
           ...
         }
     }
```

чтобы запустить такой поток, нужно

```
     Primes p = new Primes();
     new Thread(p).start();
     ...
```

Рассмотрим следующий пример :

```
import java.awt.*;
import java.applet.*;
import java.util.*;

public class ShowTime extends Applet implements Runnable
{
   Thread TimeThread = null;
   Font font = new Font ( "TimesRoman", Font.BOLD, 18 );
   FontMetrics fontMetrics;
   String DateTime;
   Date CurrentDateTime;
   int x,y;

   public void init ()
   {
      y = size().height / 2;
   }

   public void start ()
   {
```

```
        if (TimeThread == null)
        {
            TimeThread = new Thread(this);
            TimeThread.start();
        }
    }

    public void stop ()
    {
        if (TimeThread != null)
        {
            TimeThread.stop();
            TimeThread = null;
        }
    }

    public void run ()
    {
        while (true)
        {
           CurrentDateTime = new Date();
           repaint();
           try
           {
              TimeThread.sleep(500);
           }
           catch (InterruptedException e)
           {
           }
        }
    }

    public void paint (Graphics g)
    {
        g.setFont(font);
        fontMetrics = g.getFontMetrics();
        DateTime = CurrentDateTime.toString();
        x = (size().width - fontMetrics.stringWidth(DateTime)) / 2;
        g.drawString(DateTime, x, y);
    }

}
```

## 2.8 События мыши и клавиатуры

Для обработки событий мыши и клавиатуры в апплете переопределяются методы класса Component :

```
boolean mouseDown(Event event, int x, int y) // нажатие кнопки
boolean mouseUp(Event event, int x, int y)   // отжатие кнопки
boolean mouseMove(Event event, int x, int y) // перемещение
boolean mouseDrag(Event event, int x, int y) // перетаскивание
        (т.е. движение мыши с нажатой кнопкой)
boolean mouseExit(Event event, int x, int y) // выход за пределы

boolean keyDown(Event event, int key) // нажатие кнопки
boolean keyUp(Event event, int key)   // нажатие кнопки

if(event.id==Event.KEY_PRESS)  // нажата обычная кнопка
if(event.id==Event.KEY_ACTION) // нажата функциональная клавиша
if(event.modifiers & Event.SHIFT_MASK)!=0) // нажата Shift
if(event.modifiers & Event.CTRL_MASK)!=0)  // нажата Ctrl
if(event.modifiers & Event.META_MASK)!=0)  // нажата метаклавиша
if(event.modifiers & Event.ALT_MASK)!=0)   // нажата Alt
Event.F1...Event.F12,
Event.UP,Event.DOWN,Event.LEFT,Event.RIGHT,
Event.PGUP,Event.PGDN,Event.HOME
```

## 2.9 Шрифты

В Java существует класс шрифтов Font.
Font(String name,int style,int size)
Имена шрифтов - "Helvetica","TimesRoman","Courier","Symbol"
Стили шрифтов - Font.PLAIN, Font.BOLD, Font.ITALIC

```
String getName()
int getStyle()
int getSize()
boolean isPlain()
```

```
   boolean isBold()
   boolean isItalic()
   String toString()
```

Для получения информации об указанном шрифте можно воспользоватся классом FontMetrics

```
   FontMetrics fm = g.getFontMetrics();

   int charWidth(char data[],int offset,int len)
   int charWidth(char ch)
   int getAscent()  // высота верхнего выносного элемента
   Int getDescent() // высота нижнего выносного элемента
   Font getFont()   // возвращает шрифт
   int getHeight()  // расстояние между нижней и верхней кромкой
   int[] getWidth() // массив, содержащий ширину первых 256 символов
   int stringWidth(String str) // длина указанной строки
   String toString()
```

## 2.10 Класс Image, двойная буферизация

## 2.11 Исключения

## 2.12 Простейшие элементы управления на Java

Для создания кнопок в окне апплета используется класс Button (пакет java.awt) и метод add апплета :

```
   add(new Button("String"));
```

Обычно создание элементов управления происходит в методе init() создаваемого апплета. Чтобы определить, что кнопка нажата, используется метод action :

```
   String ButtonEvent = null;

   public boolean action (Event event, Object object)
   {
     if (event.target instanceof Button)
     {
         ButtonEvent = (String) object;
     }
     return (true);
   }
```

Для создания переключателей (Checkbox) используется класс Checkbox :

```
   add(new Checkbox()); // пустой переключатель без имени
   add(new Checkbox("Label")); // пустой переключатель с именем
   add(new Checkbox("Label,null,true)); // зачеркнутый с именем
```

Для создания радиокнопок (нажата может быть только одна) используется все тот же класс Checkbox, но кроме этого нам необходимо создать группу переключателей CheckboxGroup :

```
   CheckboxGroup RadioGroup = new CheckboxGroup();

   add(new Checkbox("Name1",RadioGroup,false));
   add(new Checkbox("Name2",RadioGroup,true));
   add(new Checkbox("Name2",RadioGroup,false));

   public boolean action (Event event, Object object)
   {
     Checkbox box;
     if (event.target instanceof Checkbox)
     {
         box = (Checkbox) event.target;
         String label = box.getLabel();
         boolean state = box.getState();
     }
     return (true);
   }
```

Для создания выпадающего меню используется класс Choice :

```
   Choice menu = new Choice();

   menu.addItem("String 1");
   menu.addItem("String 2");
   menu.addItem("String 3");
   add(menu);
```

```
   public boolean action (Event event, Object object)
   {
     Choice menu;
     if (event.target instanceof Choice)
     {
         menu = (Choice) event.target;
         String eventLabel = menu.getItem(menu.getSelectedIndex());
     }
     return (true);
   }
```

Для создания поля текстового ввода используется TextField :

```
   add(new TextField()); // пустое поле
   add(new TextField(int)); // оганичение по количеству символов
   add(new TextField(String)); // поле со строкой по умолчанию
   add(new TextField(String,int)); // поле со строкой и ограничением
```

Чтобы у поля было имя, необходимо поставить метку :

```
   add(new Label("Name: "));
   add(new TextField(50));

   public boolean action (Event event, Object object)
   {
     TextField text;
     if (event.target instanceof TextField)
     {
         text = (TextField) event.target;
         String textval = text.getText();
     }
     return (true);
   }
```

2.13 Система Layout Manager

2.14 Панели

2.15 Окна и диалоговые панели

2.16 Файлы и потоки

2.17 Работа с сетью

3 Различие версий Java

Все вышесказанное относилось к первой версии языка Java (последняя модификация - java 1.0.2). Сейчас существуют еще две: Java 1.1 (последняя модификация 1.1.8) и Java 2 (которая изначально называлась 1.2).

3.1 Новое в Java 1.1

4. Взаимодействие с другими языками

shcount.pl
---

```
#!/usr/local/bin/perl

$debug=1;

$path = "";

if ($debug==0) {
 $path = "/home/robots/cgi-bin";
}

# make a date and time strings
($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time);
$timestr = sprintf ("%02d:%02d:%02d", $hour, $min, $sec);
$datestr = sprintf ("%02d.%02d.%04d", $mday, $mon+1, $year+1900);

$ip = $ENV{REMOTE_ADDR};
```

```
if ($ENV{'REQUEST_METHOD'} eq 'GET') {
        @pairs = split(/&/, $ENV{'QUERY_STRING'});
}
elsif ($ENV{'REQUEST_METHOD'} eq 'POST') {
        read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
        @pairs = split(/&/, $buffer);
}
else {
print "This script must be called from the Web\nusing either GET or POST requests\n\n";
}

($arg1,$arg2,$arg3) = @pairs;

$counter = $arg1;
$logfile = "$counter.log";

if(-e "$counter") {
  open(CNTR1,"$counter");
  $n = 0;
  while(<CNTR1>) {
    chop $_;
    if($n==0) {
       $n = $_;
    }
  }
  close(CNTR1);
  $n = $n+1;
  open(CNTR2,">$counter");
  print CNTR2 "$n\n";
  close(CNTR2);
}
else {
  open(CNTR,">$counter");
  print CNTR "1\n";
  close(CNTR);
  $n = 1;
}

if($debug==0) {
  print "Content-type: text/html\n\n";
}
print "$n\n";


if(-e "$counter") {
  open(LOG,">>$logfile");
}
else {
  open(LOG,">$logfile");
}
print LOG "$n,$datestr,$timestr,$ip,$arg1,$arg2,$arg3\n";
close(LOG);

exit;
```

shcount.htm
---

```
<applet code=shcount.class width=40 height=13>
<param name="counter" value="cgi/shcount.pl">
<param name="name" value="page2">
<param name="bcol" value="FF0000">
<param name="scol" value="00FF00">
</applet>
```

shcount.java
---

```
/* SHCOUNT.JAVA - Alexander Shabarshin  11.12.2000 */

import java.applet.Applet;
import java.awt.*;
import java.net.*;
import java.io.*;

public class shcount extends Applet
{
 int number;
 int bcol = 0xFFFFFF;
 int scol = 0;
 int dx,dy;
 String name;
 String counter;
 String value;
 byte bvalue[];
 public void init()
 {
   name = getParameter("name");
   counter = getParameter("counter");
   bcol = Integer.valueOf(getParameter("bcol"),16).intValue();
   scol = Integer.valueOf(getParameter("scol"),16).intValue();
   Dimension dim = size();
   dx = dim.width;
   dy = dim.height;
   bvalue = new byte[10];
   try {
    URL url = new URL(getDocumentBase(),counter+"?"+name);
    InputStream is = url.openStream();
    DataInputStream s = new DataInputStream(is);
    value = s.readLine();
    value.getBytes(0,value.length(),bvalue,0);
    number = Integer.valueOf(value).intValue();
    is.close();
   }
   catch (Exception e)
   {
    showStatus("Error "+e);
   }
 }
 public void paint (Graphics g)
 {
   g.setColor(new Color(bcol));
   g.fillRect(0,0,dx,dy);
   g.setColor(new Color(scol));
   int i,j;
   int x = dx-7;
   int y = 1;
   for(i=value.length()-1;i>=0;i--)
   {
    j = bvalue[i]-48;
    switch(j)
    {
      case 8:
        drawSegment(g,x,y,4);
      case 0:
        drawSegment(g,x,y,2);
        drawSegment(g,x,y,5);
        drawSegment(g,x,y,7);
      case 7:
        drawSegment(g,x,y,1);
      case 1:
        drawSegment(g,x,y,3);
        drawSegment(g,x,y,6);
        break;
```

```
      case 6:
        drawSegment(g,x,y,5);
      case 5:
        drawSegment(g,x,y,1);
        drawSegment(g,x,y,2);
        drawSegment(g,x,y,4);
        drawSegment(g,x,y,6);
        drawSegment(g,x,y,7);
        break;
      case 2:
        drawSegment(g,x,y,1);
        drawSegment(g,x,y,3);
        drawSegment(g,x,y,4);
        drawSegment(g,x,y,5);
        drawSegment(g,x,y,7);
        break;
      case 3:
        drawSegment(g,x,y,1);
        drawSegment(g,x,y,3);
        drawSegment(g,x,y,4);
        drawSegment(g,x,y,6);
        drawSegment(g,x,y,7);
        break;
      case 9:
        drawSegment(g,x,y,1);
        drawSegment(g,x,y,7);
      case 4:
        drawSegment(g,x,y,2);
        drawSegment(g,x,y,3);
        drawSegment(g,x,y,4);
        drawSegment(g,x,y,6);
        break;
    }
    x -= 8;
   }
 }
 public void drawSegment(Graphics g,int x,int y,int n)
 {
   switch(n)
   {
     case 1: g.drawLine(x+1,y,x+4,y); break;
     case 2: g.drawLine(x,y+1,x,y+4); break;
     case 3: g.drawLine(x+5,y+1,x+5,y+4); break;
     case 4: g.drawLine(x+1,y+5,x+4,y+5); break;
     case 5: g.drawLine(x,y+6,x,y+9); break;
     case 6: g.drawLine(x+5,y+6,x+5,y+9); break;
     case 7: g.drawLine(x+1,y+10,x+4,y+10); break;
   }
 }
}
```

## 5 JavaBeans

Существует простое определение JavaBean:
JavaBean - это любой программный компонент многоразового использования, которым можно визуально манипулировать в любом средстве разработки. Изначально подразумевалось, что это будут графические компоненты (кнопки, меню, диалоги и т.д.), однако впоследствии появилась спецификация Enterprise JavaBeans, описывающая компоненты без графического представления - в них можно хранить логику работы приложения вцелом.

Что нужно сделать для установки JavaBeans? Итак,
Во-первых нужно поставить Java2 - в нашем случае это JDK1.2.2 (20 Мбайтный инсталляционный файл и 17 Мбайт

документации). Затем нужно поставить так называемый BDK (Beans Development Kit). В нашем случае это BDK1.1 (2.5 Мбайт).

Для того, чтобы класс Java превратился в компонент JavaBeans, он должен обладать некоторыми свойствами:
1) Способность к инициализации нового экземпляра;
2) Наличие принимаемого по умолчанию конструктора;
3) Возможность сериализации;
4) Удовлетворять соглашениям об именах и методах;
5) Использовать новую модель делегирования событий.

Пункт 1 означает невозможность использования в качестве JavaBean абстрактных классов или интерфейсов.

Пункт 2 означает наличие конструктора без параметров.

Пункт 3 означает обязательную реализацию интерфейса Serializable (implements Serializable).

Пункт 4 означает следующее. JavaBean не имеет открытых членов класса. Для доступа к каждому из них извне, необходимо определить пару методов чтения и записи - Type getProperty() и void setProperty(Type val).
Если соответствующее свойство класса имеет тип boolean, то метод getProperty() допустимо заменить на isProperty().
Если речь идет о массиве элементов, то допустимо описывать методы обращения к каждому элементу массива а именно: Type getProperty(int i) и void setProperty(int i,Type val).
Также в случае наличия данных "только для чтения" или "только для записи" соответствующие методы можно не определять.

Пункт 5 означает наличие у JavaBean, генерирующего события, пары методов на каждое событие добавляющее "слушателя" и удаляющего его в виде:
public void addEventListenerType(EventListenerType l)
public void removeEventListenerType(EventListenerType l)

Небольшой пример. Класс событий MyEvent:

```
public class MyEvent extends EventObject
{
 public int num = 0;
}
```

Интерфейс "слушателя" события MyEvent:

```
public interface MyListener
{
 public void eventAct(MyEvent e);
}
```

Класс, генерирующий события MyEvent:

```
public class MySource
{
 public void addMyEventListener(MyListener e);
 public void removeMyEventListener(MyListener e);
}
```

Класс, являющийся "слушателем" события MyEvent:

```
public class My implements MyListener
{
 MySource source;
 ...
 public void method() {
  source.addMyEventListener(this);
```

```
  }
  // метод из интерфейса MyListener
  public void eventAct(MyEvent e) {
   ...
  }
 }
```

В BDK1.1 есть подкаталог beanbox в котором находится одноименное java-приложение, являющееся средой тестирования компонентов JavaBeans. На примере него мы и посмотрим возможности по использованию компонентов.

Стартует приложение с помощью файла RUN.BAT. В среде Windows нужно позаботиться о том, чтобы приложению было выделено достаточно памяти под переменные среды, т.к. в BAT-файле происходит модификация переменной CLASSPATH. Кроме того может оказаться, что требуется произвести компиляцию всего пакета с помощью утилиты MAKE (в Windows - NMAKE).

После старта откроются 4 окна:
- ToolBox - список доступных компонентов;
- BeanBox - главное окно приложения;
- Properties - свойства выбранного компонента;
- Method Tracer - окно сообщений.

Теперь нам нужно выбрать в ToolBox компонент "кнопка" (например BlueButton). После этого курсор мыши превратится в крест - это значит, что теперь можно указывать в главном окне местоположение и размеры компонента. Следует заметить, что затем компонент всегда можно переместить или изменить его размеры. Далее мы устанавливаем вторую кнопку. В окне Properties в поле label для кнопок указываем имена Start и Stop соответственно. Далее выбирается специальный компонент Juggler - жонглирующий человечек. А сейчас произведем привязку кнопок к этому компоненту. Для этого выбираем кнопку "Start", а в меню Edit выбираем Events -> button push -> actionPerformed. После этого за курсором мыши последует линия, идущая от кнопки. Сейчас мы должны указать объект, на который будем действовать - это компонент Juggler. При выборе этого объекта откроется диалог EventTargetDialog со списком доступных методов компонента Juggler. Выбираем startJuggling и нажимаем Ok. Точно также кнопку "Stop" привязываем к методу stopJuggling. После этих манипуляций мы получили возможность останавливать и стартовать анимацию с помощью кнопок "Stop" и "Start" соответственно.

УПРАЖНЕНИЕ 1. Создать простейший компонент JavaBean

Создадим в папке BDK1.1\demo\my файл HelloBean.java

```
package my;

import java.awt.*;
import java.io.Serializable;

public class HelloBean extends Canvas implements Serializable
{
 public HelloBean()
 {
  setSize(40,30);
  setBackground(Color.white);
 }
}
```

Как мы можем видеть, этот класс соответствует всем необходимым соглашениям, чтобы считаться JavaBean

компонентом.

Для того, чтобы создать полноценный JavaBean, нам
понадобится файл hellobean.mk в папке BDK1.1\demo

```
CLASSFILES=my\HelloBean.class
JARFILE=..\jars\hellobean.jar

all: $(JARFILE)

$(JARFILE): $(CLASSFILES)
        echo Name: my/HelloBean.class > manifest.tmp
        echo Java-bean: true >> manifest.tmp
        jar cfm $(JARFILE) manifest.tmp my\*.class

my\HelloBean.class: my\HelloBean.java
        javac my\HelloBean.java
```

Запуск компиляции - nmake /f hellobean.mk
При отсутствии утилиты NMAKE (MAKE) можно создать
простую замену - BAT-файл (hellobean.bat):

```
        javac my\HelloBean.java
        echo Name: my/HelloBean.class > manifest.tmp
        echo Java-bean: true >> manifest.tmp
        jar cfm ..\jars\hellobean.jar manifest.tmp my\*.class
```

После сборки проекта мы получим файл hellobean.jar,
который содержит в себе простейший JavaBean компонент,
который можно загрузить в BeanBox (File->Load jar).
Но так как make создает JAR-файл сразу же в BDK1.1\jars,
то при новом старте BeanBox, компонент HelloBean уже будет
в списке ToolBox и его можно положить на рабочий стол.
После того, как мы поставим наш компонент на рабочий стол
приложения BeanBox, то увидим в окне Properties - BeanBox
список свойств нашего компонента:

```
background
foregroubd
name
font
```

Эти свойства являются свойствами класса Canvas.

УПРАЖНЕНИЕ 2. Добавить новое свойство в компонент HelloBean

Для того, чтобы добавить сюда наше свойство, например
цвет, опишем соответствующий член класса:

```
Color col = new Color(Color.white);
```

И добавим два метода для доступа к этому свойству извне:

```
public void setCol(Color c)
public Color getCol()
```

Итак, окончательный вид класса HelloBean с нашимсвойством:

```
package my;

import java.awt.*;
import java.io.Serializable;

public class HelloBean extends Canvas implements Serializable
{
 Color col = null;
 public HelloBean()
```

```
 {
  col = new Color(0,0,0);
  setSize(40,30);
  setBackground(col);
 }
 public void paint(Graphics g)
 {
  setBackground(col);
 }
 public void setCol(Color c)
 {
  col = c;
  repaint();
 }
 public Color getCol()
 {
  return col;
 }
}
```

Теперь список свойств в окне Properties - BeanBox выглядит так:

```
background
foregroubd
col
name
font
```

Меняя свойство col мы видим, что оно изменяет цвет нашего компонента и вместе с тем значение свойства background.

УПРАЖНЕНИЕ 3. Добавить возможность управления компонентом HelloBean из других компонентов посредством событий

Для того, чтобы внешние события могли менять состояние нашего компонента, нам необходимо сделать его слушателем событий, например событий класса ActionEvent. Для этого добавим методы

```
public void makeWhite(ActionEvent x)
public void makeBlack(ActionEvent x)
```

соответственно окрашивающие компонент в белый и черный цвет по приходу события ActionEvent. Окончательный вид класса:

```
package my;

import java.awt.*;
import java.awt.event.*;
import java.io.Serializable;

public class HelloBean extends Canvas implements Serializable
{
 Color col = null;
 public HelloBean()
 {
  col = new Color(0,0,0);
  setSize(40,30);
  setBackground(col);
 }
 public void paint(Graphics g)
 {
  setBackground(col);
 }
 public void setCol(Color c)
 {
```

```
  col = c;
  repaint();
 }
 public Color getCol()
 {
  return col;
 }
 public void makeWhite(ActionEvent x) {
  setCol(Color.white);
 }
 public void makeBlack(ActionEvent x) {
  setCol(Color.black);
 }
}
```

Теперь расположим на рабочем столе BeanBox наш компонент и две кнопки из списка компонентов (например BlueButton). Нажатие одной кнопки привяжем к методу makeWhite, а другой к методу makeBlack - теперь нажатием этих кнопок можно переключать состояние нашего компонента.

Следует сказать, что одно событие можно увязать с несколькими компонентами.

УПРАЖНЕНИЕ 4. Добавить возможность отправоять события из HelloBean в другие компоненты (в том числе и в HelloBean)

```
package my;

import java.awt.*;
import java.awt.event.*;
import java.io.Serializable;
import java.util.Vector;

public class HelloBean extends Canvas implements Serializable, MouseListener
{
 private Color col = null;

 private Vector pushListeners = new Vector();

 public HelloBean()
 {
  col = new Color(0,0,0);
  setSize(40,30);
  setBackground(col);
  addMouseListener(this);
 }
 public void paint(Graphics g)
 {
  setBackground(col);
 }
 public void setCol(Color c)
 {
  col = c;
  repaint();
 }
 public Color getCol()
 {
  return col;
 }
 public void makeWhite(ActionEvent x) {
  setCol(Color.white);
 }
 public void makeBlack(ActionEvent x) {
  setCol(Color.black);
 }
```

```
  public synchronized void addActionListener(ActionListener l) {
        pushListeners.addElement(l);
  }
  public synchronized void removeActionListener(ActionListener l) {
        pushListeners.removeElement(l);
  }

    //---------------------------------------------------------------------

    // Mouse listener methods.

    public void mouseClicked(MouseEvent evt) {
    }
    public void mousePressed(MouseEvent evt) {
        Vector targets;
        synchronized (this) {
          targets = (Vector) pushListeners.clone();
        }
        ActionEvent actionEvt = new ActionEvent(this, 0, null);
        for (int i = 0; i < targets.size(); i++) {
          ActionListener target = (ActionListener)targets.elementAt(i);
          target.actionPerformed(actionEvt);
        }
    }
    public void mouseReleased(MouseEvent evt) {
    }
    public void mouseEntered(MouseEvent evt) {
    }
    public void mouseExited(MouseEvent evt) {
    }
    public void mouseDragged(MouseEvent evt) {
    }
    public void mouseMoved(MouseEvent evt) {
    }

    //---------------------------------------------------------------------
}
```

Собранное приложение можно сохранить File -> Save …
Не забудьте указать расширение jar в сохраняемом файле.
Затем его можно подгружать в приложение BeanBox через
File -> Load.

Кроме того можно создать апплет, который будет аналогичен
по функциональности собранному приложению. Это делается
с помощью File -> MakeApplet. Апплет создается в каталоге
BDK1.1\beanbox\tmp\myApplet. Для тестирования запускаем
appletviewer myApplet.html (далеко не все браузеры
поддерживают Java2). Если мы посмотрим внутрь HTML файла,
то увидим:

```
<applet
    archive="./myApplet.jar,./support.jar
        ,./hellobean.jar
        ,./buttons.jar
    "
    code="MyApplet"
    width=382
    height=513
>
Trouble instantiating applet MyApplet!!
</applet>
```

Для полной функциональности нам потребуются файлы:

myApplet.html

myApplet.jar
support.jar
hellobean.jar
buttons.jar