Контрольная работа - 2 семестр

Цель работы ознакомиться с языком Java для написания простых программ.

Для решения поставленной цели, необходимо решить следующие задачи:

- ознакомиться с материалом;
- ответить на вопросы;
- выполнить задание.

Язык программирования Java

Java является объектно-ориентированным языком программирования, разработанным фирмой **Sun Microsystems** (сокращенно, Sun). Основные достоинства языка

- Наибольшая среди всех языков программирования степень переносимости программ.
- Мощные стандартные библиотеки.
- Встроенная поддержка работы в сетях (как локальных, так и Internet/Intranet).

Основные недостатки

- Низкое, в сравнении с другими языками, быстродействие, повышенные требования к объему оперативной памяти (ОП).
- Большой объем стандартных библиотек и технологий создает сложности в изучении языка.
- Постоянное развитие языка вызывает наличие как устаревших, так и новых средств, имеющих одно и то же функциональное назначение.

Основные особенности

- Java является полностью объектно-ориентированным языком. Например, C++ тоже является объектно-ориентированным, но в нем есть возможность писать программы не в объектноориентированном стиле, а в Java так нельзя.
- Реализован с использованием интерпретации P-кода (байт-кода). Т.е. программа сначала транслируется в машинонезависимый P-код, а потом интерпретируется некоторой программойинтерпретатором (виртуальная Java-машина, JVM).

Версии Java

Есть несколько версий Java (1.0, 1.1, 2.0). Последняя версия Java 2.0. Это версии языка. Существуют также версии стандартных средств разработки Javaпрограмм от Sun. Sun выпускает новые версии этих стандартных средств раз или два в год. Ранее они назывались **JDK (Java Development Kit)**, в последних версиях название изменено на **SDK (Software Development Kit)**. Официальное название текущей версии **"Java (TM) SDK, Standart Edition, Version 1.3.0"**.

Предыдущая версия имеет номер 1.2.2. Как и текущая версия она соответствует версии языка Java 2.0. SDK - это базовая среда разработки программ на Java. Она является невизуальной и имеет бесплатную лицензию на использование. Есть и визуальные среды разработки (**JBuilder, Semantec Cafe, VisualJ** и др.).

Новые версии языка и версии SDK являются расширением прежних. Это сделано для того, чтобы не возникала необходимость переписывать существующие программы. Так программа, написанная на Java 1.0 или Java 1.1, будет работать и под SDK 1.3. Правда, при компиляции некоторых таких программ могут выдаваться предупреждающие сообщения типа "Deprecated ...". Это означает, что в программе использованы возможности (классы, методы), объявленные в новой версии устаревшими (deprecated).

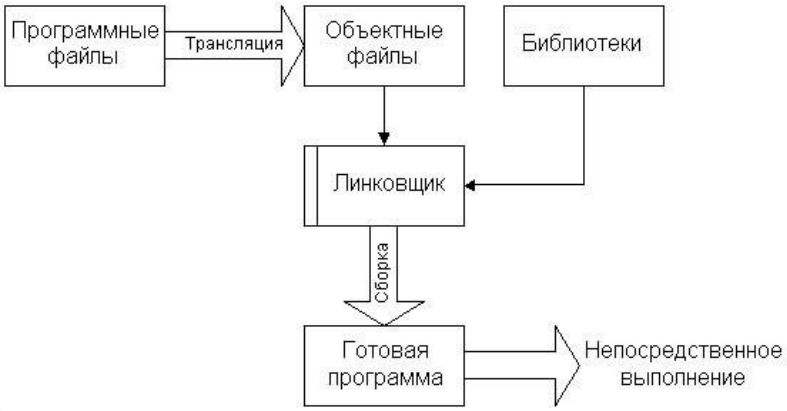
Аплеты

Аплеты являются одной из важных особенностей Java. Java позволяет строить как обычные приложения так и аплеты.

Аплет - это небольшая программа, выполняемая браузером (например, на Internet Explorer или Netscape Navigator). Аплет встраивается специальным образом в web-страничку. При подкачке такой странички браузером он выполняется виртуальной Javamашиной самого браузера. Аплеты расширяют возможности формирования web-страниц.

Жизненный цикл программы на Java

Под жизненным циклом мы будем понимать процесс, необходимый для создания работающего приложения. Для программ на Java он отличается от жизненного цикла программ на других языках программирования. Типичная картина жизненного цикла для большинства языков программирования выглядит примерно так.



Примерно такая схема действует в случае использования таких популярных языков как C++ или VB.

Для Java картина иная.

Из рисунка видно, что исходная Java-программа должна быть в файле с расширением **java**. Программа транслируется в байт-код компилятором **javac.exe**. Оттранслированная в байт-код программа имеет расширение **class**. Для запуска программы нужно вызвать интерпретатор **java.exe**, указав в параметрах вызова, какую программу ему следует выполнять. Кроме того, ему нужно указать, какие библиотеки нужно использовать при выполнении программы. Библиотеки размещены в файлах с расширением **jar** (в предыдущих версиях SDK использовались файлы ***.zip** и некоторые библиотеки все еще в таких файлах).

Структура пакета SDK

Пакеты Java SDK являются бесплатными. Текущая версия пакета может быть получена по адресу http://java.sun.com/j2se/.

Загруженная с этого адреса инсталляция пакета SDK будет, скорее всего, в виде одного большого exe-файла - файла инсталляции. Для установки SDK этот файл нужно запустить. При этом будет запрошен каталог для установки. Лучше выбрать тот каталог, который предлагается в инсталляции по умолчанию.

В выбранном Вами при инсталляции каталоге будет построена структура подкаталогов пакета SDK. Кратко рассмотрим эту структуру и выясним содержимое и назначение всех подкаталогов.

bin - каталог инструментария разработчика

demo - каталог с примерами

include - для взаимодействия с программами на C, C++

include-old - аналогично, но предыдущая версия

jre - каталог инструментария пользователя (то, что поставляется конечному пользователю при установке (deployment) готового приложения)

jre\bin - Java-машина(ы) (JVM) **jre\lib** - библиотеки Java для конечных пользователей + ряд настроечных файлов **lib** - библиотеки Java для разработчиков

Каталог bin\ содержит ряд программ, которые необходимы в процессе разработки Java-приложений. В частности, в нем расположен транслятор Java - **javac.exe**.

Каталог demo\ содержит ряд примеров, демонстрирующих те или иные возможности Java. Это, как всегда, полезно и интересно. Каталоги include\ и include-old\ мы сейчас рассматривать не будем. Они используются при совместном использовании Java и C++. Каталог jre\ нужен для поставки конечному пользователю разработанного приложения. Он состоит из двух подкаталогов jre\bin\ и jre\lib\. Подкаталог jre\bin\, в частности, содержит виртуальную машину Java (JVM) - **java.exe**, которая нужна для запуска готовых приложений. Этот же файл (java.exe) можно найти и в подкаталоге bin\. Вы можете вызывать JVM как из bin\, так и из jre\bin\. Лично я предпочитаю jre\bin\.

Подкаталог jre\lib\ содержит библиотеки, необходимые для выполнения готовых приложений, а также ряд настроечных файлов. Основная системная библиотека находится в файле **rt.jar** (очевидно, от Run Time). Ее необходимо подключать при трансляции и выполнении любой Java-программы. Вторая по важности библиотека **i18n.jar**. Если Вам необходимо, чтобы программа поддерживала русскоязычную кодировку, то

придется подключить и эту библиотеку.

Отступление. Продолжим инсталляцию SDK

Запустив программу инсталляции и указав каталог для инсталляции SDK мы сделали почти все, что нужно для установки рабочей версии инструментальных средств разработчика. Единственное действие, которое нужно еще произвести, это - настройка на русскоязычную кодировку.

Для настройки на ту или иную кодировку JVM использует файл **font.properties** из jre\lib\. Там же, в jre\lib\, находятся заготовки этих настроек для различных локализаций. Файл **font.properties.ru** содержит настройки для русскоязычной локализации.

Переименуйте **font.properties** в **font.properties.std**, а **font.properties.ru** в **font.properties**.

Это все, что требуется.

Процедура переименования font.properties.ru в font.properties не нужна, если в Window установлена по умолчанию русская локализация.

Практическая работа

Рассмотрим демонстрационный пример - приложение типа "Hello World". Но сначала выполним некоторые подготовительные действия, которые облегчат нам жизнь в дальнейшем.

Подготовительные действия

После инсталляции пакета SDK можно транслировать программу на Java, вызывая компилятор javac.exe из командной строки. Аналогично, для выполнения можно тоже вызывать JVM - java.exe.

Но по ряду причин удобнее создать отдельные batфайлы для трансляции и выполнения Javaприложений. В частности, bat-файл позволит подключить при необходимости любой набор дополнительных библиотек, когда Вам это понадобится.

Ниже приведены примеры bat-файлов трансляции и выполнения. В них все установки сделаны на основе переменной JDKHOME, которая указывает каталог, где проинсталлирован SDK. Если у Вас другой каталог, измените значение этой переменной.

Файл для трансляции (**j.bat**)

```
REM Компилятор JAVA  set
JDKHOME=d:\jdk1.3  set CLASSPATH=.;%JDKHOME%\jre\lib\rt.jar
%JDKHOME%\bin\javac %1 %2 %3 %4 %5
```

Файл для выполнения (**jr.bat**)

```
REM Запуск программы на JAVA set JDKHOME=d:\jdk1.3
set
CLASSPATH=.;%JDKHOME%\jre\lib\rt.jar;%JDKHOME%\jre\lib\i18n.ja r
%JDKHOME%\jre\bin\java -cp %CLASSPATH% %1 %2 %3 %4 %5 %6
```

Первая программа

Для подготовки Java-программы подойдет любой текстовый редактор. Существенным является только наличие в нем поддержки длинных имен файлов. В простейшем случае Javaприложение состоит из одного java-файла. Наберем простейшую Java-программу (файл Hello.java):

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

Откомпилируем программу при помощи команды j Hello.java

Если Вы набрали текст правильно, то в результате компиляции на экран ничего не будет выведено. Если же нет, то на экран будут выданы сообщения об ошибках.

Альтернативный вариант компиляции программы

(без bat-файла): javac.exe Hello.java Теперь запустим ее на выполнение jr Hello

На экран будет выведено одно слово **Hello**.

Альтернативный вариант (без bat-файла):

java.exe Hello

Замечание

Обратите внимание, что при трансляции программы задается имя файла (с расширением), а при выполнении имя файла без расширения. Очень часто начинающие работать с Java допускают здесь ошибки.

Что демонстрирует данный пример

Данный пример примитивен, но тем не менее на этом примере можно познакомиться с очень важными понятиями. Рассмотрим, что он демонстрирует.

Как мы рассмотрим подробнее позже, весь программный код в Java заключен внутри классов. Не может быть никакого программного текста (за исключением нескольких специальных директив) вне класса (или интерфейса).

Каждый файл с именем **Name.java** должен содержать класс с именем **Name** (причем, учитывается регистр). Каждый public-класс с именем **Name** должен быть в своем файле **Name.java**.

Внутри указанного файла могут быть и другие классы, но их имена должны отличаться от **Name** и они не должны быть public.

Внутри класса может быть конструкция

public static void main(String[] args) { . . . }

Это метод класса. Здесь public, static, void - это описатели, main - имя метода.

Указанный метод main является специальным случаем. При запуске Java-программы мы указываем имя класса, и Java-машина ищет этот класс среди всех доступных ей файлов *.class, и в этом классе запускает на выполнение метод main.

Описание метода main должно быть в точности таким, как приведено в примере (можно разве что изменить имя args на какое-то другое).

В скобках после имени метода указываются параметры метода. Для main-метода параметры должны быть такими как указано. Это - массив строк. При вызове программы на Java можно задать параметры вызова. Java-машина обработает их и сформирует массив строк, который будет передан в main-метод в качестве параметра.

Так, если вызвать программу командой j Hello one two 3 4

то внутри программы args будет массивом из 4-х элементов

```
args[0] = "one"
args[1] = "two"
args[2] = "3"
args[3] = "4"
```

Следующая страница

◄ Лекции - 2 семестр

Перейти на...

Идентификация личности ►