Лабораторная 1. Введение в технологии Java.

Цель:

- установка среды и настройка переменных окружения для выполнения Java-программ;
- создание первой программы;
- компиляция, исправление ошибок, выполнение программы;
- работа в интегрированных средах разработки: NetBeans и Eclipse;

Упражнение 1. Установка среды Java

Чтобы начать программирование на Java, необходимо загрузить специальное программное обеспечение с веб-сайта компании Oracle.

Комплект последних версий SDK можно свободно загружать с сайта http://www.oracle.com/technetwork/java/index.html или по старому адресу http://java.sun.com/. При занятии в классе требуемый дистрибутив можно скопировать из сетевой папки.

1. Скопируйте на локальную машину Java Development Kit из сетевой папки.
2. Установите JDK. Для этого запустите исполняемый файл

jdk-7u3-windows-i586-p.exe и следуйте инструкциям по инсталляции для вашей платформы.

Процедура инсталляции предлагает по умолчанию имя каталога для установки JDK. В имя каталога входит номер версии, это упрощает работу с новыми версиями JDK, установленными для проверки.

3. *Замечание.* Если вы работаете в системе Windows, Хорстманн [1] рекомендует не принимать предлагаемый каталог (обычно c:\Program Files\jdk7.0), а разместить пакет в другой позиции файловой системы. Рекомендуют также при инсталляции указывать для размещения корневой каталог.
4. Настройте переменные окружения JAVA_HOME, PATH и CLASSPATH.
5. Установка переменной окружения PATH позволяет запустить исполняемый файл без перехода в каталог, содержащий установленную JAVA. При вводе любой исполняемой команды, запускающей утилиты (javac, java и т.д.), не обнаружив файла в текущем каталоге, система производит его поиск в каталогах, указанных в переменной PATH.

Переменная CLASSPATH выполняет сходные функции для файлов байт-кода Java: позволяет исполняющей системе находить и запускать Java-программы из различных каталогов (по умолчанию – только из каталога с исполняемыми файлами JDK). CLASSPATH указывает, какие директории в файловой системе являются корневыми для иерархии пакета Java.

Переменная JAVA_HOME используется многими приложениями Java для определения расположения Java SDK в файловой системе.

Для создания и настройки переменных окружения для работы с Java в ОС Windows 7 выполните следующие действия:

- нажмите кнопку *Пуск*, далее щелкните правой кнопкой мыши по пункту *Компьютер,* выберите пункт *Свойства* и далее *Дополнительные параметры системы*.
- перейдете на вкладку *Дополнительно* и нажмите на кнопку *Переменные среды...* Откроется окно **Переменные среды**.
- для создания переменной среды пользователя щелкните на кнопке *Создать*, откроется окно **Новая пользовательская переменная.**
- в поле *Имя переменной* введите имя переменной JAVA_HOME, а в поле *Значение переменной* – путь к Sun Java SDK, например C:\Program Files\Java\jdk1.7.0_03. Нажмите OK, переменная должна появиться в списке переменных среды.
- создайте аналогичным образом переменную CLASSPATH. В поле *Значение переменной* поставьте символ «.» (точка обозначает текущий каталог), затем разделитель «;» и %JAVA_HOME%\lib\tools.jar (путь к библиотекам Java JDK). Первое значение нужно для доступа к коду ваших Java-приложений, второе – для поиска библиотек JDK.
- в окне Пользовательские переменные создайте переменную PATH. Выделите ее и нажмите кнопку *Создать*.
- в поле *Значение переменной* добавьте разделитель “;” и далее %JAVA_HOME%\bin (подкаталог bin каталога Java SDK). Нажмите OK.

6. Проверьте правильность настроек.
   - откройте командную строку и наберите команду set. ОС выдаст список установленных переменных окружения с их значениями. В списке должны присутствовать переменные *JAVA_HOME*, *PATH* и *CLASSPATH*.
   - наберите команду java –version. Должно появиться сообщение о версии Java Runtime Environment (JRE). (если сообщения нет, то попробуйте прописать в переменных *PATH* и *CLASSPATH* непосредственно пути к корневому каталогу и библиотекам).

7. Откройте страницу http://docs.oracle.com/javase/7/docs/api/ и изучите справочную систему JDK. По остальным версиям спецификацию можно найти по адресу: http://www.oracle.com/technetwork/java/api-141528.html.

Упражнение 2. Создание первой программы

Для создания программы на Java необходимо:

1. Написать программу на Java и сохранить ее на диск.
2. Выполнить компиляцию программы, чтобы перевести ее с языка Java в специальный байт-код, который понимает виртуальная машина JVM.
3. Запустить программу.

Пакет JDK не имеет никаких средств, даже отдаленно напоминающих интегрированную среду разработки программ. Все команды выполняются в командной строке.

Интегрированные среды разработки программ не совсем удобны для написания простых программ, поскольку они работают медленно, потребляют большой объем ресурсов. Интегрированные среды полезны, если разрабатывается большой проект, состоящий из многих файлов. Кроме того, в состав этих сред входит отладчик, крайне необходимый при разработке серьезных программ (отладчик, работающий в режиме командной строки, поставляемый в составе JDK, крайне неудобно использовать).

В следующих упражнениях будет изучаться работа с программами NetBeans и Eclipse, представляющие собой свободно распространяемые среды разработки.

В этом упражнении описывается вызов компилятора и запуск программы на выполнение из командной строки.

Выполните следующие действия для выполнения программы из командной строки:

1. Откройте текстовый редактор, например, Блокнот (Notepad) и введите следующий код:

```
public class TestGreeting {
        public static void main (String[] args) {
                Greeting hello = new Greeting();
                hello.greet();
        }
}
```

2. Сохраните файл, указав имя в кавычках "TestGreeting.java". Текстовый редактор Notepad сохраняет текст в файлах с расширением .txt. По умолчанию в среде Windows программа Explorer скрывает расширение .txt, поскольку оно предполагается по умолчанию и поэтому нужно сохранить файл, указав имя в кавычках. Весь код должен находиться внутри класса и имя этого класса должно быть согласовано с именем файла, который содержит программу.
3. Создайте аналогичным образом еще файл с именем Greeting.java, содержащий следующий класс:

```
public class Greeting {
        public void greet() {
                System.out.println("Hello");
        }
}
```

4. Откомпилируйте программу. Откройте окно командной строки, перейдите в каталог с исходным файлом и введите:

```
javac TestGreeting.java
```

При отсутствии ошибок javac создаст два файла, которые содержат программы в виде байт-кода, с именем TestGreeting.class, и автоматически (так как класс TestGreeting использует класс Greeting) файл Greeting.class.

После компиляции исходного кода каждый класс с модификатором public помещается в собственный выходной файл, имя которого совпадает с именем этого класса, а этот файл имеет расширениее .class.

Байт-код Java — это промежуточное представление программы, состоящее из инструкций, которые будет выполнять интерпретатор Java. Когда запускается интерпретатор Java, в действительности специфицируется имя класса, который должен исполнить интерпретатор. Он автоматически отыскивает файл с тем же именем и расширением .class. Если интерпретатор находит такой файл, то он выполняет код, содержащийся в указанном классе.

Таким образом, результат работы компилятора javac не является непосредственно выполняемым кодом.

5. Для выполнения программы следует использовать Java-интерпретатор с именем java. В окне командной строки введите следующее:

java TestGreeting

После выполнения программы на экран должна быть выведена строка приветствия.

Для решения задачи приветствия используются два класса. Оба класса объявлены с модификатором доступа public. Предполагается, что какие-то другие классы тоже захотят поздороваться. В Java публичный класс должен быть реализован в своем файле.

6. Создайте программу, реализовав оба класса в одном файле, при этом класс Greeting должен быть объявлен без модификатора public.

Для удобства работы можно порекомендовать текстовый редактор TextPad, который соответствует стандартам системы Windows. Информацию об этом редакторе и саму программу можно найти на странице http://www.textpad.com. Редактор TextPad относится к условно бесплатным программам.

Также можно использовать JEdit — свободно распространяемую программу, написанную на Java: http://www.jedit.org.

Независимо от того, какой редактор используется, основные принципы работы остаются неизменными. Закончив редактировать исходный код, можно вызывать компилятор. Редактор корректно запустит программу компиляции и предоставит вам сообщения об ошибках. Устранив ошибки, повторите компиляцию, после чего вы сможете запустить программу на выполнение.

Упражнение 3. Создание в NetBeans простейшего приложения Java

Для создания приложения Java с помощью среды NetBeans выполните следующие действия:

1. Выберите в главном меню **File (Файл) → New Project…(Создать проект**…).
2. В открывшемся окне диалога (см. рис. 1) в категории **Java** выберите тип проекта **Java Application (Приложение Java) → Next (Далее).**

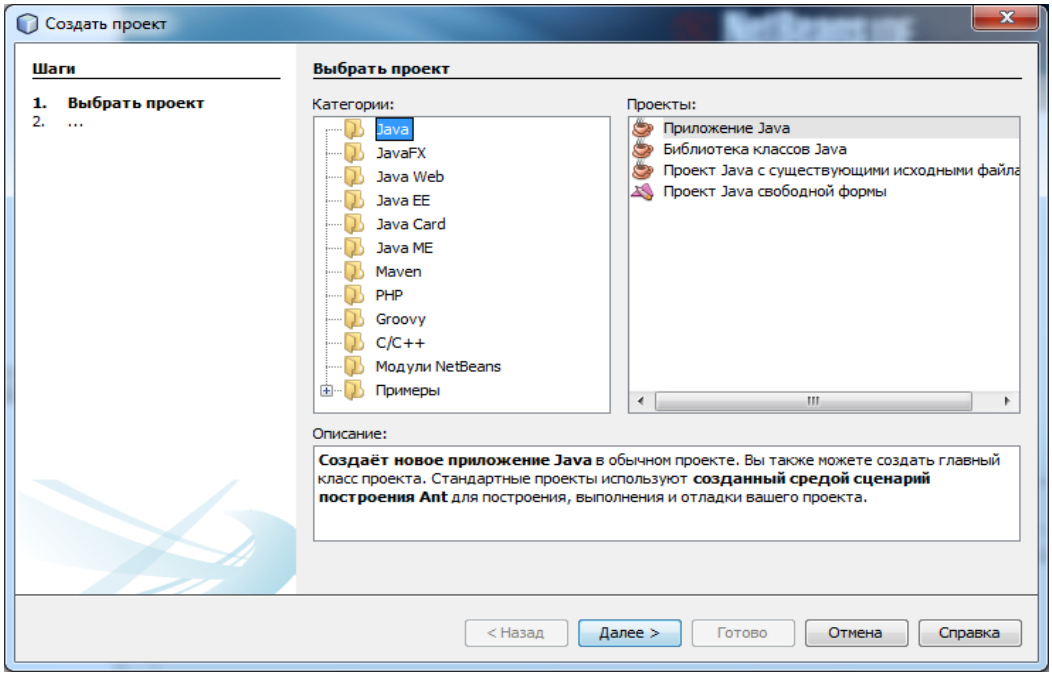

Рис. 3.1 Создание проекта

3. В следующем окне (см. рис. 2) определите имя и расположение проекта. Оставьте включенными опции “Создать класс” и “Сделать главным проектом”. Нажмите кнопку Finish (Готово).

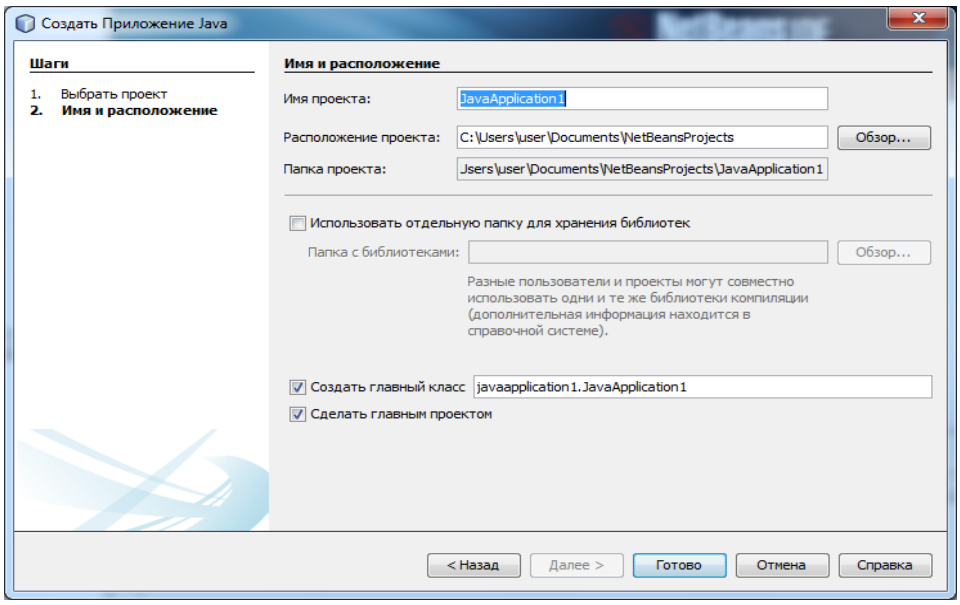

Рис.2 Создание приложения

4. Изучите среду разработки NetBeans. На рис. 3 показано, как выглядит приложение в данной среде разработки.

В левом верхнем окне "Projects" (Проекты) показывается дерево проектов. Это окно может быть использовано для одновременного показа произвольного числа проектов. По умолчанию все деревья свернуты, нужные узлы следует разворачивать щелчком по узлу, отмеченным "плюсом" или двойным щелчком по соответствующему имени.

В правом окне "Source" (Исходный файл) показывается исходный код проекта.

В левом нижнем окне "Navigator" (Просмотр элементов) показывается список имен членов класса приложения (имена переменных и подпрограмм). Двойной щелчок по имени приводит к тому, что в окне редактора исходного кода происходит переход на то место, где задана соответствующая переменная или подпрограмма.

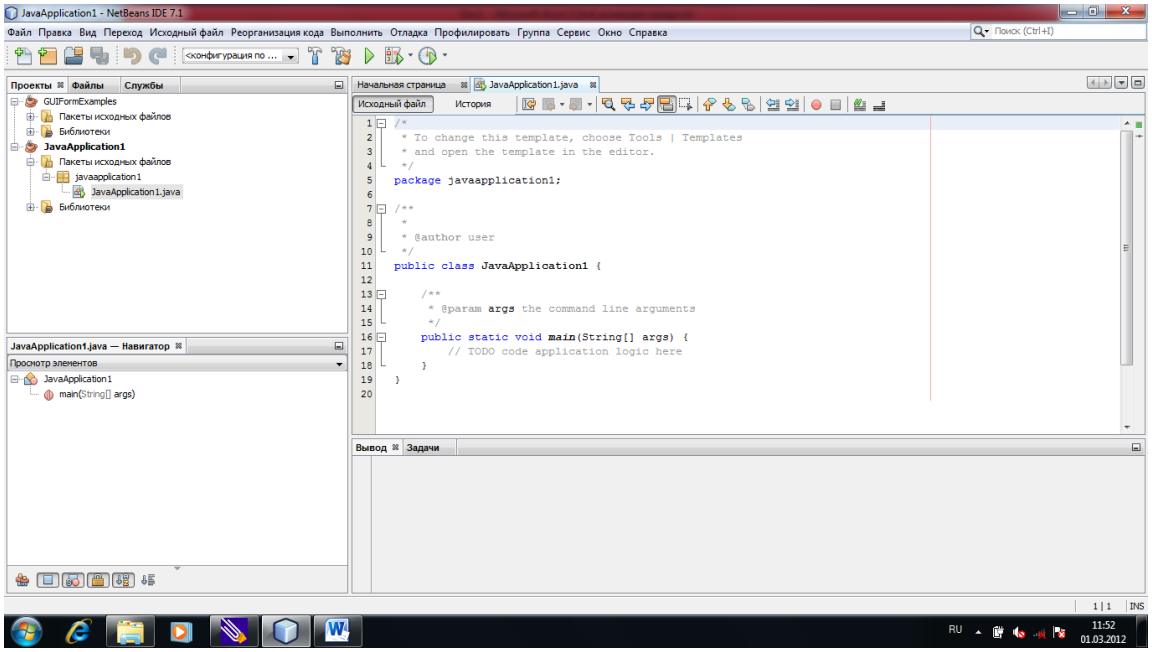

Рис.3 Редактирование исходного кода приложения

5. Рассмотрите сгенерированный исходный код приложения Java:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package javaapplication1;
/**
 *
 * @author user
 */
public class JavaApplication1 {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
```

```
    }
}
```

Сначала идет многострочный комментарий /* … */. Он содержит служебную информацию.

Затем объявляется, что класс будет находиться в пакете javaapplication1, который является на самом деле каталогом на диске. Для приложения, состоящего из двух классов, наличие пакетов не является необходимостью. При отсутствии слова package классы будут отнесены к пакету по умолчанию, размещенному в корне проекта. Если же приложение состоит из нескольких сотен классов (вполне обычная ситуация), то размещение классов по пакетам является жизненной необходимостью.

После этого идет многострочный комментарий /** … */, предназначенный для автоматического создания документации. В нем присутствует инструкция задания метаданных с помощью выражения @author – информация об авторе проекта для утилиты создания документации javadoc. Метаданные – это некая информация, которая не относится к работе программы и не включается в нее при компиляции, но сопровождает программу и может быть использована другими программами для проверки прав на доступ к ней или ее распространения, проверки совместимости с другими программами, указания параметров для запуска класса и т.п. В данном месте исходного кода имя "User" берется средой разработки из операционной системы по имени папки пользователя. Его следует заменить именем реального автора программного кода.

Далее следует объявление класса JavaApplication1, который является главным классом приложения. В нем объявлен метод main().

Все классы и объекты приложения вызываются и управляются из метода main(), который выглядит следующим образом:

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

Он является методом класса (на это указывает модификатор static), и поэтому для его работы нет необходимости в создании объекта, являющегося экземпляром класса. Хотя если этот объект создается, это происходит во время работы метода main().

Метод main является главным методом приложения и управляет работой запускаемой программы. Он автоматически вызывается при запуске приложения. Параметром args этого метода является массив строк, имеющий тип String[]. Это параметры командной строки, которые передаются в приложение при его запуске. Слово String означает "Строка", а квадратные скобки используются для указания, что это массив.

После окончания выполнения метода main приложение завершает свою работу.

При объявлении любого метода в Java сначала указывается модификатор видимости, указывающий права доступа к методу, затем другие модификаторы, после чего следует тип возвращаемого методом значения. Если модификатор видимости не указан, считается, что это private (закрытый, частный), не позволяющий доступ к методу из других классов.

Далее следует имя метода, после чего в круглых скобках идет список параметров (аргументов), передаваемых в данный метод при его вызове. После этого в фигурных скобках идет тело метода, то есть его реализация, пишется тот алгоритм, который будет выполняться при вызове метода.

Если надо написать подпрограмму-процедуру, в которой не надо возвращать никакого значения, то используются подпрограммы-функции с типом возвращаемого значения void – как и происходит в случае метода main.

Среда NetBeans создает заготовку методов, в них имеется пустое тело. Для осуществления методом какой-либо деятельности следует дописать свой собственный код.

6. После комментария
// TODO code application logic here ("описать логику работы приложения здесь") добавьте следующий код:

```
int a = 5;
int d = 7;
int sum = a + d;
System.out.println("Sum = " + sum);
```

Класс System, содержит объект out, предназначенный для поддержки вывода. У него есть метод println, предназначенный для вывода текста в режиме консоли.

Консольный ввод-вывод ранее широко применялся в операционных системах, ориентированных на работу в режиме командной строки. При этом основным средством взаимодействия пользователей с программами служила текстовая консоль. В ней устройством ввода служила клавиатура, а устройством вывода – окно операционной системы, обеспечивающее вывод текста в режиме пишущей машинки (системным шрифтом с буквами, имеющими одинаковую ширину).

В настоящее время в связи с тем, что подавляющее большинство пользователей работают с программами в графическом режиме, работу в консольном режиме нельзя рассматривать как основную форму ввода-вывода. Тем более что NetBeans позволяет создавать графический пользовательский интерфейс (GUI – Graphics User Interface) приложения.

Консольный режим следует применять только как промежуточный, удобный в отладочном режиме как средство вывода вспомогательной информации.

Упражнение 4. Компиляция файлов проекта и запуск приложения

В современных средах разработки используется два режима компиляции: compile ("скомпилировать") и build ("построить").

В режиме "compile" происходит компиляция только тех файлов проекта, которые были изменены в процессе редактирования после последней компиляции. А в режиме "build" перекомпилируются заново все файлы.

1. Скомпилируйте проект, выполнив следующие действия:

Build → Build Main Project (Выполнить →Построить главный проект)

(или клавиша <F11>, или на панели инструментов иконка в виде молотка с голубой ручкой). При этом будут заново скомпилированы из исходных кодов все классы проекта.

2. Изучите другие возможности компиляции приложения:

Пункт **Build → Clean and Build Main Project (Выполнить → Очистить и построить главный проект)** (или комбинация клавиш <Shift><F11>, или на панели инструментов иконка в виде молотка с голубой ручкой и веником) удаляет все выходные файлы проекта (очищает папки build и dist), после чего по новой компилируются все классы проекта.

Пункт **Build → Generate Javadoc (Выполнить → Создать документацию Java)** запускает создание документации по проекту. При этом из исходных кодов классов проекта выбирается информация, заключенная в документационные комментарии /** … */, и на ее основе создается гипертекстовый HTML-документ.

Пункт **Build → Compile (Выполнить → Компилировать файл)** (или клавиша <F9>) компилирует выбранный файл проекта, в котором хранятся исходные коды соответствующего класса.

3. Для запуска скомпилированного приложения из среды разработки, выберите в меню:

Run → Run Main Project (Выполнить → Запустить главный проект)

(или клавиша <F6>, или на панели инструментов иконка с зеленым треугольником).

При запуске приложение всегда автоматически компилируется, так что после внесения изменений для запуска обычно достаточно нажать <F6>.

4. Проверьте, что в выходной консоли, которая находится в нижней части окна проекта, выполнится вывод метода System.out.println, а также появится служебная информация о ходе компиляции и запуска.

5. Изучите структуру папок проекта **NetBeans**. Откройте в проводнике папку вашего проекта. По умолчанию головная папка проекта располагается в папке пользователя.

- В папке **build** хранятся скомпилированные файлы классов, имеющие расширение .class: build \ classes \ javaapplication \ JavaApplication.class.
- В папке **dist** - файлы, предназначенные для распространения как результат компиляции: модуль JAR приложения или библиотеки (JavaApplication.jar) , а также документация к нему (README.TXT).
- В папке **nbproject** находится служебная информация по проекту.

- В папке **src** - исходные коды классов. Кроме того, там же хранится информация об экранных формах (которые будут видны на экране в виде окон с кнопками, текстом и т.п.). Она содержится в XML-файлах, имеющих расширение .form.
- В папке **test** - сопроводительные тесты, предназначенные для проверки правильности работы классов проекта.

В компонентной модели NetBeans пакеты приложения объединяются в единую конструкцию – модуль.

Модули NetBeans являются базовой конструкцией не только для создания приложений, но и для написания библиотек. Они представляют собой оболочку над пакетами (а также могут включать в себя другие модули).

В отличие от библиотек Java скомпилированный модуль – это не набор большого количества файлов, а всего один файл, архив JAR (Java Archive, архив Java). В нашем случае он имеет то же имя, что и приложение, и расширение .jar : это файл JavaApplication1.jar.

Модули NetBeans гораздо лучше подходят для распространения, поскольку не только обеспечивают целостность комплекта взаимосвязанных файлов, но и хранят их в заархивированном виде в одном файле, что намного ускоряет копирование и уменьшает объем занимаемого места на носителях.

После сохранения проекта и закрытия среды разработки и открытия ее снова по умолчанию сначала открывается окно Welcome ("Начальная страница"). Среда разработки сохраняет список открытых окон, и в верхней части окна редактирования кода щелчком мыши можно выбрать нужное имя окна. Хотя при этом не видна структура проекта, так что первый способ во многих случаях может быть предпочтительным (однако, вы можете увидеть открытый файл в структуре проекта, если нажмёте ctrl+shift+1).

Если вы открываете новый проект, старый не закрывается. И в дереве проектов видны все открытые проекты. То же относится и к списку открытых окон. Это позволяет работать сразу с несколькими проектами, например – копировать в текущий проект участки кода из других проектов.

Один из открытых проектов является главным (Main Project), именно он будет запускаться на исполнение. Для того чтобы установить какой-либо из открытых проектов в качестве главного, следует в дереве проектов с помощью правой кнопки мыши щелкнуть по имени проекта и выбрать пункт меню Set Main Project (Установить как главный проект). Аналогично, для того, чтобы закрыть какой-либо из открытых проектов, следует в дереве проектов с помощью правой кнопки мыши щелкнуть по имени проекта и выбрать пункт меню Close Project (Закрыть).