ББК 84.14

П 16

Поляков Е. В.

«Разработка приложений в WebSphere Studio Application Developer v.5»

Книга посвящена вопросам разработки приложений в среде WebSphere Studio Application Developer версии v 5 и представляет собой полный конспект одноименного курса, читаемого в Авторизованном центре обучения компании Интертраст по продуктам IBM.

Главная отличительная особенность этой книги (и курса) состоит в том, что для понимания излагаемого материала не требуется наличия опыта программирования на языке Java 2. Все базовые знания об этом языке и сопутствующих технологиях рассматриваются в настоящем издании.

Изложение материала сопровождается обширным набором иллюстраций и примеров, наглядно демонстрирующих варианты применения конкретных технологий.

Книга предназначена разработчикам приложений в среде WebSphere Studio Application Developer и может быть полезна как начинающим разработчикам, так и специалистам со стажем.

ISBN 5-7419-0071-2

Предисловие автора

Самому дорогому для меня на свете человеку,

моей жене Светлане Улазовской посвящается.

Книга, которую Вы держите в руках, посвящена вопросам создания приложений в среде WebSphere Studio Application Developer v. 5 (WSAD) - программного продукта компании IBM Company.

При написании этой книги автор ставил перед собой две цели: рассказать о собственно среде разработки приложений WSAD и при этом дать разработчикам минимально необходимые знания о языке программирования Java 2 и сопутствующих технологиях, рассматриваемых в рамках этого издания. Насколько это получилось – судить Вам.

Материал книги состоит из 11 уроков. Первые четыре урока посвящены вопросам создания во WSAD независимых (stand-alone) Java-приложений и здесь же параллельно рассматривается базис языка Java 2. В уроках 5-6 рассказывается о создании и организации Web-проектов во WSAD. Уроки с 7 по 11 можно считать практическим введением в ряд Java-технологий (сервлеты, JSP, EJB, Web-службы и Struts). Для более глубокого «погружения» в эти технологии желательно ознакомиться с дополнительными материалами, приведенными в указанных уроках и разделе «Литература» (см. раздел **Ошибка! Источник ссылки не найден.**).

Автор будет искренне рад Вашим замечаниям и предложениям по содержанию книги и любой информации технического характера по затронутым в книге вопросам. Направляйте их по e-mail: EPolyakov@inttrust.ru или «оставляйте» на WWW-сервере нашей компании http://www.intertrust.ru.

Автор выражает искреннюю благодарность и признательность всем, кто оказывал прямую или косвенную помощь в работе над книгой. Особые благодарности ведущим сотрудникам компании InterTrust Co.: Ионцеву Н. Н. (светлая ему память), Пучкову П.А., Иванову Д.Ю., Макалкину М.Ю., Сендеровичу Л.М. – без помощи которых, книга выглядела бы другой. Специальные благодарности: Бреусу И. Б., Богданову К. А., Зыковой Н. П., Поляковой Г. Д. и Полякову В. Г. за существенную помощь, оказанную при написании настоящей книги.

С уважением, Е. Поляков.

1 Урок 1. Создание независимых (stand-alone) Java-программ в рамках WebSphere Studio Application Developer

Данный урок посвящен первоначальному знакомству с продуктом WebSphere Studio Application Developer v 5 (WSAD) и созданию в нем первого независимого (stand-alone) Java-приложения.

После завершения урока слушатели должны уметь:

- создавать в WSAD независимые Java-приложения;
- тестировать Java-приложения в рамках WSAD.

В этом уроке мы будем заниматься созданием простого приложения в рамках WebSphere Studio Application Developer v 5 (WSAD).

Пусть первое приложение состоит из одного независимого Java-класса, который всего лишь осуществляет вывод на системную консоль определенного текста.

Загрузим WSAD. При первом запуске продукта появляется окно, в котором разработчик может определить/изменить директорию для хранения файлов проекта. Эта директория называется workspace (рабочее пространство):

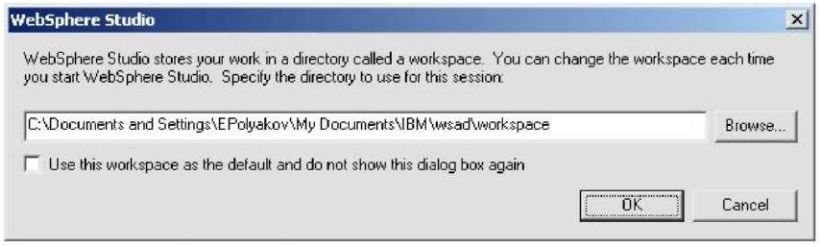

Рис. 1.1 Окно для определения workspace

После определения пути к workspace и клику по кнопке **Ok** через некоторое время загрузится основное окно WSAD:

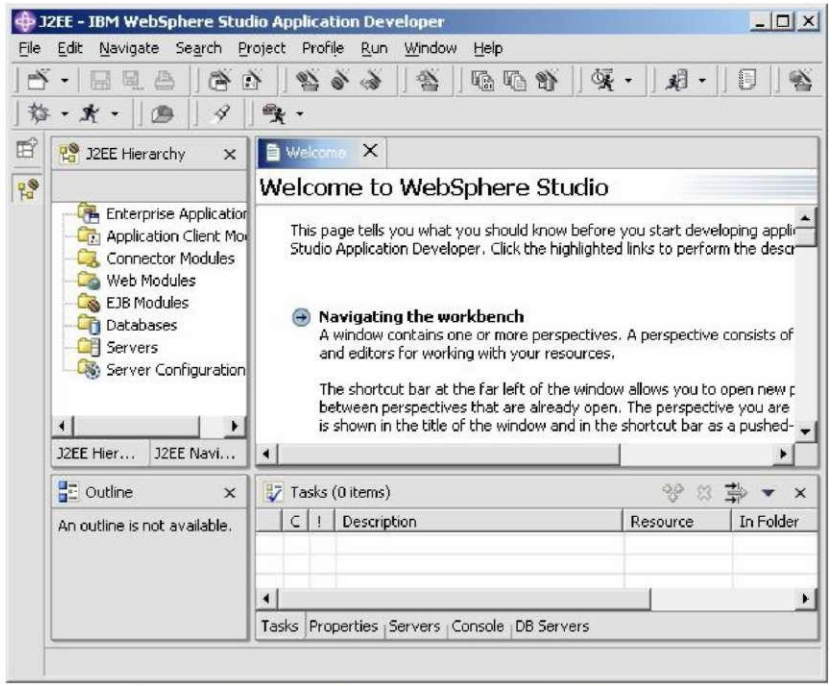

Рис. 1.2 Основное окно WebSphere Studio Application Developer

В рамках основного окна WSAD отображается ряд других окон. В терминологии WSAD эти окна делятся на два типа: перспективы (**perspective**) и редакторы (**editor**). Последние предназначены для редактирования файлов, составляющих проект и зависят от типа файла (для редактирования HTML-кода вызывается один редактор, для исходного кода Java – другой и т.д.). Перспективы, в свою очередь, состоят из видов (**view**), отображающих однотипную информацию. Каждая из перспектив предназначена для разработки определенных типов приложений (например, Web, Java, базы данных и т.д.). Для открытия перспективы можно воспользоваться меню **Window -> Open Perspective**, а для вида - **Window -> Show View**. В рамках настоящего урока мы не будем детально изучать предназначение всех указанных окон.

1.1 Установка кодовой страницы по умолчанию для ресурсов рабочего пространства

После установки WSAD на платформу Windows с русскими региональными настройками желательно проверить какую кодовую страницу по умолчанию будет использовать WSAD. Получить доступ к этому параметру можно через меню **Window -> Preferences**.

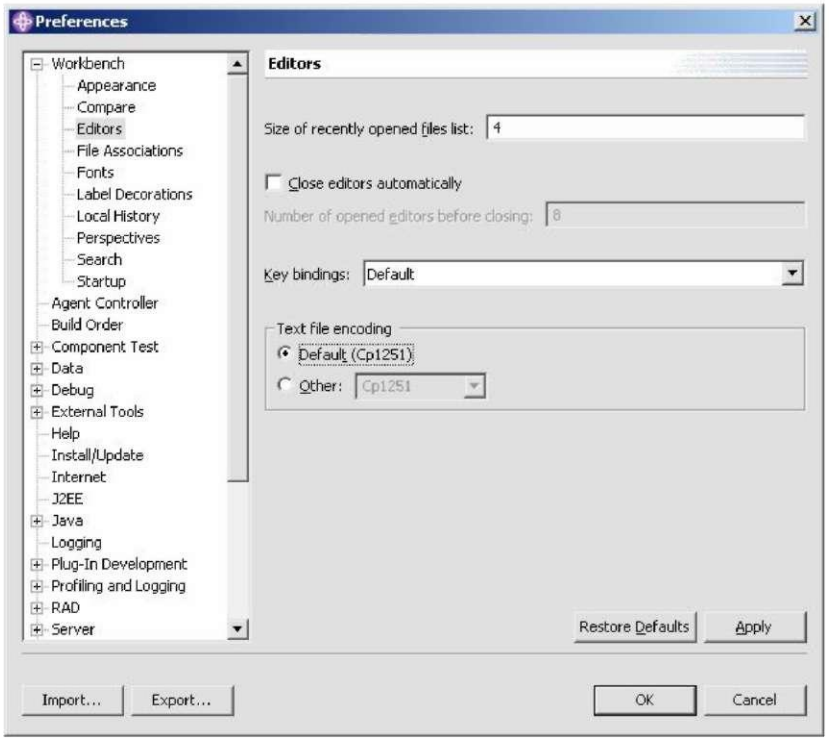

Рис. 1.3 Окно Window -> Preferences

В левом навигационном поле перейдем на пункт **Workbench -> Editors**. При этом в правой части окна рассмотрим секцию **Text file encoding** (кодовая страница для текстовых файлов). Обычно в этой секции для русской региональной настройки выбрано значение **Default (Cp1251)**, что не всегда приводит к корректной работе Web-проекта. При выборе радиокнопки **Other** (иное) в поле-списке, расположенном справа, представлены другие доступные кодировки. Однако требуемой **«windows-1251»** в списке нет. Решение проблемы состоит во вводе этого значения в поле непосредственно с клавиатуры. После этого закрываем окно по кнопке **Ok**.

1.2 Создание нового проекта

Для создания нового проекта можно воспользоваться меню **File -> New -> Project**. При этом вызывается мастер построения нового проекта:

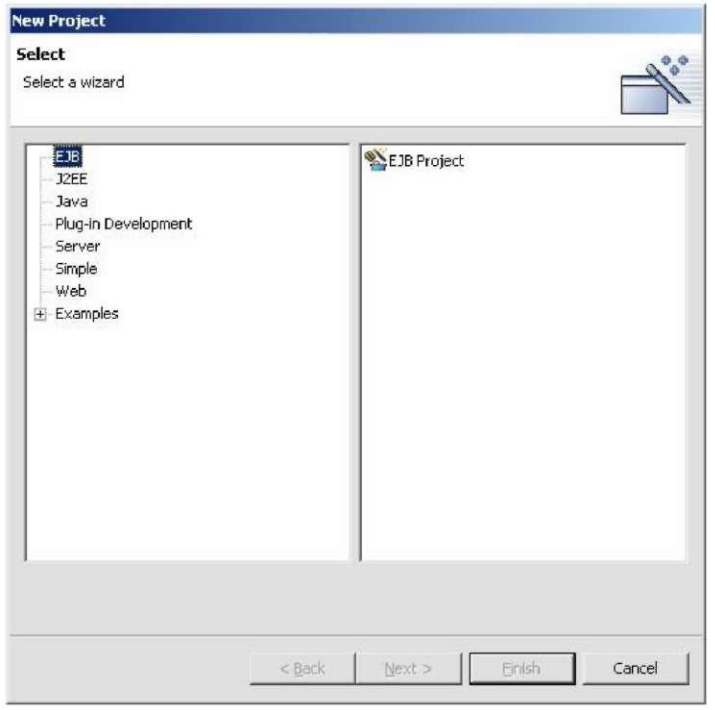

Рис. 1.4 Первое окно мастера построения проектов

Для нашей задачи мы выбираем в левом поле тип проекта как **Java**, а в правом поле – **Java Project**, и кликаем по кнопке **Next**. Второе окно мастера Java-проектов выглядит следующим образом:

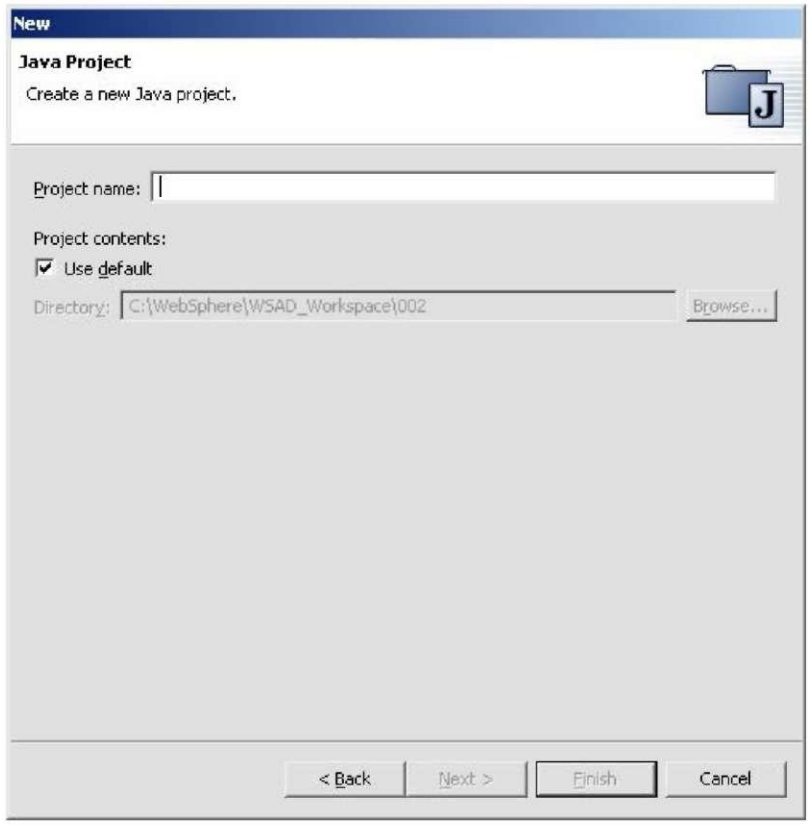

Рис. 1.5 Второе окно мастера построения Java-проектов

В этом окне введем значение для поля **Project name** (имя проекта, например, First) и кликаем по кнопке **Finish** (окончание). Если после выполнения указанных действий зайти в папку рабочего пространства средствами операционной системы (например, для Win-2000 это может быть проводник), то обнаружим в папке рабочего пространства новую папку с именем проекта (в нашем случае First).

Теперь проект создан, и необходимо приступить к наполнению его содержимым.

1.3 Создание Java-класса

После создания нового проекта откроем вид **Navigator** (навигатор). Это можно сделать с помощью меню **Window -> Show View -> Navigator**. В этом виде появилось имя нового проекта и его составляющие в иерархическом порядке (в принципе навигацию по нашему проекту можно было осуществлять и по виду **J2EE Navigator**).

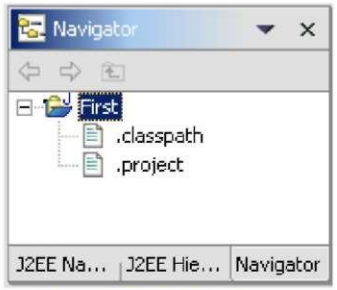

Рис. 1.6 Вид Navigator

Для этого в виде **Navigator** встанем на папку ***First*** нашего проекта и воспользуемся правой кнопкой мыши. В появившемся контекстном меню выберем пункт: **New -> Other**. Данное действие вызывает мастера составляющих проекта (аналогичного результата можно было достичь и через главное меню **File -> New -> Other**).

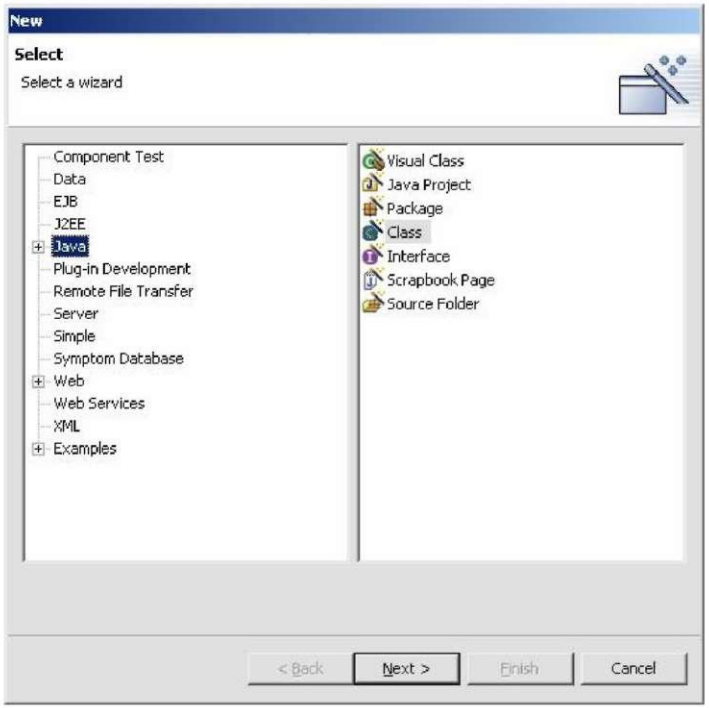

Рис. 1.7 Первое окно мастера составляющих проекта

В левом поле мастера выбираем значение **Java**, а в правом – **Class**, нажимаем кнопку **Next**.

Рис. 1.8 Второе окно мастера создания нового Java-класса

В появившемся окне определяем в поле **Name** имя создаваемого класса (например, Example), поднимаем флаг **public static void main(String[] args)** и нажимаем кнопку **Finish**.

При этом в редакторе откроется шаблон создаваемого нами Java-класса. Код которого выглядит следующим образом:

```
/**
 * @author epolyakov
```

```
 *
 * To change this generated comment edit the template variable
"typecomment":
 * Window>Preferences>Java>Templates.
 * To enable and disable the creation of type comments go to
 * Window>Preferences>Java>Code Generation.
 */
public class Example {


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

В рамках этого шаблона нам надо определить тело метода main в виде двух строк кода:

```
System.out.print("Первая Java-программа\n");
System.out.println("Hello World");
```

Обратите внимание, что после ввода кода в заголовке окна текущего редактируемого файла перед его именем появился символ «*». Это говорит о том, что файл изменен, но еще не сохранен. Для сохранения изменений можно воспользоваться либо меню **File -> Save**, либо комбинацией клавиш **CItr+S**.

Если при вводе не было допущено ошибок, то в папке First рабочего пространства будут присутствовать файлы Example.java и Example.class. Первый содержит исходный код нашей программы, второй - ее Java-байт код.

Теперь можно приступить к запуску приложения.

1.4 Запуск на выполнение Java-класса

Для запуска созданного Java-класса воспользуемся меню **Run -> Run**. При этом обычно появляется следующее окно:

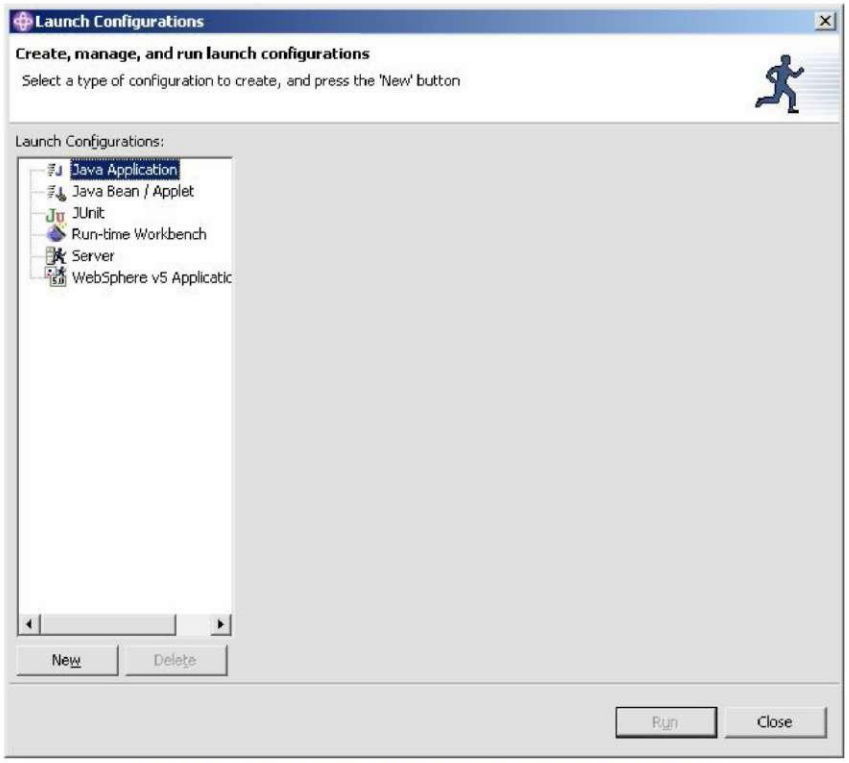

Рис. 1.9 Первое окно мастера запуска приложений

В этом окне в правом поле выберем значение **Java Application** и нажимаем кнопку **New**. Это вызывает появление второго окна мастера запуска приложений:

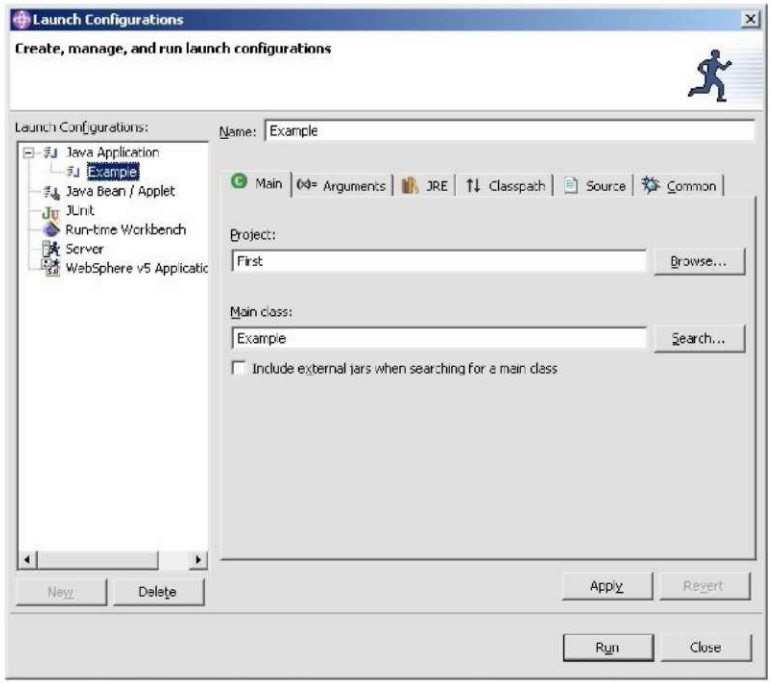

Рис. 1.10 Второе окно мастера запуска приложений

В этом окне проверяем значения полей **Project** (в нашем примере First) и **Main class** (Example). Остальные параметры оставляем без изменения, и нажимаем кнопку **Run**. При этом загружается вид **Console**, в котором отображается вывод системной консоли. Иногда требуется повторно выполнить команду **Run -> Run**.

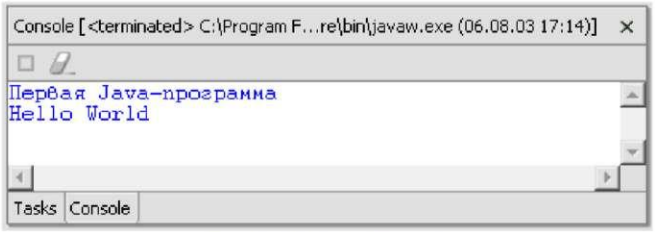

Рис. 1.11 Вид Console

1.5 Упражнение

В рамках этого упражнения слушатели должны самостоятельно выполнить все действия из данного урока, а именно: изменить кодовую страницу по умолчанию, создать новый Java-проект, создать Java-класс и запустить его на выполнение.

Следующие шаги используются для выполнения данного упражнения:

Шаг	Действие
1	Загрузить WSAD, определив местоположение рабочего пространства.
2	Установить кодовую страницу по умолчанию для рабочего пространства. Вызвать меню **Window -> Preferences**. В диалоговом окне перейти на пункт **Workbench -> Editors**, установить в поле напротив радиокнопки **Other** значение **windows-1251** и кликнуть по **Ok**.
3	Создать новый Java-проект с помощью меню **File -> New –> Project**.
4	В первом окне мастера проектов выбрать значение **Java**, а в правом поле – **Java Project**, и кликнуть по кнопке **Next**.
5	Во втором окне мастера проектов введем значение для поля **Project name** значение ***First*** и кликнем по кнопке **Finish**.
6	Создать новый Java-класс. Для этого в виде **Navigator** встаньте на папку ***First***, вызовите правой кнопкой мыши контекстное меню и выберите в нем пункт **New -> Class**.
7	В первом окне мастера построения Java-классов

	определить для поля **Name** значение ***Example***, поднять флаг **public static void main(String[] args)** и нажать кнопку **Finish**.

8	В тело метода main ввести следующий код:
	`System.out.print("Первая Java-программа\n");`
	`System.out.println("Hello World");`
	Сохранить текст Java-класса с помощью **Ctrl+S**.

9	Запустить приложение на выполнение с помощью меню **Run -> Run**.
10	В первом окне мастера запуска приложение в правом поле выбрать значение **Java Application** и нажать кнопку **New**.
11	Во втором окне мастера приложений нажать кнопку **Run**.
12	Проверить, что выводится в виде **Console**. В случае необходимости повторить шаги 9-11.

2 Урок 2. Краткий обзор синтаксиса и базовых конструкций языка Java

Данный урок посвящен первоначальному знакомству с основами языка Java.

После завершения урока слушатели должны:

- иметь представления о синтаксисе Java;
- знать о типах данных допустимых в Java;
- иметь представления о встроенных операциях языка;
- иметь представления об управляющих операторах языка.

Материал данной и ряда последующих глав никоим образам не претендует на исчерпывающий источник знаний по языку Java 2. Для этих целей существует достаточно много других весьма хороших изданий (например, [**Ошибка! Источник ссылки не найден.**]). Цель настоящих глав - дать очень сжатое представление о Java 2, заостряя внимание на механизмах реализации возможностей Java 2 в рамках WSAD.

Java – объектно-ориентированный мультиплатформенный многопоточный язык программирования. Язык Java был разработан компанией Sun и стал доступен для сторонних разработчиков в 1995 году. К настоящему времени язык претерпел две серьезные модификации (соответственно Java 1.1 и Java 2) и продолжает дальше успешно развиваться.

Java является по сути интерпретируемым языком. Исходный код Java-приложения компилируется в Java байт-код, который в свою очередь может быть выполнен в рамках виртуальной Java-машины (Java Virtual Machine, JVM). В настоящее время существуют компиляторы, выполняющие преобразование Java байт-кода в «родной» код операционной системы. Такие компиляторы называются JIT (Just In Time). Следует отметить, что они выполняют эти преобразования «на лету», т.е. это происходит в момент

выполнения приложения, и компиляции подвергается не весь бай-код, а только «ближайшая» его часть по отношении к выполнению.

На имя файла с исходным кодом Java-приложения накладывается ряд ограничений. Первое - расширение у такого файла должно быть *.java. Второе – само имя файла должно совпадать с именем класса из исходного кода. Откомпилированный байт-код имеет такое же имя и расширение *.class. Здесь следует не забывать, что язык Java чувствителен к регистру. Если вернуться к упражнению из первого урока, то, зайдя проводником в папку проекта (First) из рабочего пространства WSAD, мы обнаружим файлы Example.java и Example.class.

2.1 Синтаксис Java

При вводе исходного кода Java-приложения не обязательно придерживаться каких-либо строгих правил структурирования текста программы, т.к. Java является языком программирования свободного формата (free form language). Иными словами, можно разместить весь исходный код из файла Example.java на одной строке. С другой стороны, между элементами программы может находиться любое число символов пробела, табуляции или перевода на новую строку.

Операторы программы отделяются друг от друга с помощью символа точка с запятой «;».

2.1.1 Комментарии

В Java допустимы два типа комментариев:

- многострочные – весь текст, заключенный в ограничители «/*» и «*/», рассматривается компилятором как комментарий;

- однострочные – текст от символов «//» до конца строки так же считается комментарием.

Существует еще один тип комментариев, который используется для автоматического построения документации на Java-классы. Такие комментарии могут быть многострочными и заключаются в ограничители «/**» и «*/».

2.1.2 Блочный оператор

В Java допускается заключение нескольких операторов программы в символы фигурных скобок («{» и «}»), после этого данный фрагмент программы может рассматриваться как один общий оператор. Такие операторы очень часто используются там, где по синтаксису требуется указание одного оператора, а по смыслу должна идти последовательность действий. Блочные операторы могут быть вложенными

Пример. Блочные операторы.

```
for(i = 1; i <= 100; i++)
{ //начало первого блочного оператора
  if (i % 2 == 0)
  {//начало второго блочного оператора
        j = j + i;
        k = k + 1;
  }//окончание второго блочного оператора
}//окончание первого блочного оператора
```

2.1.3 Зарезервированные ключевые слова

Следующая таблица содержит список зарезервированных в Java 2 ключевых слов:

abstract	else	interface	strictfp
boolean	extends	long	super
break	false	native	switch
byte	final	new	synchronized
case	finally	null	this
catch	float	package	throw
char	for	private	throws
class	goto	protected	transient
const	if	public	true
continue	implements	return	try
default	import	short	void
do	instanceof	static	volatile
double	int		while

2.1.4 Идентификаторы

Идентификаторы предназначены для задания имен классам, методам и переменным. В качестве алфавита для идентификатора могут быть использованы буквы обеих регистров, цифры, символ подчеркивания и знак доллара. Идентификатор не должен начинаться с цифры. Еще раз напомним, что Java чувствителен к регистру.

2.2 Типы данных

Java – язык с жесткой типизацией. Как следствие этого в Java нельзя использовать переменные до их объявления (в отличие, например, от JavaScript).

2.2.1 Скалярные типы данных

Следующая таблица дает представление о допустимых в Java скалярных типах данных:

Тип	Разрядность	Диапазон	Пример
Целые			
byte	8	[-128; 127]	17
short	16	[-32`768; 32`767]	29123
int	32	[-2`147`483`648; 2`147`483`647]	-35555
long	64	[-9`223`372`036`854`775`808; 9`223`372`036`854`775`807]	111111111111
Вещественные			
float	32	от 3.4e-38 до 3.4e+38 в положительном и отрицательном диапазоне	1.2
double	64	от 1.7e-308 до 1.7e+308 в положительном и отрицательном диапазоне	3.62e-50
Булевский			
boolean		true и false	false
Символьный			
char	16	Символы Unicode	'Ы'