Федеральное агентство по образованию

С.А. Казарин, А.П. Клишин

Среда разработки Java-приложений Eclipse

(ПО для объектно-ориентированного программирования и разработки приложений на языке Java)

Учебное пособие

Москва 2008

Казарин С.А., Клишин А.П.

К 143 Среда разработки Java-приложений Eclipse: (ПО для объектно-ориентированного программирования и разработки приложений на языке Java): Учебное пособие. Москва 2008. — 77 с.

Учебное пособие представляет собой руководство по использованию среды разработки Eclipse для создания приложений на языке Java. Основное внимание уделено практическим приемам работы с Eclipse.

Пособие предназначено для преподавателей, студентов, школьников старших классов, а также для специалистов, желающих освоить самостоятельно программирование на языке Java.

Оглавление

Предисловие.....4
Введение.....6
Глава 1.Среда разработки приложений Eclipse.....9
1.1.Системные требования.....9
1.2.Введение в Eclipse.....10
1.3.Установка Eclipse.....12
1.4.Первый запуск Eclipse.....14
1.5.Интерфейс пользователя.....18
1.6.Настройки среды.....25
1.7.Создание проекта.....27
1.8.Поддержка, советы, рекомендуемые ресурсы.....33
Глава 2.Отладка и тестирование приложений.....37
Глава 3.Лабораторный практикум.....44
Глоссарий.....71
Список литературы.....77

Предисловие

Подготовка пакета свободного программного обеспечения (ПСПО) для школ является важным этапом на пути решения поставленных временем задач перед российским образованием. Обучение школьников азам современного языка программирования и навыкам работы с системой разработки программных приложений СПО на ранних этапах позволит приобщить учащихся к общемировым достижениям в области информационных технологий и заметно поднять их образовательный и культурный уровень.

Концепция языка Java, как совокупности языка программирования и виртуальной машины, ведет свое происхождение от проекта фирмы Sun под названием Green (Oak), открытого в 1990 г. Первоначальной целью проекта было создать среду разработки программного обеспечения для бытовой электроники. Компания Sun поставила задачу заменить множество разнообразных архитектур микроконтроллеров одной единственной масштабируемой архитектурой (прототипом виртуальной Java-машины). Затем проект был переориентирован на World Wide Web и в августе 1995 г. состоялся первый официальный выпуск Java.

Своим стремительным распространением в широких кругах сообщества Java обязана необычайно удачному сочетанию следующих факторов: быстрого роста сети Internet и телекоммуникационных технологий, возрастанию потребностей образования в новых подходах к программированию, успехам в создании мобильных, портативных вычислительных устройств.

Отметим принципиальные нововведения, которые отличают Java от других: независимость небольших мобильных программ в сочетании с генерацией кода в процессе выполнения, переносимость, строгая типизация с поддержкой динамических типов и системы «сборки мусора».

Java предоставляет мощные объектно-ориентированные принципы разработки приложений, сочетая простой и ясный синтаксис с надежной и удобной в работе системой программирования, что позволяет быстро обучаться и создавать новые программы.

Как правило, современные средства создания Java-приложений разработаны для различных платформ: Linux, Solaris, Windows и MacOS. Важнейшее преимущество языка Java заключается в том, что приложение, написанное на основе данного языка, является независимым от платформы и архитектуры процессора, который будет выполнять алгоритм данного приложения. Главным звеном в данном процессе является виртуальная машина Java — это специальная программа, которая имеющийся откомпилированный независимый байт-код преобразует в

набор инструкций для конкретного процессора. Программа должна быть предварительно установлена на компьютер, где планируется запуск приложения.

Язык Java является объектно-ориентированным и поставляется с достаточно объемной библиотекой классов. Библиотеки классов Java значительно упрощают разработку приложений, представляя в распоряжение программиста мощные средства решения стандартных задач.

Почти сразу же после появления Java было создано большое количество интегрированных сред разработки программ для этого языка: Eclipse (Eclipse Foundation), NetBeans (Sun), JBuilder (Inprise), Visual Age (IBM), VisualCafe (Symantec) и др. Причем, что интересно: большинство из существующих инструментальных сред разработки приложений написаны полностью на Java и имеют развитые средства визуального программирования.

Создавая это пособие, авторы стремились изменить установившееся мнение о сложности языка Java, показать, что его с успехом может использовать каждый — от школьника старших классов до специалистов в области IT. Различные пользователи, решая задачи с применением современных средств разработки, в зависимости от квалификации и уровня компетенции могут применять разные подмножества языка. Используя среду Eclipse в качестве инструментария разработки приложений, постепенно углубляя свои знания, учащиеся будут постигать поистине безграничные возможности, предоставляемые современной вычислительной техникой и информационными технологиями.

Пакет Eclipse может быть рекомендован к использованию в старших классах общеобразовательной школы, в качестве материала факультативных занятий, элективных курсов, а также может быть применен в специальных курсах профильного обучения. Программирование на языке Java может использоваться как на уроках информатики, так и в ходе самостоятельной работы.

Пособие снабжено многочисленными примерами из школьного курса информатики, которые наглядно иллюстрируют различные возможности изучаемого объектно-ориентированного языка.

Для закрепления материала, представленного в пособии, в конце имеется практикум, где приведены задачи, многие из которых относятся к школьному курсу информатики. Кроме того, в конце каждой главы даны упражнения для самостоятельной работы.

Авторы с благодарностью примут все замечания и предложения от читателей на адрес электронной почты spo_method_support@armd.ru

Введение

Программы, созданные на языке программирования Java, подразделяются по своему назначению на две группы.

К первой группе относятся приложения Java, предназначенные для локальной работы под управлением интерпретатора (виртуальной машины) Java.

Вторую группу программ называют апплетами (aplets). Апплеты представляют собой небольшие специальные программы, находящиеся на удаленном компьютере в сети, с которым пользователи соединяются с помощью браузера. Апплеты загружаются в браузер пользователя и интерпретируются виртуальной машиной Java, встроенной практически во все современные браузеры.

Приложения, относящиеся к первой группе, представляют собой обычные локальные приложения. Поскольку они выполняются интерпретатором и не содержат машинного кода то их производительность заметно ниже, чем у обычных компилируемых программ (C++, Delphi).

Апплеты Java можно встраивать в документы HTML и помещать на Web-сервер. Использование в интернет-страницах Java-апплетов придает динамический и интерактивный характер поведению последних. Апплеты берут на себя сложную локальную обработку данных, полученных от Web-сервера или от локального пользователя. Для более быстрого выполнения апплетов в браузере применяется особый способ компиляции — Just-In-Time compilation (JIT, «на-лету)», что позволяет увеличить скорость выполнения апплета в несколько раз.

Для разработки программ на языке Java нам потребуется специальное программное обеспечение. Самые новые версии системного программного обеспечения, необходимого для поддержки, можно загрузить с сайта компании Sun (http://java.sun.com/): JRE, JDK. Первое приложение JRE — это программа для запуска и исполнения программ (среда выполнения Java) Java Runtime Environment (JRE).

Для разработки программ также требуется комплект разработки программного обеспечения — JDK (Java Development Kit). Он содержит компилятор, стандартные библиотеки и т.п.

В настоящее время имеется три Java-платформы:

1) Java 2 Platform, Standard Edition (J2SE);

2) Java 2 Platform, Enterprise Edition (J2EE);

3) Java 2 Platform, Micro Edition (J2ME).

Каждая из этих платформ предназначена для разработки определенного типа программ.

Первая платформа J2SE позволяет разрабатывать обычные (desktop) локальные приложения и апплеты.

Вторая платформа J2EE предназначена для разработки серверных приложений (сервлетов, jsp-страниц, компонентов JavaBeans).

Третья платформа (J2ME) применяется при разработки приложений для мобильных и небольших устройств (сотовых телефонов, карманных компьютеров и др.), которые имеют существенно ограниченные аппаратные ресурсы (емкость оперативной памяти, быстродействие процессора и др.).

Таким образом, в минимальный комплект для разработки программ на Java входят следующие:

- JRE — среда выполнения;
- JDK для соответствующей платформы (J2SE, J2EE, J2ME) — компилятор и библиотеки;
- среда программирования.

Системы программирования на Java состоят из нескольких частей: среда разработки, язык программирования, программный интерфейс приложений (Java API), различные библиотеки классов. В первой главе подробно рассматривается одна из лучших сред разработки программ Java Eclipse.

Программы Java обычно проходят пять стадий обработки, прежде чем будут выполнены: редактирование, компиляция, загрузка, проверка байт-кода и выполнение.

На первом этапе в редакторе Eclipse вводится программа, а затем исправления в случае необходимости. Файл с программой необходимо сохранить, после окончательного редактирования он имеет стандартное расширение «имя».java. С таким же успехом можно использовать распространенные текстовые редакторы vi, kate, kwrite и emacs. В Eclipse содержится неплохой встроенный редактор и среда программирования, поэтому мы рекомендуем все действия проводить там.

На следующем втором этапе вы должны откомпилировать программу, что достаточно просто сделать, выбрав команду Run главного меню. Компилятор Java должен выполнить трансляцию программы Java в байт-код, в этой форме она уже будет доступна интерпретатору Java. Если ваша программа успешно откомпилируется, то будет создан файл с именем «имя».class. В данном файле содержатся байт-коды, которые будут интерпретироваться во время выполнения.

Третий этап называется загрузкой. Программа помещается в оперативную память и ей передается управление. Загрузчик классов в Eclipse считывает файл «имя».class и помещает его в оперативную па-

мять. Файл может загружаться как с локального диска компьютера, так с удаленного компьютера по сети. Файл «имя».class может содержать программы двух видов: обычные локальные приложения и апплеты, о которых мы говорили в начале введения. Загрузчик загружает в ОП наш файл и затем программа начинает выполняться интерпретатором Java. Среда Eclipse сама позаботится и вызовет интерпретатор для выполнения приложения.

Загрузчик классов также может вызываться и в том случае, когда ваш интернет-браузер загружает интернет- страницу с встроенным Java-апплетом. Интернет-страница в формате HTML может ссылаться на Java-апплет. Когда браузер загружает такую страницу и начинает ее интерпретировать, то в момент ссылки на апплет он вызывает загрузчик классов и загружает этот апплет. Практически все современные интернет-браузеры поддерживают Java, т.е. имеют встроенный интерпретатор языка Java.

Перед тем как интерпретатор Java, вызываемый в Eclipse или встроенный в браузер, приступит к выполнению байт-кода, последний проверяется верификатором байт-кода на четвертом этапе выполнения программы. Этот этап назовем проверкой байт-кода. Успешное прохождение данного этапа гарантирует нам то, что загруженные классы не нанесут ущерб защите и не содержат ошибок, которые могли бы вызвать сбой работы программы.

На последнем пятом этапе Eclipse интерпретирует программу, последовательно выполняя байт-коды. Программа может сразу не заработать в результате ошибок, вызванных на разных этапах выполнения, тогда вам следует вернуться к исходному тексту программы и внести необходимые исправления. Старайтесь писать программы на Java в простом и ясном стиле. Следует избегать неправильных конструкций и способов употребления языка, тогда ваши программы будут быстро проходить все этапы выполнения.

В настоящее время отмечается большой прогресс в развитии свободно распространяемого программного обеспечения для разработки программ на языке Java. По своему функциональному составу и набору предоставляемых сервисных услуг Eclipse практически не уступает платным программам, поэтому изучение данной среды является перспективным и отвечающим большинству требований, выдвигаемых к современному программному обеспечению, функционирующему на различных аппаратных платформах.

Глава 1. Среда разработки приложений Eclipse

Eclipse — один из лучших инструментов Java, созданных за последние годы. SDK Eclipse представляет собой интегрированную среду разработки (IDE, Integrated Development Environment) с открытым исходным кодом.

В начале своего существования Eclipse появилась как коммерческий продукт, но в ноябре 2001 г. его исходные коды были опубликованы. Создателем системы является компания Object Technology International (OTI), которая впоследствии была приобретена корпорацией IBM. Начиная с 2001 г. Eclipse была загружена более 50 миллионов раз и в настоящее время используется десятками тысяч программистов по всему миру. Поддержкой и разработкой Eclipse в настоящее время занимается организация Eclipse Foundation и сообщество Eclipse, информацию о которых можно найти на официальном сайте в сети Интернет http://www.eclipse.org.

Основные инструментальные средства Eclipse Java включают в себя: редактор исходного кода (создание и редактирование исходного текста программ), средства отладки и интеграции с Ant. Кроме этого в Eclipse доступны множество бесплатных и коммерческих дополнений (плагинов), таких, как инструментальные средства создания схем UML, разработка баз данных и др.

Собственно сама по себе Eclipse — это только платформа, которая предоставляет возможность разрабатывать дополнения, называемые плагинами, которые естественным образом встраиваются в платформу. В Eclipse доступны дополнения для следующих языков: C/C++, Html, Cobol, Perl, Php, Ruby и др. Вы можете также разработать собственное дополнение для расширения возможностей Eclipse.

1.1. Системные требования

Eclipse разработана для широкого круга операционных систем, таких как Linux, Microsoft Windows и Mac OS. Для ее запуска требуется JVM (Java Virtual Machine) — виртуальная Java-машина, а также JDK (Java Development Kit) — набор для Java-разработки. Загрузить данные пакеты можно с официального сайта разработчика Java — http://java.sun.com. В стандартной сборке ALT Linux «Мастер» данные пакеты уже предустановлены

В табл. 1 представлены минимальные и рекомендуемые системные требования для работы Eclipse.

Таблица 1

Системные требования

Требование	Минимальное значение	Рекомендуемое значение
Версия Java	1.4.0	1.6.0 и выше
Оперативная память	128 Мб	1 Гб и более
Свободное пространство на ЖД	300 Мб	1 Гб и более
Процессор	533 МГц	1,5 ГГц и более

1.2. Введение в Eclipse

При первоначальном знакомстве со средой IDE Eclipse она может показаться несколько сложной для неподготовленного пользователя. Для того чтобы понять основы работы с системой, нужно уяснить себе основные концепции среды: рабочее пространство, инструменты, компоновки, редакторы и представления.

Рабочее пространство

В простейшем случае *рабочее пространство (workspace)* — это каталог для проектов пользователя, в котором располагаются файлы проекта. Все, что находится внутри этого каталога, считается частью рабочего пространства. В нашем пособии будет использоваться для примеров рабочее пространство: /home/user/workspace.

Инструментальные средства Eclipse

Инструментальные средства Eclipse становятся доступны сразу после запуска приложения. Это по существу сама платформа с набором различных функциональных возможностей главного меню, где прежде всего выделяется набор операций по управлению проектом. Фактическая обработка, как правило, осуществляется дополнениями (плагинами), например редактирование и просмотр файлов проектов осуществляется JDT, и т.д.

К инструментам (workbench) относится набор соответствующих *редакторов* и *представлений*, размещенных в рабочей области Eclipse (рис. 6). Для конкретной задачи определенный набор редакторов и представлений называют *перспективой* или *компоновкой*.

Компоновки

Компоновка (perspective) — это набор представлений и редакторов, расположенных в том порядке, который вам требуется. В каждой компоновке присутствует свой набор инструментов, некоторые компоновки могут иметь общие наборы инструментов. В определенный момент времени активной может быть только одна компоновка. Переклю-

чение между различными компоновками осуществляется нажатием клавиш <Ctrl+F8>.

Используя компоновки, вы можете настроить свое рабочее пространство под определенный тип выполняемой задачи. В пособии будут использоваться компоновки, связанные в основном с программированием на Java, такие, как: Debug, Java Browsing, Java.

В Eclipse имеется также возможность создавать свои компоновки. Открыть компоновку можно командой Window / Open Perspective.

Редакторы

Редакторы представляют собой программные средства, позволяющие осуществлять операции с файлами (создавать, открывать, редактировать, сохранять и др.).

Представления

Представления по существу являются дополнениями к редакторам, где выводится информация сопроводительного или дополнительного характера, как правило, о файле, находящемся в редакторе. Открыть представления можно командой Window / Show View. Наиболее часто используемые представления для различных компоновок приведены в табл. 2.

Таблица 2

Компоновки

Компоновка	Представление
Debug	Breakpoints, Debug, Variables, Expressions, Task, Outline, Console
Java Browsing	Projects, Packages, Types, Members
Java	Package Explorer, Problems, Hierarchy, Outline, Javadoc, Declaration

Проект

Проект (project) представляет собой набор файлов приложения и сопутствующих дополнений. При работе с Java используются в основном файлы, имеющие следующие расширения: .java, .jsp, .xml.

Дополнение

Дополнением (plug-in) называют приложение, которое дополнительно может быть установлено в Eclipse. Примером дополнения может выступать JDT.

Мастера

Мастер — это программное средство, которое помогает пользователю в настройках и проведении сложной операции. В Eclipse имеется множество различных мастеров, которые делают работу пользователя в системе удобной и эффективной, беря часть рутинных операций на себя. Примером мастера может выступить мастер создания нового класса, который помогает пользователю в таких операциях, как создание нового файла в нужной директории, создание начального кода класса, автоматическая расстановка модификаторов и т.д. (рис. 1).

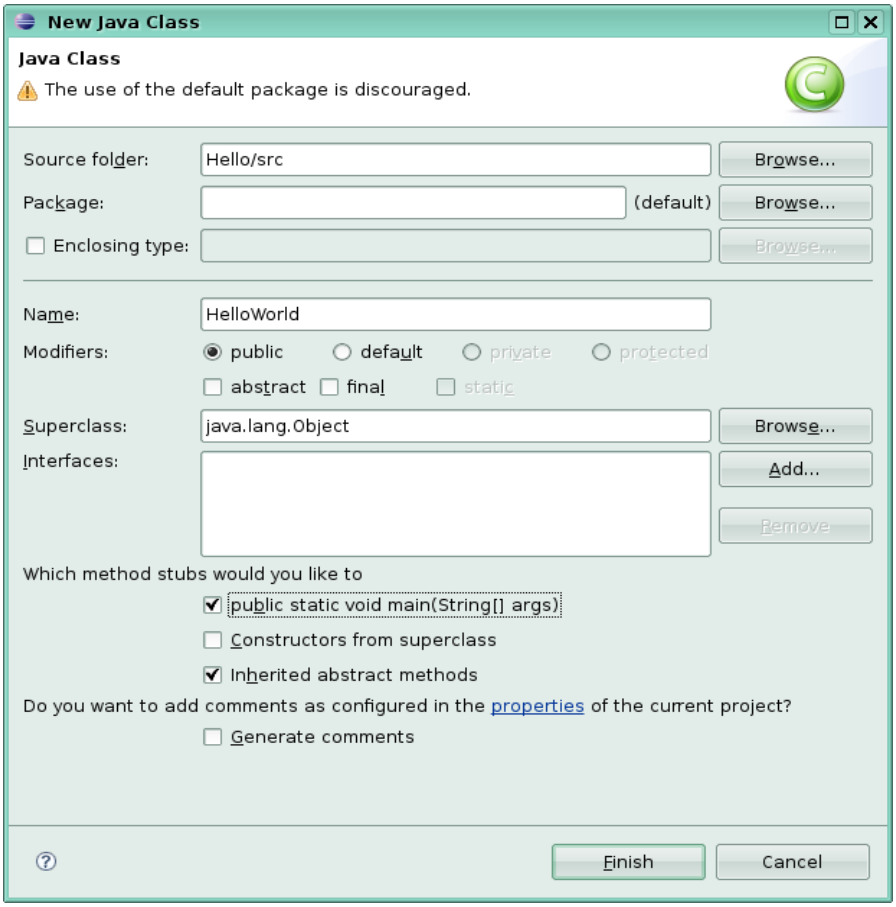


Рис. 1. Мастер создания нового класса

1.3. Установка Eclipse

Перед тем как запустить Eclipse, требуется убедиться, что на вашем компьютере установлена необходимая среда выполнения JRE (версия 1.3 или более поздняя). Проверить версию установленной JRE на компьютере можно командой, введенной в командной строке:

```
java -version
```

Если Eclipse не установлен в сборке Linux, которую вы используете, его можно установить с DVD диска Linux «Мастер». Произвести его установку можно с помощью менеджера пакетов Synaptic. Запустить

менеджер пакетов можно из главного меню KDE. Находится данный менеджер в разделе «Настройка»/«Менеджер пакетов (Программа управления пакетами Synaptic)». Запуск данной программы потребует от вас пароль суперпользователя (Администратора). Для начала нужно произвести поиск компонентов Eclipse, для этого нужно нажать кнопку «Искать» и в поле ввода, появившегося диалогового окна, ввести «Eclipse» (рис. 2).

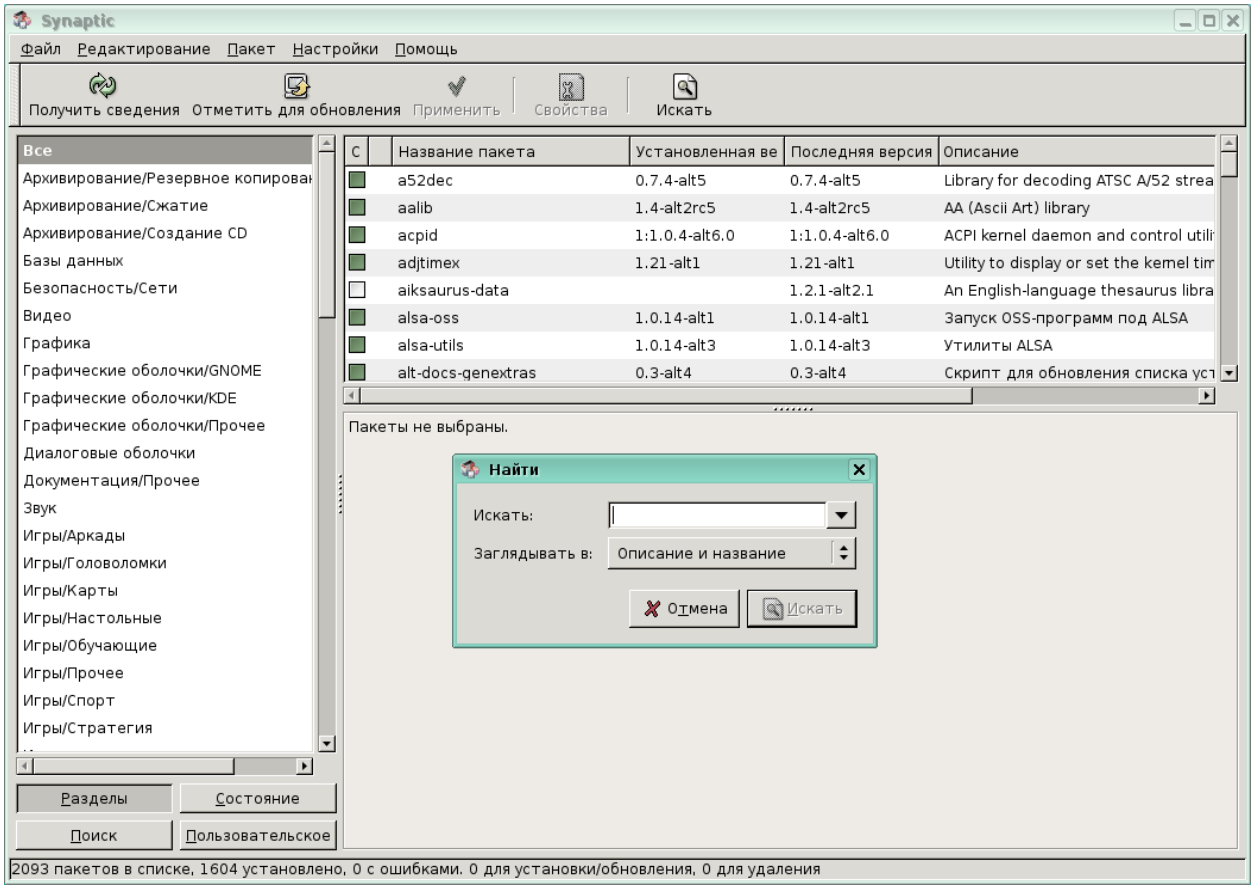


Рис. 2. Поиск пакетов в Synaptic

Если пакета Eclipse нет в имеющихся репозиториях, то необходимо добавить репозитории с DVD диска Linux «Мастер». Сделать это можно с помощью команды:

```
apt-cdrom add
```

Вставьте диск в компьютер и введите указанную команду в терминале. Открыть терминал можно, щелкнув правой кнопкой мыши на пустом месте рабочего стола, и выбрать в раскрывшемся списке «Открыть терминал».

1.4. Первый запуск Eclipse

Первое окно, которое отобразится на экране — это диалоговое окно выбора рабочего пространства (workspace) (рис. 3).

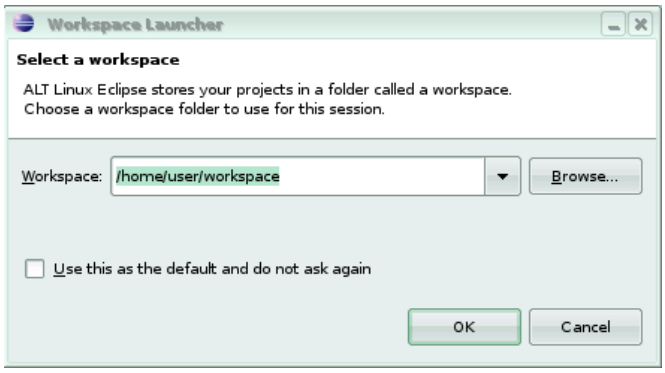


Рис. 3. Выбор папки с рабочим пространством

В данном окне вы можете выбрать нужную вам папку, в которой будут храниться файлы ваших проектов Java. Установка опции на диалоговом окне выбора рабочего пространства, находящейся под списком workspace, даст указание оболочке использовать данное рабочее пространство по умолчанию, и больше данное окно появляться не будет. Для смены рабочего пространства в дальнейшем это окно может быть открыто с помощью команд меню File / Switch Workspace.

Файлы каждого проекта — исходные тексты программ (.java, .jsp), файлы настроек (.xml) и прочие данные будут храниться в указанном вами рабочем пространстве Workspace.

После того, как вы нажмете кнопку «OK», появится страница приветствия (рис. 4), на которой имеется 5 графических кнопок:

- Overview — обзор, содержащий ссылки на обучающие интернет-ресурсы eclipse;
- Tutorials — уроки, содержит несколько примеров создания простейших приложений Java;
- What’s new — «что нового», содержит обзор основных нововведений;
- Samples — примеры, содержит несколько примеров разработки, которые должны быть предварительно установлены для того, чтобы их можно было просмотреть;
- Workbench — «рабочий стол» — это рабочая область программиста.

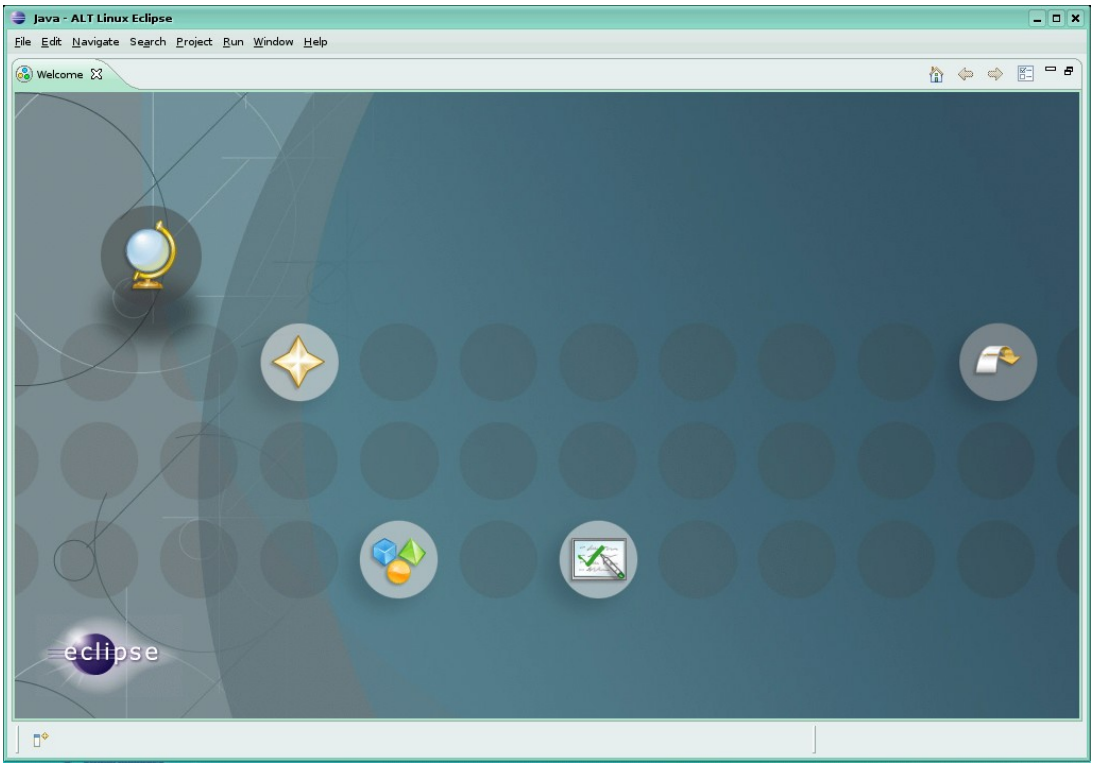


Рис. 4. Страница приветствия

Для того, чтобы приступить к работе, нажмите кнопку «Workbench». По умолчанию откроется универсальный рабочий стол (рис. 5).

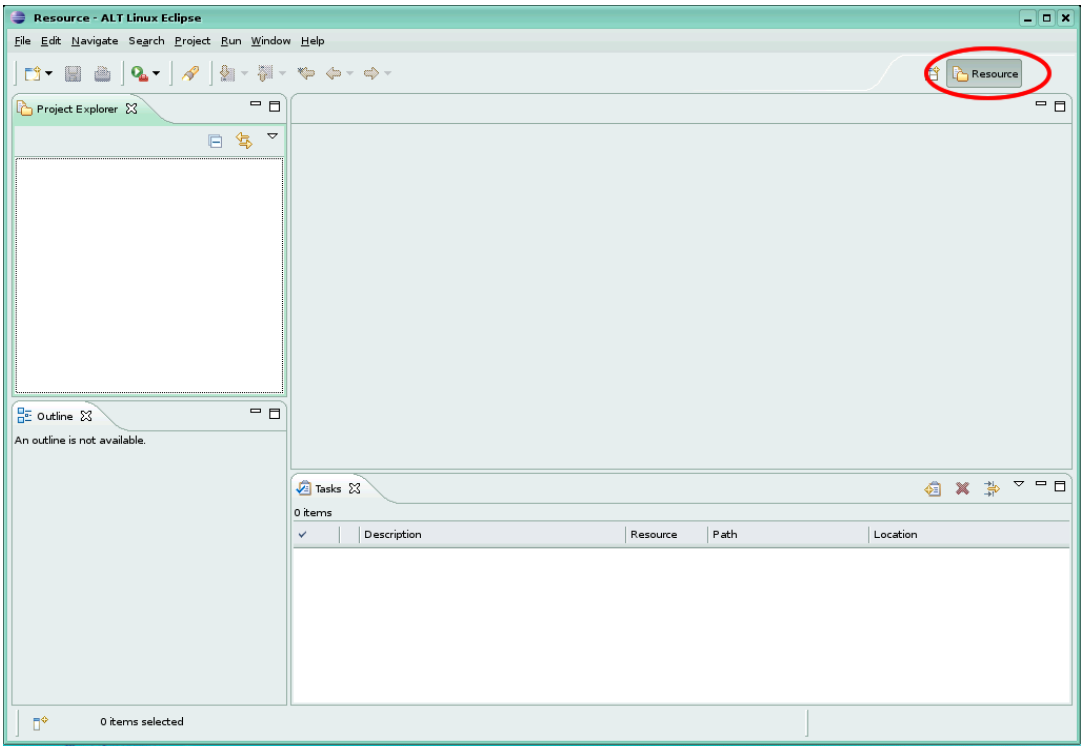


Рис. 5. Универсальный рабочий стол

Этот универсальный рабочий стол пока еще не содержит в полном объеме всех важных элементов рабочего стола Java. На рис. 5 выделена метка, которая отображает текущий режим рабочего стола. Для того, чтобы переключиться в другой режим, нужно нажать кнопку, находящуюся слева от выделенной метки, и в раскрывающемся списке выбрать нужный режим. В нашем случае нужно выбрать Java.

Компоновка Java

На рис. 6 можно выделить несколько основных элементов графического интерфейса пользователя среды Eclipse в компоновке Java.

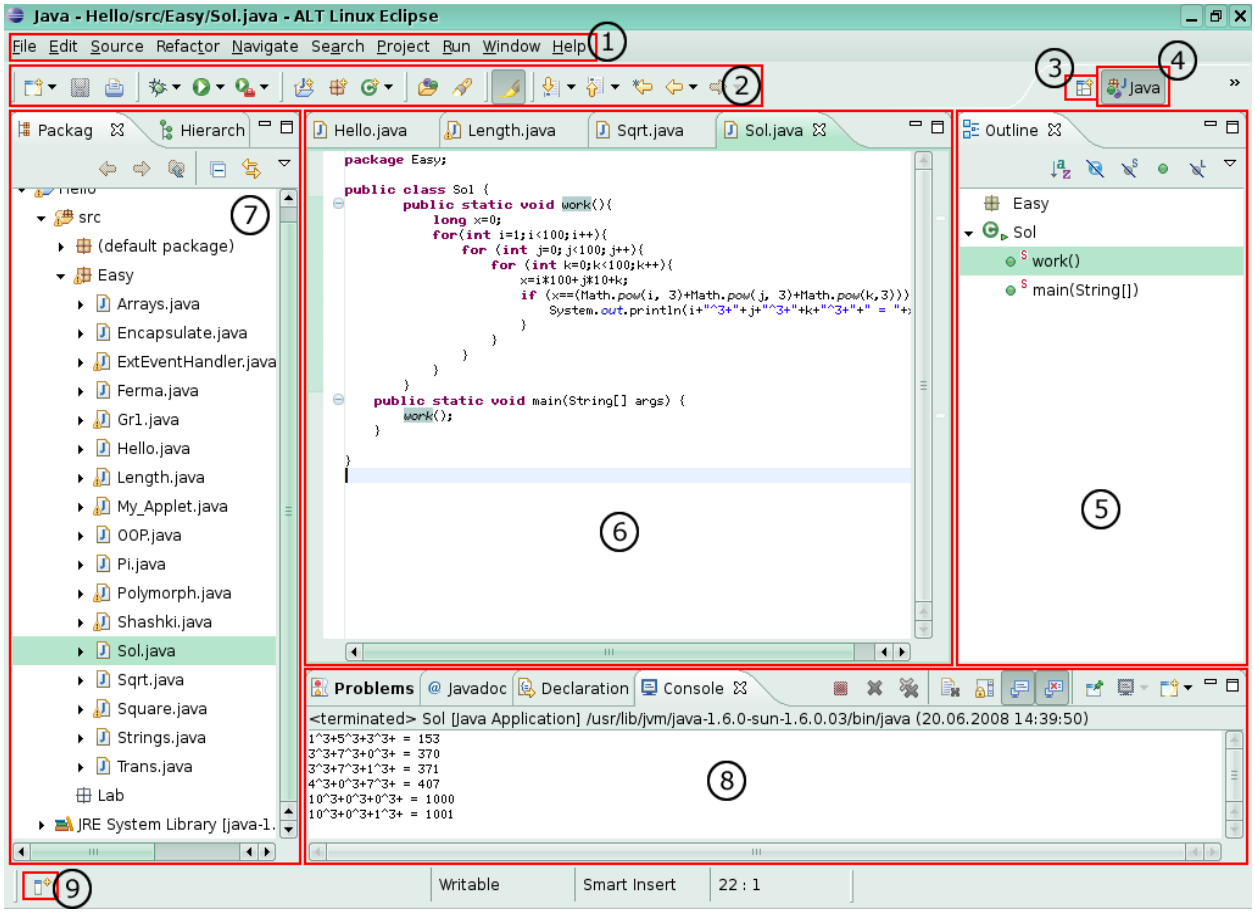


Рис. 6. Компоновка Java

На рис. 7 изображены три основные компоновки, которые используются в работе над Java-приложениями. Каждая компоновка содержит свой набор различных панелей и представлений, а так же их форму и расположение.

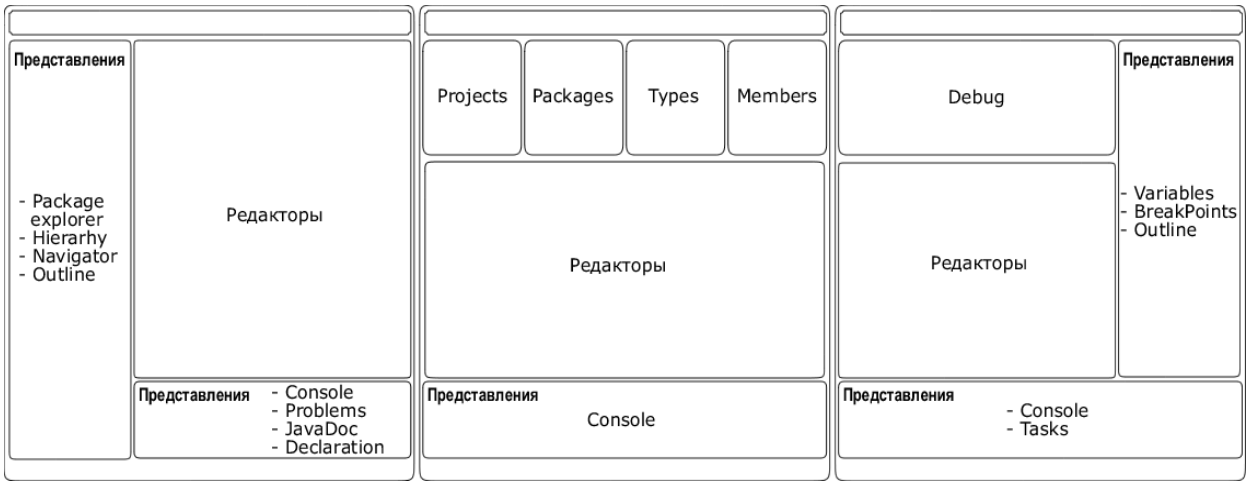


Рис. 7. Компоновки Java, Java Browsing, Debug

Рассмотрим примерный состав возможных представлений на примере компоновки Java. Примерный состав компоновки Java будет следующий.

- Строка меню (№ 1, рис. 6).

Главное меню платформы Eclipse с набором функций для работы с проектами (подробно рассмотрена в следующем разделе).

- Панель инструментов (№2, рис. 6).
- Окно браузера проекта и иерархии (№ 7, рис. 6).

Предназначено для отображения структуры рабочего пространства проекта и представляет собой иерархическую структуру каталогов и файлов, входящих в проект.

- Окна представлений (№ 8, рис. 6).

Имеется 4 основных вкладки:

1) Problems — предназначена для отображения ошибок при компиляции, а также во время написания программы;

2) Javadoc — отображение комментариев к выбранным объектам;

3) Declaration — отображение части кода, в котором происходит декларация выбранного объекта;

4) Console — системная консоль, в которую происходит вывод данных программы.

- Перспективы (компоновки) (№ 3, рис. 6)

Позволяют переключаться из одной компоновки в другую. В пособии будут использоваться следующие компоновки: Java, Debug и Java Browsing (рис. 7);

- Редактор кода (№6, рис. 6)

Предназначен для ввода и редактирования исходного текста программ Java.

1.5. Интерфейс пользователя

Рассмотрим подробнее назначение основных компонентов, составляющих интерфейс пользователя. В нашем пособии мы в основном будем работать с компоновкой Java.

Таблица 3

Главное меню

Команда меню	Назначение
File	Содержит элементы управления проектами и файлами. Позволяет создавать новые проекты, классы, интерфейсы и пр., сохранять, закрывать, переносить, переименовывать, экспортировать имеющиеся структуры проекта, а также импортировать внешние объекты, изменять местоположение рабочего пространства и переключаться между различными рабочими пространствами
Edit	Содержит элементы управления текстовой информацией: копирование, вырезание, вставка, удаление, отмена последнего действия, возврат после отмены, добавление закладок и заданий, а также поиск
Source	Содержит элементы управления исходным кодом, такие, как: управление импортом элементов, управление комментариями, генерацию методов и конструкторов классов и т.д.
Refactor	Рефакторинг элементов проекта. Помогает производить модификации членов проекта (классов, методов и полей), такие, как перемещение, переименование и пр. без потери связей и целостности приложения
Navigate	Содержит элементы навигации по проекту
Search	Основной инструмент поиска. Искать можно файлы, проекты и текст в любом диапазоне от файла до рабочего пространства
Project	Инструмент управления проектами: открытие, закрытие, компиляция, генерация javadoc и пр.
Run	Управление запуском приложения, позволяет запустить приложение, отладку, а также приложения по точкам прерывания
Window	Управление окнами и перспективами. Переключение между компоновками. Вызов требуемых представлений

Команда меню	Назначение
Help	Вызов справки, а также содержит вызов обновлений и описание программы Eclipse

Панель инструментов

Данная панель содержит кнопки быстрого доступа к наиболее часто используемым функциям Eclipse. Кнопки по функциональному назначению сгруппированы в группы. Первая группа — создание (проекта, пакета, класса, интерфейса, перечисления и т.д), сохранение и печать относится к команде File. Вторая группа — отладка, запуск и запуск с параметрами — относится к команде Run. Третья группа — новый проект, новый пакет и новый Java-файл (класс, интерфейс, перечисление) — дублирует первую кнопку для максимально быстрого доступа. Последняя, четвертая, группа содержит кнопки, относящиеся к поиску и навигации.

Представления

Окна браузера проекта и иерархии

Окно, в котором приводится древовидная структура проекта, предназначено для быстрого просмотра или выбора элемента в иерархии классов проекта. Здесь в виде дерева отображается структура проекта, с помощью которой можно быстро переключаться между пакетами, классами, методами и полями проекта приложения.

Окна отчетности

В этих окнах отображаются основные события при работе с приложением — ошибки (вкладка Problems); комментарии (вкладка javadoc); код декларации (вкладка declaration).

Перспективы

С помощью кнопки  производится переключение между различными компоновками, такими, как Java, Debug, Java Browsing, C++ и др.

Быстрые окна

Кнопка «быстрые окна», или быстрый вызов представления, находится в левом нижнем углу рабочего стола (№ 9, рис. 6).

Данная кнопка добавляет в окно браузера проекта различные панели (представления), такие, как Ant, Problems, Search, Navigator и т.д. Если вы случайно закрыли какую-либо панель, например Package Explorer, с помощью данной кнопки можно ее восстановить.

При вызове новой панели с помощью данной кнопки, вызываемая панель помещается поверх всех структурных элементов рабочего стола, закрывая собой некоторую часть рабочего стола. Для более оптималь-

ного использования пространства рабочего стола нужно после вызова панели поместить ее в одну из областей: ее можно добавить к левой панели, на которой располагается по умолчанию Package Explorer и Hierarchy, можно добавить к окнам отчетности или разместить ее отдельно от остальных панелей в любом месте рабочего стола, не перекрывая при этом редактора кода, для этого необходимо нажать мышью на вкладке заголовка панели и перетащить ее в необходимое место. В процессе перемещения на рабочем столе будут появляться рамки, которые покажут, как будет располагаться перемещаемая панель.

Редактор кода

Это окно, предназначенное для ввода и редактирования исходных текстов программ. Отображаться может содержимое только одного файла проекта. Если открыто более одного файла, в верхней части редактора появляется строка вкладок, с помощью которой можно быстро переключаться между различными файлами, выбирая курсором мыши нужную вкладку.

Небольшая, но очень удобная особенность, которая есть у редакторов кода многих сред программирования, — это сворачивание кода. Если в каком-нибудь классе есть метод, занимающий своим кодом много места в редакторе, его можно свернуть, выигрывая тем самым не только рабочее место, но и делая код более читаемым. Для того чтобы свернуть содержимое кода метода, нужно щелкнуть мышью по значку минус в левом столбце. Для развертывания — по аналогии, на значке плюс.

Сообщения об ошибках

С помощью синтаксического контроля программы интерпретатором Java выявляются конструкции (сочетание символов), недопустимые с точки зрения правил их построения, принятых в Java. Поиск осуществляется в автономном (фоновом) режиме, то есть в процессе написания кода программы, без запуска приложения.

Проблемы сборки проекта отображаются в представлении Problems и отмечаются (маркируются) в редакторе определенным образом с помощью различных маркеров: графических значков, подчеркиваний, информации в всплывающих информационных окнах и др.

В программах ошибки подразделяются на три основных типа: синтаксические (компиляции), выполнения и логические. «Ошибочные» места кода подчеркиваются волнистой красной линией.

Синтаксические ошибки обычно возникают из-за неправильного набора текста программы на клавиатуре: пропуск запятых, точек с запятой в конце оператора, пропуск скобок в методе, классе, незакрытые

кавычки и другие неверно использованные синтаксические конструкции языка. Для поиска и устранения таких ошибок в арсенале Eclipse используется широкий набор программных средств: маркера, помощники в создании кода, дополнения и т.п.

Ошибки выполнения возникают в тех случаях, когда синтаксически правильная программа совершает какое-либо неверное действие в процессе своего исполнения. Например, ситуация — попытка деления на ноль, обработка отсутствующих данных, обращение к индексу массива, значение которого выходит за допустимые границы и т.д. Эти ошибки можно обнаружить только в процессе выполнения программы. Компилятор выдает сообщение в ходе работы программы, если ошибка обработана или распознана внутренними средствами.

Логические ошибки не приводят к прекращению выполнения программы в Eclipse. Информацию о них можно получить только после выполнения программы по неправильным результатам решения задачи. Логические ошибки являются следствием неправильного алгоритма программы, и среда разработки тут мало может помочь. Требуется вернуться на этап проектирования алгоритма, проверить его правильность и только затем приступить к исправлению.

Далее мы остановимся главным образом на устранении синтаксических ошибок и рассмотрим на примерах возможные способы и средства их устранения, предоставляемые Eclipse.

Если кликнуть мышью на строку с ошибкой, Eclipse откроет или сделает активным окно кода файла класса и выделит в коде «проблемное» место: метод, модификатор, тип, поле и т.д., которое вызывает ошибку компиляции.

Пример. Локализация и устранение ошибок в программе.

1. Создайте новый класс ErrorTests.java и введите следующий код:

```
public class ErrorTests {
     public static void main(String[] args) {
          System.out.print(«Hello»);
     }
  }
```

2. Добавьте синтаксическую ошибку путем удаления фигурной скобки, открывающей тело класса (рис. 8).

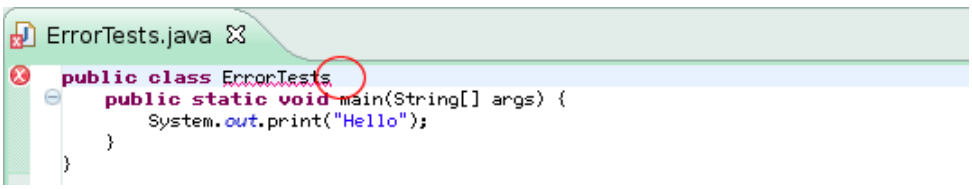


Рис. 8. Создание ошибки

3. Щелкните «Сохранить». Проект соберется заново и проблемное место отобразится несколькими способами (рис 9):

- В представлении Problems, проблемы можно пролистывать.
- В представлениях Package Explorer, Type, Hierarchy или Outline проблемные метки появляются на всех элементах Java, вызывающих проблемы, а так же на их родительских элементах.
- В редакторе в вертикальной линейке, отображается маркер напротив строки, вызывающей проблему.
- Волнистые линии могут появится под словом или символом, являющимся причиной ошибки, и

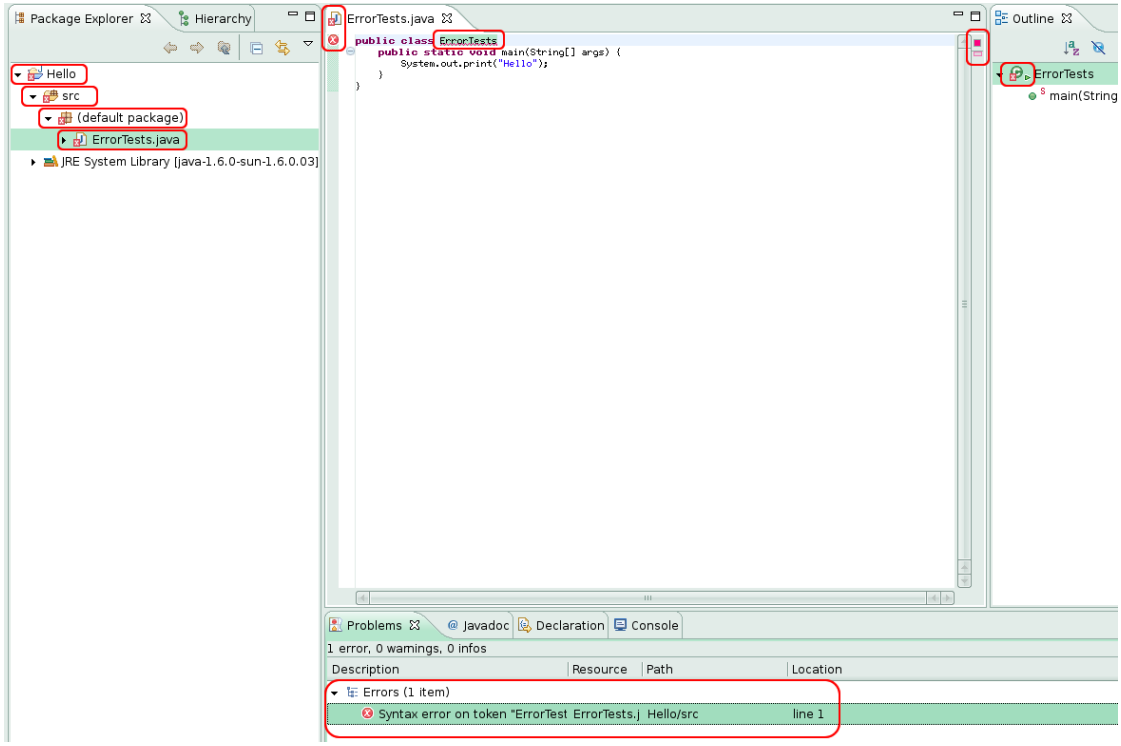


Рис. 9. Отображение ошибок

4. Вы можете навести курсор на маркер в вертикальной линейке для отображения описания ошибки (рис. 10).

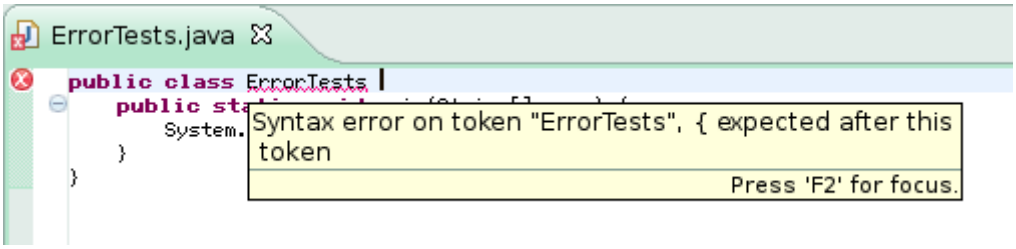


Рис. 10. Комментарий к ошибке

5. Щелкните кнопку «Закрыть» ("X") на вкладке редактора, что бы его закрыть.

6. В представлении Problems, выберите проблему из списка. Открыв его контекстное меню и выберите «Go To» (рис. 11). Файл откроется в редакторе на проблемном месте.

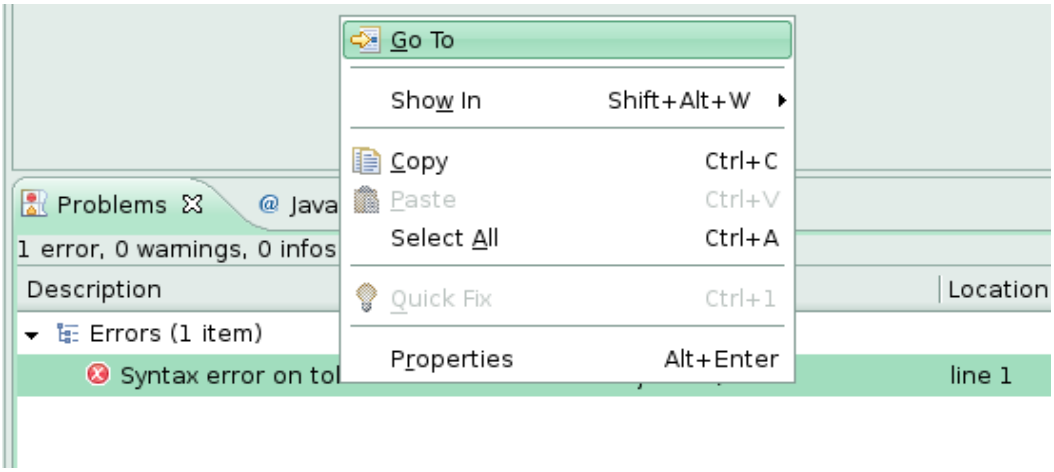


Рис. 11. Быстрый переход к ошибке

7. Исправьте ошибку путем добавления фигурной скобки. Щелкните кнопку «Save». Проект соберется заново и все маркеры ошибок пропадут.

8. Строку `System.out.print(«Hello»);` измените на `System.out.print(«Hello»+x);`.

9. В процессе ввода, ошибка отметится волнистой линией на переменной x, для индикации проблемы. Наведение курсора на подчеркнутое слово, вызывающее проблему вызовет всплывающее окно с описанием данной проблемы.

10. В вертикальной линейке маркеров появится значок с лампочкой. Лампочка сигнализирует о том, что для данной проблемы доступны варианты коррекции (рис. 12).

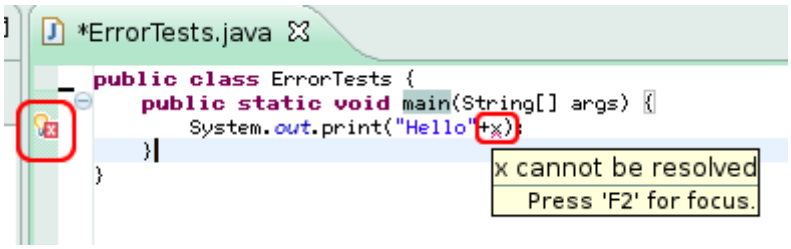


Рис. 12. Исправляемая ошибка

11. Щелкните на проблемное место, и выберите Quick Fix из пункта Edit строки меню. Вы так же можете нажать клавиши Ctrl+1 или щелкнуть левой кнопкой мыши на лампочке. Отобразится диалог с возможными вариантами исправлений (рис. 13).

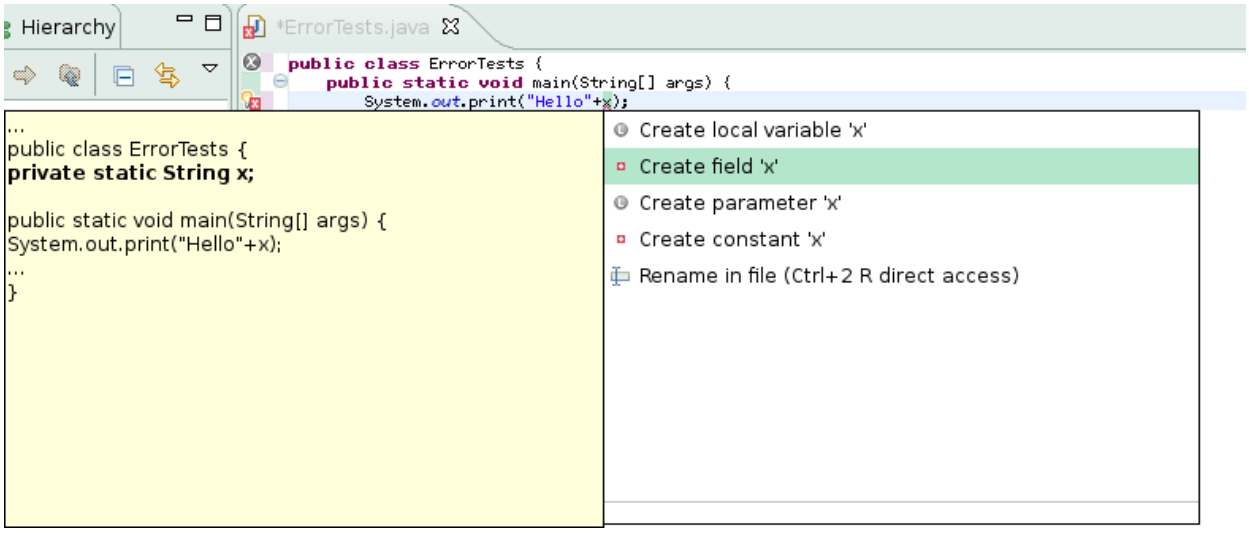


Рис. 13. Возможности QuickFix

12. Выберите «Create field 'x'» что бы исправить ошибку. В класс добавится поле x и волнистая линия, подсвечивающая проблемное место исчезнет.

13. Вы можете настроить процесс отображения ошибок в меню `General/Editors/Text Editors/Annotations`.

Панель Navigator

Панель (представление) Navigator служит для отображения структуры проекта, в котором расположены файлы проекта в виде дерева каталогов и файлов. Как и другие панели, она может быть вызвана нажатием кнопки , находящейся в левом нижнем углу для быстрого просмотра, либо командой из главного меню Window/Show View/Navigator.

В папке *src* (source) вы видите список файлов с расширением .java — это исходные тексты программы.

В папке *bin* (binary) находятся скомпилированные файлы классов программы, содержащие двоичный байт-код Java. Файлы с расширением .classpath и .project содержат служебную информацию о расположении классов и структуре проекта.

Панель Outline

Панель (представление) Outline отображает структуру текущего открытого файла Java, где в виде дерева отображаются структуры ООП (классы, поля, методы и т.д.), находящиеся в текущем открытом файле. Как и другие панели, она может быть вызвана нажатием кнопки , нахо-

дящейся в левом нижнем углу для быстрого просмотра, либо командой из главного меню Window/Show View/Outline.

С помощью данной панели можно быстро осуществлять переход к нужным методам и полям классов, что актуально в случае очень больших программ, когда количество программного кода исчисляется десятками страниц и относительно трудно найти нужные члены класса вручную, а также при просмотре кода, написанного другими людьми. Для перехода к нужному члену файла нужно дважды щелкнуть на требуемый элемент списка.

1.6. Настройки среды

Перед началом работы можно настроить поведение оболочки под свои требования. Окно настроек (рис. 14) можно открыть из главного меню, выбрав Window/Preferences.

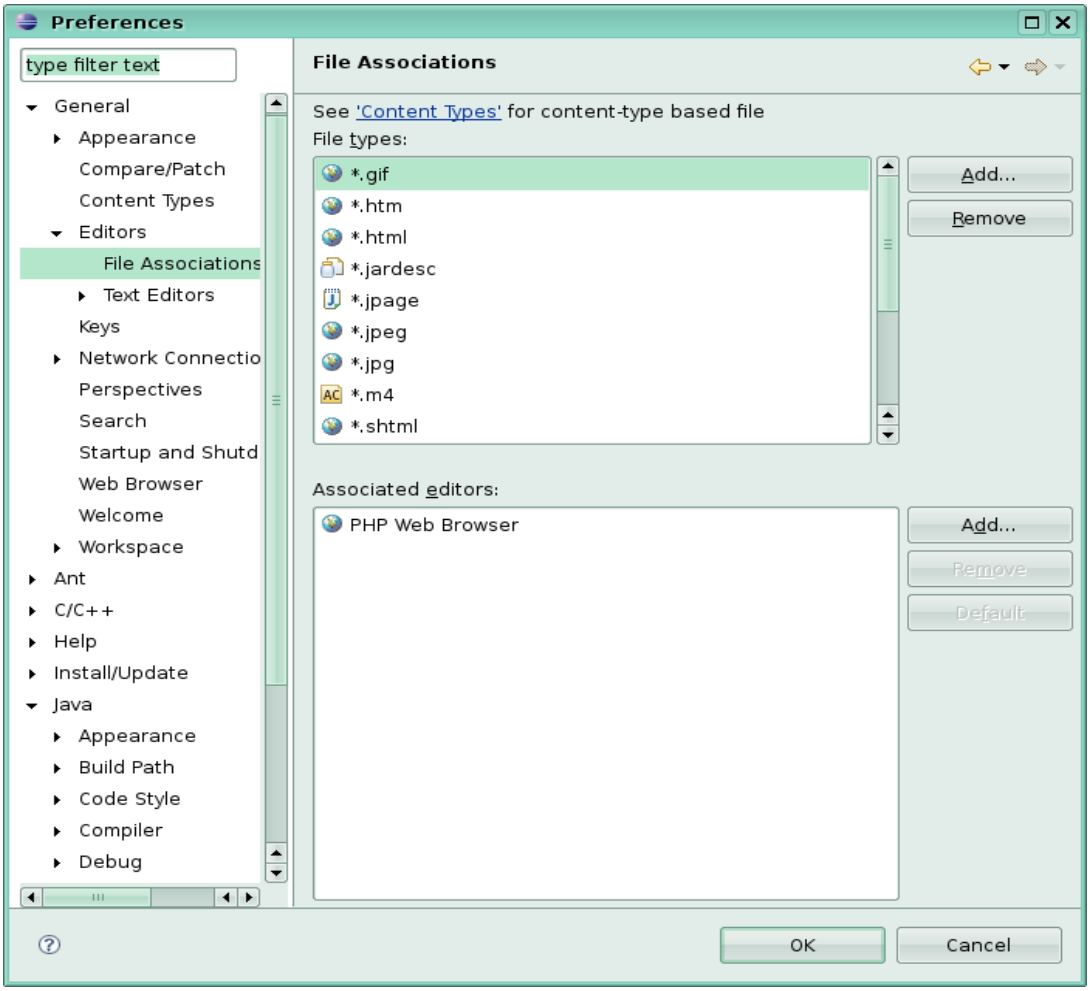


Рис. 14. Окно настроек Eclipse

Окно настроек состоит из двух частей: списка разделов в форме дерева и панели их настройки. В панели управления настройками отоб-

ражаются доступные настройки выбранного раздела. Пункты, перед которыми имеется черный треугольник, содержат в себе подпункты, для доступа к которым нужно либо сделать двойной щелчок мышью, либо один щелчок на треугольнике.

Набор настроек состоит из девяти основных разделов, указанных в табл. 4.

Таблица 4

Настройки Eclipse

Разделы	Назначение
General	Содержит общие настройки, не попадающие в остальные пункты
Ant	Содержит настройки сборщика проектов Ant
C/C++	Содержит настройки поддержки и разработки C++ приложений
Help	Содержит настройки справочной системы, такие, как способы открытия справочной информации, используемый Web-браузер и настройка сторонних справочных служб в сети Интернет
Install/Update	Содержит настройки процесса обновления Eclipse, а также настройки автообновления
Java	Один из обширнейших разделов настроек, в котором содержится огромное количество различных настроек Eclipse в режиме Java, начиная от подсветки и стиля кодирования и заканчивая подробными настройками компилятора
PHPeclipse Web Development	Содержит настройки, связанные с разработкой Java-приложений для Web. Включает в себя настройки браузера, серверов Apache и MySQL
Run/Debug	Содержит настройки, параметры запуска и отладки программ
Team	Содержит настройки командной разработки приложений. Чаще всего, приложения разрабатываются не одним человеком, а целыми группами — командами программистов, которым необходимо поддерживать связь. В данном пункте можно настроить особенности командной разработки

Из всех перечисленных разделов в табл. 4 интерес начинающим программистам Java могут представлять только пункты General и Java. В пособии нам будет достаточно использовать имеющиеся настройки по умолчанию.

1.7. Создание проекта

Проект Java представляет собой каталог на жестком диске, содержащий библиотеки JRE, исходные коды, картинки и прочее. Управлять содержимым пакета лучше с помощью среды. Хотя никто не запрещает изменять содержимое файлов и структуру каталогов в проекте с помощью сторонних редакторов и браузеров, все же рекомендуется это делать средствами IDE, поскольку возможны серьезные ошибки и сбои при компиляции и работе приложения в случае использования других программ.

Пример. Создадим новый проект с именем Hello. Нажмите на черный треугольник кнопки  на панели инструментов и в раскрывшемся списке (рис. 15) выберите Java Project (Проект Java).

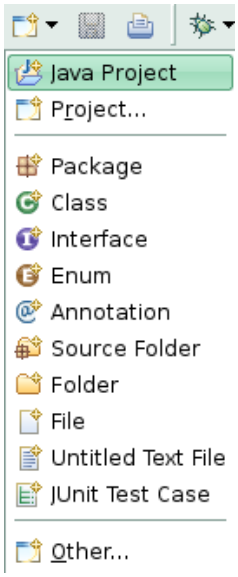


Рис. 15. Создание нового проекта

Следующим будет окно с начальными настройками проекта (рис. 16).

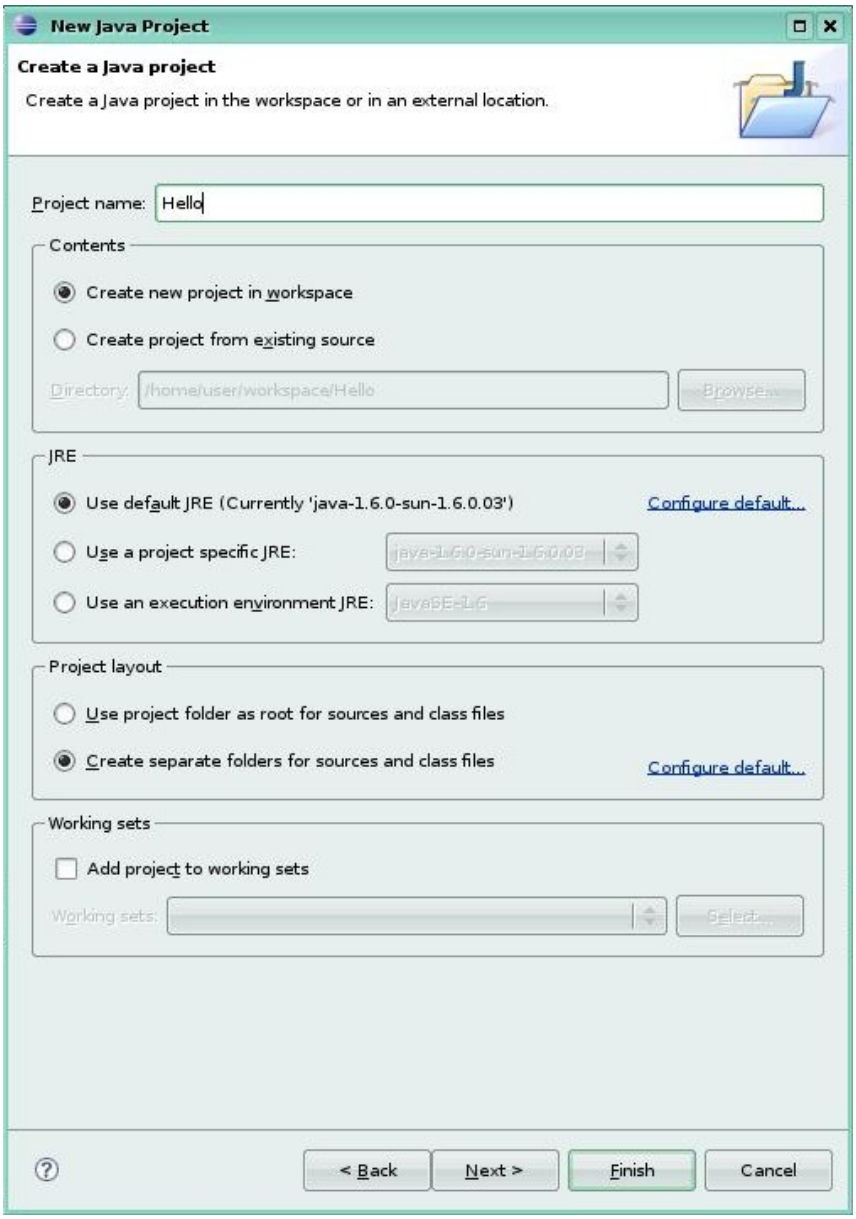


Рис. 16. Параметры нового проекта

В поле Project name (Имя проекта) введите название нового проекта «Hello». В группе Contents (содержимое) можно выбрать Create new project in workspace (Создать новый проект в рабочем пространстве), то есть в той папке, которую вы указали в качестве рабочего пространства, или же выбрать Create project from existing source (создать проект из имеющихся «исходных кодов»), то есть создать проект, который ранее уже был создан в других средах программирования и не является в строгом смысле проектом Eclipse.

Сейчас мы остановим выбор на команде Create new project in workspace (Создать новый проект в рабочем пространстве). В группе опций JRE (Java Runtime Environment) можно выбрать тип JRE для нового

проекта, остановившись на опции Use default JRE (использовать JRE по умолчанию). В группе Project layout (Формат проекта) можно выбрать один из двух форматов: либо файлы класса и файлы источника будут иметь разделенные папки (Separate folders), либо папка проекта будет корневой (as root), хотя для файлов классов и для файлов источников здесь лучше выбрать вариант разделенных папок. Далее нажмите кнопку «Finish».

Теперь проект создан и «рабочий стол» примет вид, представленный на рис. 17.

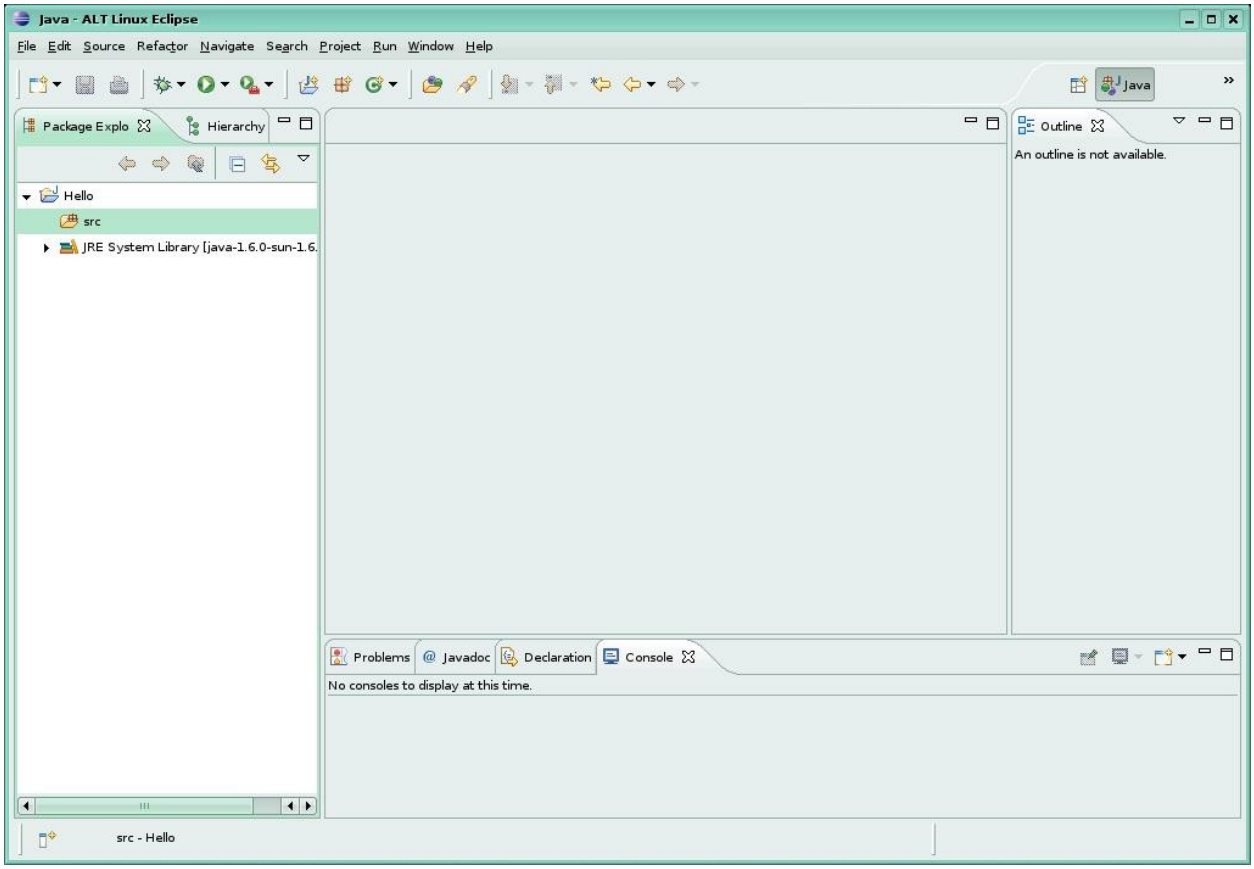


Рис. 17. Вид «рабочего стола» нового проекта

В левом окне Package explorer отображается структура текущего проекта. Теперь создадим наш первый класс: нажмем черный треугольник на кнопке  и в раскрывшемся списке выберем Class (рис. 18).

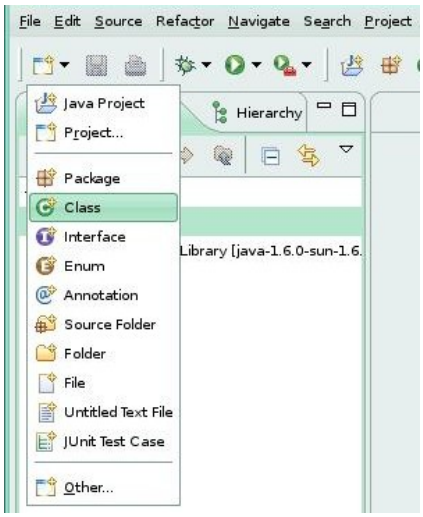


Рис. 18. Создание нового класса

Появится диалоговое окно создания нового класса (рис. 19).

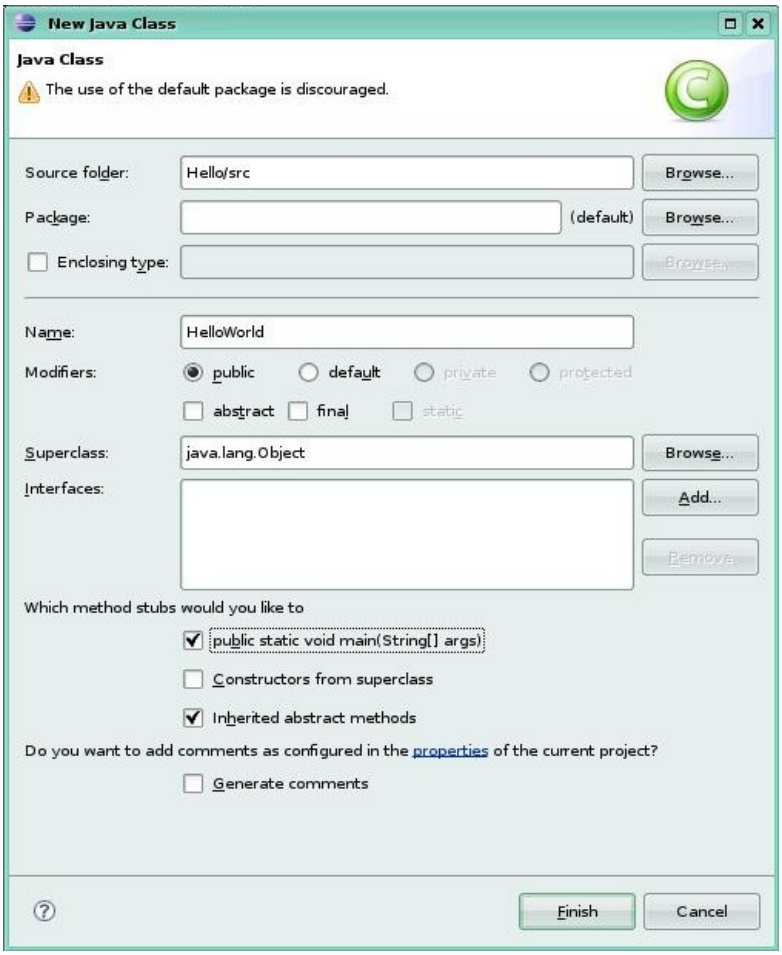


Рис. 19. Параметры нового класса

В поле Source folder находится путь к файлу класса из рабочего пространства, там вы видите Hello/src, это означает, что класс будет

располагаться в папке Hello/src, так как наш проект называется Hello, а src — это имя основного пакета, в котором хранятся исходные коды проекта. В поле Package (пакет) можно указать пакет, в котором будет находиться класс, поскольку мы не создавали еще никаких пакетов, то пусть наш класс будет находиться прямо в src. В поле Name введите имя класса HelloWorld; обратите внимание на то, что в имени класс нельзя использовать пробелы, как только в поле Name появится пробел, кнопка Finish станет неактивной. Здесь можно также выбрать набор модификаторов для класса, класс-родитель и интерфейсы класса. В нижней части окна имеется группа из трех кнопок, которая позволяет автоматически добавить некоторые основные методы:

- public static void main(String[] args) — уже известный нам метод main();
- Constructor from superclass — конструктор класса-родителя;
- Inherited abstract methods — наследованные абстрактные методы.

Нажимаем кнопку «Finish». Теперь в нашем проекте есть один класс HelloWorld (рис. 20), после чего можно приступить к созданию программы.

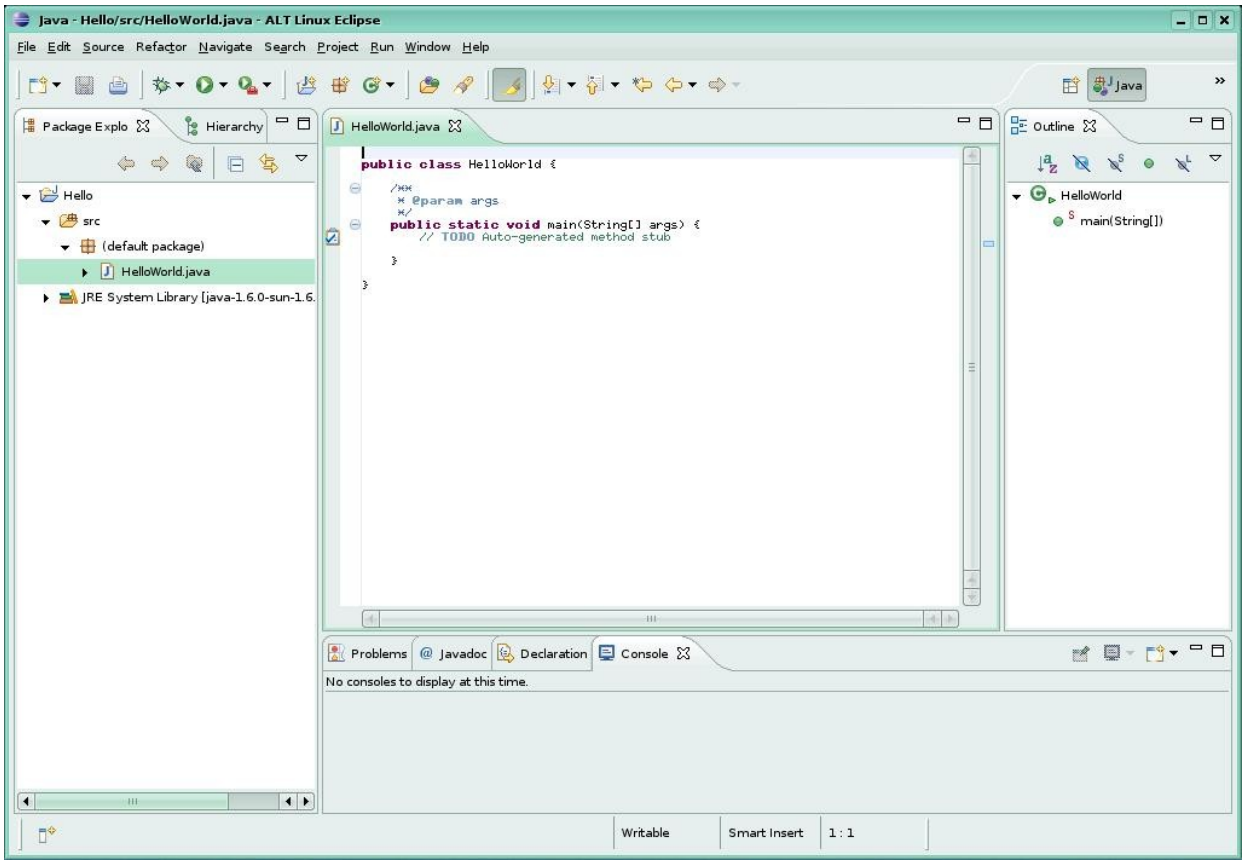


Рис. 20. Новый класс в проекте

При наличии нескольких проектов лучше закрыть предыдущий проект перед открытием или созданием нового проекта. Для закрытия проекта нужно выделить в окне Package Explorer нужный проект, затем в строке меню, в пункте «Project», выбрать «Close», либо, щелкнув правой кнопкой мыши на проект в Package Explorer, в раскрывающемся меню выбрать пункт «Close».

Компиляция и запуск средствами командной строки

Исходные тексты java-программ можно откомпилировать средствами пакета JDK. Для компиляции используется модуль javac, а для запуска скомпилированных .class файлов — интерпретатор java.

Что бы произвести компиляцию исходного файла .java, содержащего код программы, в консоли (терминале) достаточно ввести строку:

javac [*options*] *имя_файла* [*@argfiles*]

Options — параметры компиляции, задающиеся с помощью символа «-». Например:

`-version` — информация о версии установленного ПО;

`-encoding` — установка кодировки текста компилируемого файла;

`-nowarn` — отключение отображения предупреждений.

Подробную информацию о имеющихся параметрах компиляции можно узнать с помощью команды:

```
man javac
```

Имя_файла — это имя файла, содержащего исходный код программы. По умолчанию, при запуске с терминала, текущим каталогом является домашняя папка пользователя /home/*user.* В качестве имени файла указывается либо полное имя файла, вместе с путем до указанного .java-файла, либо собственное имя.

@argfiles — параметр, предназначенный для упрощения команды компиляции. Если необходимо откомпилировать несколько файлов, то требуется записать список имен файлов в текстовый файл (например files), а список параметров в другой (например params). Затем командой `javac @params @files` можно будет произвести компиляцию указанных классов с заданными параметрами, тем самым значительно упрощая саму команду.

В качестве примера произведем компиляцию из терминала простейшей программы, которая была использована в разделе описания ошибок. Исправьте все ошибки в этой программе и сохраните ее. Файл исходного кода программы находится в папке

/home/*user*/*workspace*/*Hello*/src/ErrorTests.java

Папки user, workspace и Hello в вашем случае могут иметь другие имена. (User — это имя пользователя, workspace — имя рабочего пространства, Hello — имя Java-проекта).

Запустите терминал и в соответствии с именами папок введите команду:

```
javac workspace/Hello/src/ErrorTests.java
```

В случае удачной компиляции в данной папке появится файл `ErrorTests.class`.

Для запуска откомпилированной программы нужно ввести команду:

```
java [options] имя_файла.class [params]
```

Введите в терминале команду

```
java workspace/Hello/src/ErrorTests.java
```

В результате выполнения программы в командной строке терминала будет выведена строка «Hello».

1.8. Поддержка, советы, рекомендуемые ресурсы

Общие советы

Eclipse — многофункциональная и гибко настраиваемая платформа, которая имеет множество настроек, подробное изложение которых заняло бы очень много места и времени. Мы рекомендуем начинающим использовать справочную систему (Help) Eclipse. Там вы найдете множество советов и рекомендаций по различным приемам работы с Eclipse.

Закладки

Закладки удобно использовать при работе с кодом, содержащим большое число строк. Находясь в редакторе, выберите пункт Edit/Add Bookmark. Представление Bookmark позволяет просматриваться и передвигаться по закладкам.

Восстановление конфигурации

При восстановлении исходной конфигурации (после ее случайного изменения) необходимо проделать следующие команды из главного меню: Window/Reset Perspective.

Мастер очистки

Если у вас складывается впечатление, что откомпилированные файлы .class не синхронизируются в Eclipse, когда не учитываются изменения или вы видите сообщение об ошибках неожиданного содержания, запустите мастера очистки Clean командой Proect/Clean. Данная команда сбрасывает все результаты предыдущих сборок проекта, а если включена опция Build automatically, то происходит полная перестройка проекта с учетом последних изменений произведенных пользователем.

Скрытые файлы

Надо иметь в виду, что при создании проекта создаются скрытые файлы и каталоги, например .classpath, .project, metadata и др. При удалении проекта не забудьте их удалить, если вы хотите оставить каталог проекта.

Удаление Eclipse

Удаление Eclipse, как и установка, осуществляется при помощи менеджера пакетов Synaptic. Для осуществления операции удаления необходимы права суперпользователя.

В табл. 5 представлен набор сочетаний клавиш, которые наиболее часто используются при работе с Eclipse.

Таблица 5

Комбинации клавиш

Сочетание клавиш	Назначение
Ctrl+N	Создание нового объекта, при этом будет запущен список выбора мастеров
Ctrl+M	Свертывание, развертывание окон редакторов и представлений
Ctrl+Shift+Пробел	Подсказка в параметрах метода
Ctrl+Shift+M	Вставка
Ctrl+F	Простой поиск
Ctrl+H	Сложный поиск
Ctrl+K	Повторить последний поиск
Ctrl+/	Комментировать одну строку кода
Ctrl+F6	Перебор редакторов. Следующий редактор
Ctrl+F8	Перебор перспектив. Следующая перспектива
Ctrl+F7	Перебор представлений. Следующее представление
Tab	Увеличение отступа кода
Shift+Tab	Уменьшение отступа кода
F12	Активировать редактор
F3	Открывает объявление выделенного элемента
Alt+-	Показать системное меню

Рекомендуемые ресурсы

Eclipse

1. Платформа Eclipse — http://www.eclipse.org/
2. Сообщество Eclipse — http://www.eclipse.org/community/
3. Статьи, посвященные Eclipse — http://www.eclipse.org/articles
4. Официальный обзор Eclipse — http://www.eclipse.org/whitepapers/eclipse-overview.pdf
5. Конференция EclipseCon — http://www.eclipsecon.org/
6. Common Public License Version 1.0 — http://www.eclipse.org/legal/cpl-v10.html
7. Список рекомендуемой литературы по Eclipse — http://www-128.ibm.com/developerworks/library/os-ecl-read/?S_TACT=105AGX01&S_CMP=LP
8. Ресурсы для разработчика в Eclipse — http://www.ibm.com/developerworks/ru/opensource/top-projects/eclipse.html

Ant

Ant — http://ant.apache.org/

Apache

Web-сервер Apache — http://www.apache.org/

OpenOffice.org

Офисный пакет — http://www.openoffice.org/

Java

1. Документация по классам Java. [Электронный ресурс] — http://java.sun.com/docs/
2. Учебные пособия по Java (Java Tutorial). [Электронный ресурс] — http://java.sun.com/docs/books/tutorial/
3. Ответы на часто задаваемые вопросы (FAQ). Линден П. [Электронный ресурс] — http://www.afu.com/javafaq.html

Конференции

1. Сетевая конференци программистов Java Comp.lang.java.programmer
2. Сетевая конференци Comp.lang.java.*

Контрольные вопросы

1. Укажите, для каких целей предназначена IDE Eclipse.
2. Укажите последовательность действий для вызова панели Outline.
3. Перечислите основные представления, доступные в перспективе Java.
4. Существует ли возможность создать проект Eclipse на основе уже готовых исходных текстов, созданных в других средах разработки? Что для этого необходимо сделать?
5. Для каких целей используется представление Problems?

Глава 2. Отладка и тестирование приложений

Отладка — это процесс пошаговой проверки программ и приложений с целью выявления ошибок.

Процесс отладки характеризуется тем, что программа останавливается каждый раз в точках прерывания. Если вы забыли установить точки прерывания, то отладка не будет отличаться от обычного запуска.

Рассмотрим отладку на примере. Создайте новый класс с именем Debuging и введите предложенный в листинге код.

```
public class Debuging {
  public static void main(String[] args){
    for (int i=0;i<10;i++){
      System.out.print("Шаг "+i+"\n");
    }
  }
}
```

Для запуска отладчика нажмите кнопку с изображением жука. 

При нажатии данной кнопки включается режим отладки текущей программы, если окно кода программы активно. Если активна панель Package Explorer, то отладке подвергнется выбранный (выделенный) класс, при этом класс можно дополнительно выбрать, если их несколько.

На рис. 21 выделено поле, в котором пользователь может установить точки прерывания.

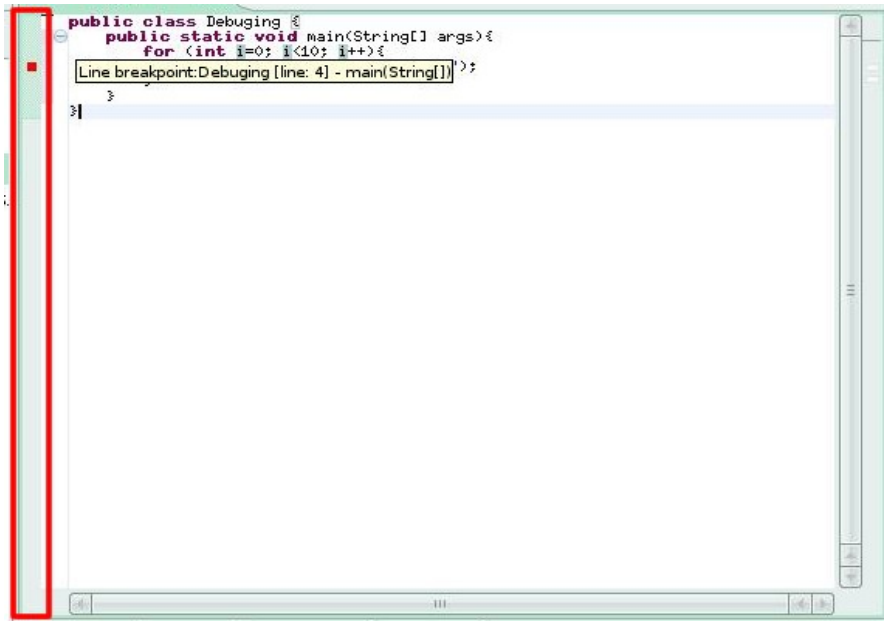


Рис. 21. Линейка точек прерывания

Для того чтобы поставить точку, нужно дважды щелкнуть мышью в данном поле напротив нужной строки, где планируется остановка программы. Маркеры точек прерывания представляют собой небольшие красные квадраты.

Установите точку прерывания напротив строки

```
System.out.print("Шаг "+i+"\n");.
```

После того, как вы установите точку прерывания и нажмете кнопку  , на экране появится диалоговое окно (рис. 22), предлагающее переключить режим рабочего стола из Java в Debug.

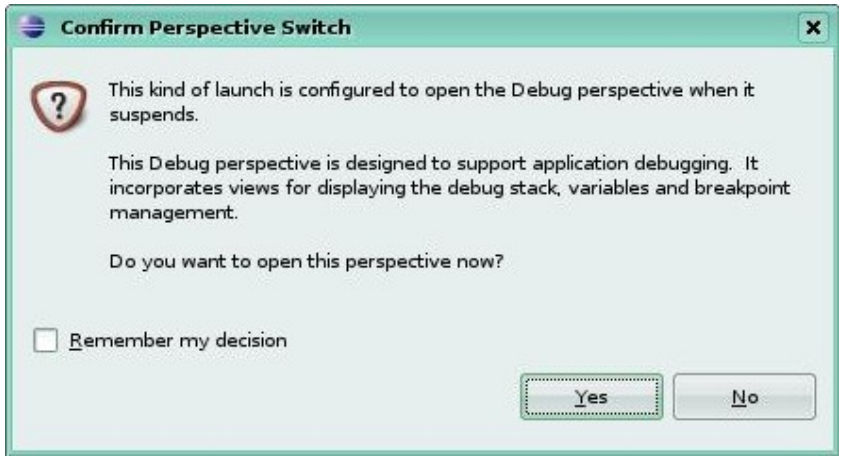


Рис. 22. Предложение переключения рабочего стола

Для того чтобы переключиться обратно в режим Java, используйте кнопку  , находящуюся в верхнем правом углу рабочего стола. Далее, примите данное предложение, нажав кнопку «Yes», и в итоге рабочий стол примет вид, представленный на рис. 23.

Это стандартный, по умолчанию, вид компоновки Debug, но вы можете его изменить, удаляя, добавляя или перетаскивая панели мышью.

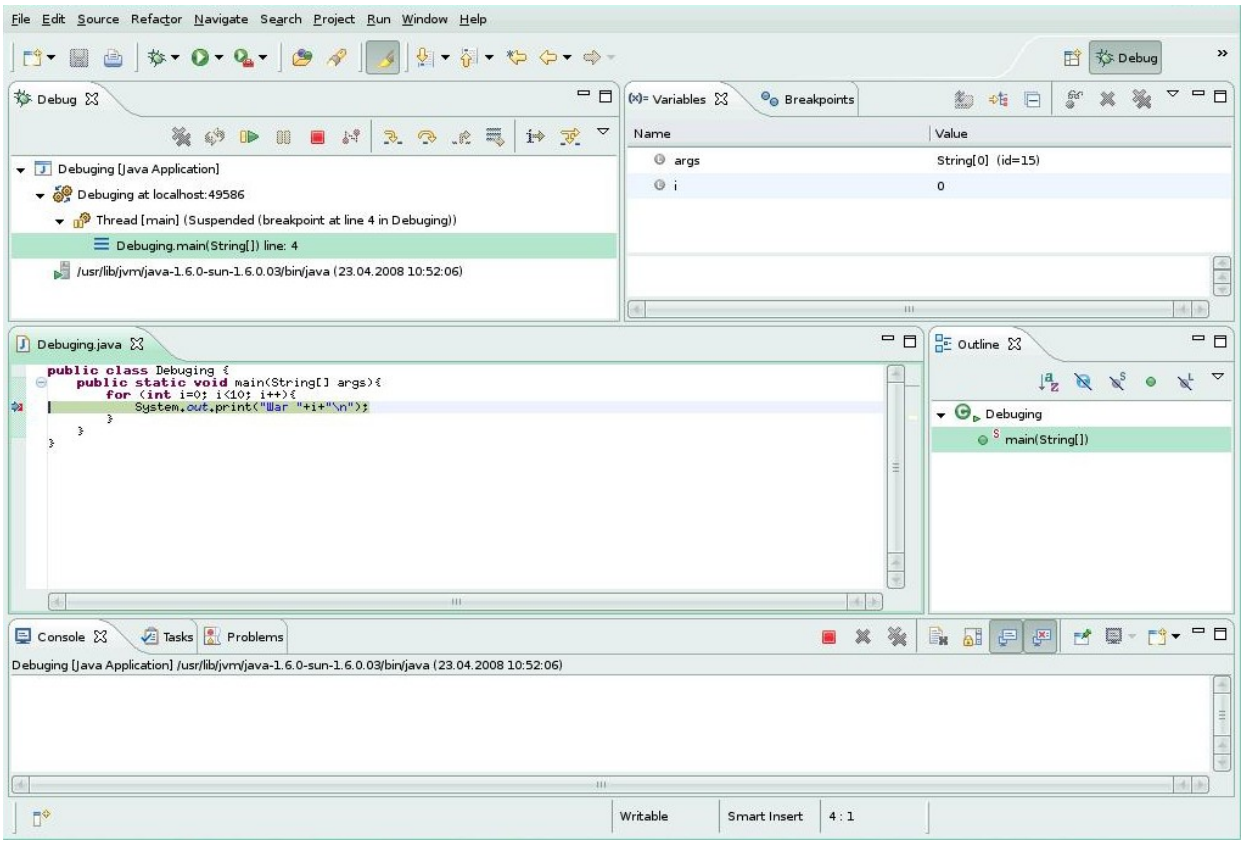


Рис. 23. Рабочий стол Debug

В компоновке Debug можно выделить следующие основные компоненты:

- Окно Debug. В данном окне отображаются задействованные в отладке элементы, а так же панель управления процессом отладки (рис. 24).

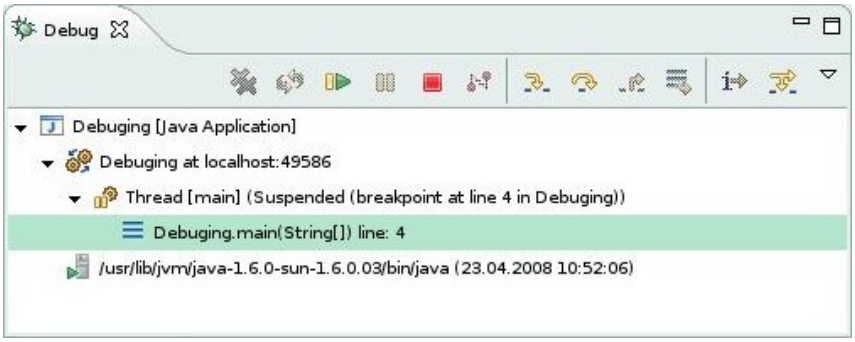


Рис. 24. Окно Debug

- Окно состояния переменных и точек прерывания. Вкладка Variables содержит список переменных, задействованных в текущей точке прерывания. Данное окно является наиболее важным в процессе отладки, оно производит мониторинг переменных на

каждой точке прерывания. Именно с помощью данного окна программист может увидеть поведение объекта, изменение его свойств более тщательно в процессе выполнения программы. Вкладка Breakpoints содержит список точек прерывания, которые установил программист. Точки прерывания можно отключать и включать в процессе отладки, устанавливая или удаляя маркер напротив нужной точки (рис. 25). В левом столбце указаны идентификаторы (имена) переменных, в правом — их текущее значение.

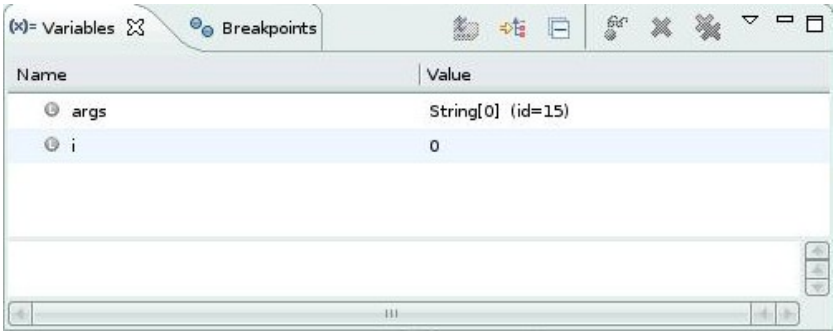


Рис. 25. Окно переменных

- Панель Outline, редактор кода и консоль. Назначение данных элементов рассмотрено ранее.

Теперь программа запущена в режиме отладки. Последовательно выполняя операторы программы, отладчик доходит до первой точки и останавливает ход выполнения. Программист имеет возможность посмотреть значение переменных программы на момент останова. После того как вы проанализируете состояние программы, ее можно продолжать. Нажмите кнопку  («Resume») или клавишу F8 для продолжения.

В нашем случае программа была остановлена на операторе вывода данных на консоль. После продолжения выполнения тело цикла продолжит выполняться, так как в теле нашего цикла только один оператор, он выполнится, и на консоль выведется первая строка, на первом шаге переменная `i=0`, в этом можно убедиться просмотрев вкладку Variables окна состояния (рис. 26).

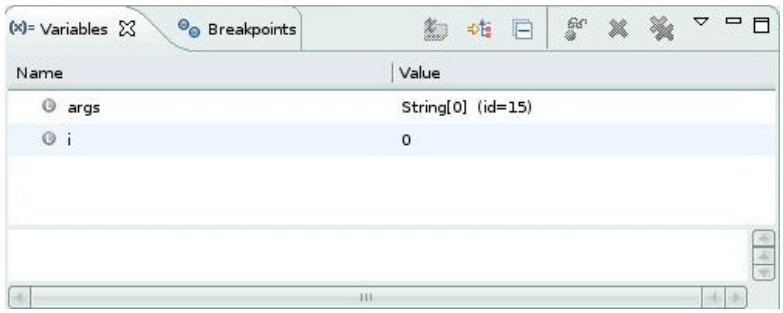


Рис. 26. Состояние переменных на первом шаге

После вывода строки цикл начнет выполняться еще раз, переменная `i` изменится и станет равной 1. При этом в панели Variables она выделится желтым цветом, что сигнализирует о изменении ее значения (рис. 27).

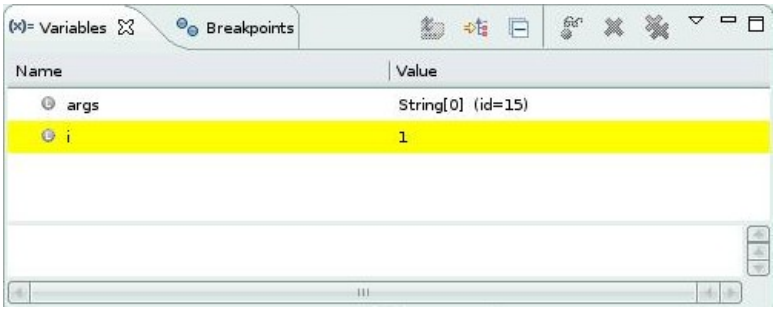


Рис. 27. Состояние переменных на втором шаге

И так далее, нажимая клавишу F8 или кнопку , программа будет переходить от одной точки прерывания к другой. Если точка прерывания установлена на оператор, находящийся в цикле, программа в этом случае будет останавливаться на этой точке столько раз, сколько раз будет проходить цикл.

В отладчике Eclipse есть еще одна полезная функция, с помощью которой можно увидеть значения сложных выражений. Например, нас интересует значение логического выражения (условие), при котором выполняется цикл. Для этого выделите нужный текст, в нашем случае это `i<10` в блоке условия цикла, и нажмите сочетание клавиш <Ctrl>+<Shift>+D, появится всплывающее окно, в котором будет отображен тип выражения (в нашем случае это `boolean`) и его состояние.

Когда программа дойдет до конца, отладка закончится. Для повтора отладки нужно еще раз нажать кнопку .

Для прерывания работы отладчика нажмите кнопку в панели Debug.

Помимо отладки с использованием безусловных точек прерывания, возможно дополнительно установить условия останова на точке прерывания. Это очень полезная особенность отладчика Eclipse, которая позволяет в некоторых случаях значительно упростить процесс отладки.

Для того, чтобы задать условие, при котором будет происходить останов, нужно открыть окно свойств точки прерывания. Для этого наведите курсор мыши на нужную точку прерывания в строке, нажмите правую кнопку мыши, и выберите в раскрывающемся меню пункт Breakpoint Properties, откроется окно свойств выбранной точки прерывания (рис. 28).

Установите галочку на кнопке выбора Enable Condition (Включить Условие) и в поле ввода введите необходимое условие, при котором будет происходить останов на данной точке. Для примера, введите логическое выражение `(i%2)==0`. Данное выражение будет принимать значение «истина» каждый раз, когда остаток от деления `i` на `2` будет равен `0`, то есть каждый раз, когда `i` будет принимать четные значения. Запустите процесс отладки и убедитесь в том, что теперь останов будет происходить каждые два прохода цикла, а не один, как было раньше.

Так же можно выбрать тип условия прерывания — ниже, под полем ввода находится две кнопки выбора — condition is 'true' и value of condition changes. При выборе первого, останов будет происходить каждый раз, когда введенное логическое выражение примет значение «истина», при выборе второго, останов будет происходить каждый раз, когда введенное значение или константа изменит свое значение.

Раскрывающийся список Suspend Policy устанавливает политику останова. При выборе значения Suspend Thread останов будет происходить на уровне текущего потока thread, при выборе Suspend VM — на уровне самой виртуальной машины Java.

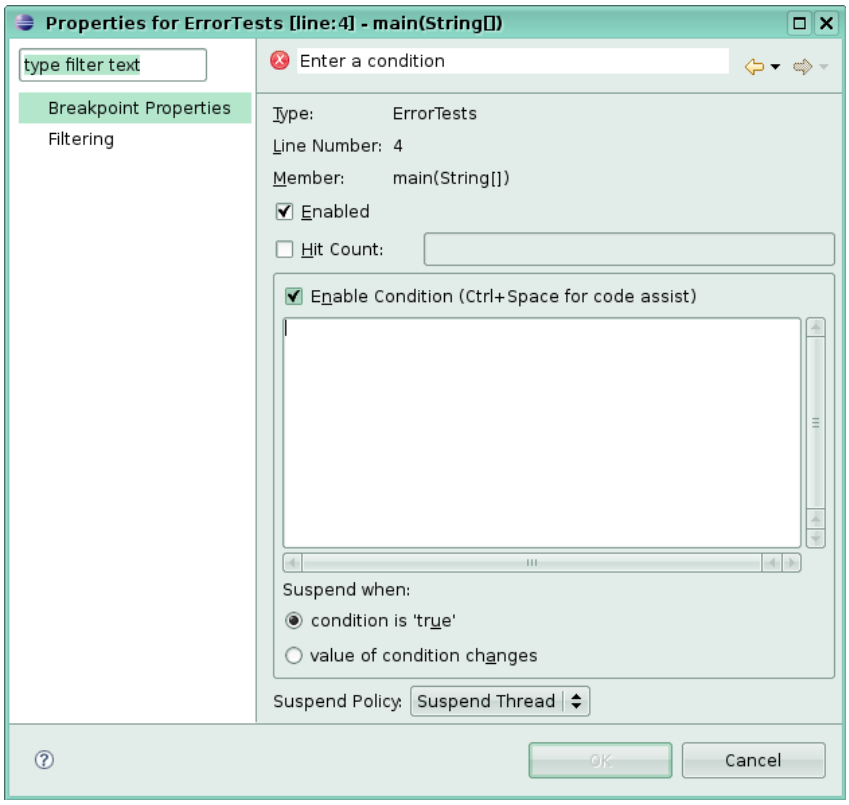


Рис. 28 Свойства точки прерывания

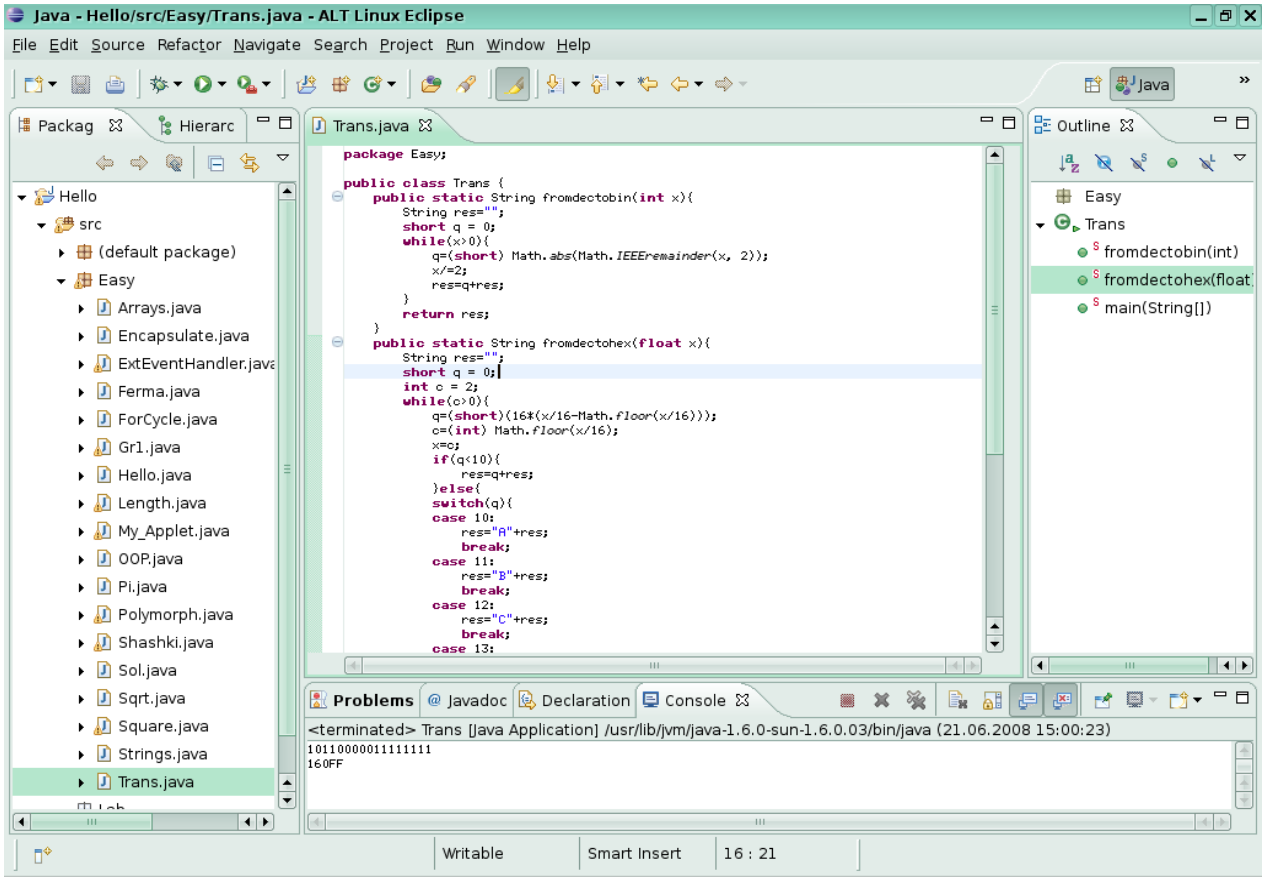
Глава 3. Лабораторный практикум

Лабораторная работа №1

Тема работы. Введение в среду программирования Eclipse.

Цель работы. Начальное знакомство с IDE Eclipse.

Содержание работы. Рассматриваются базовые компоненты интерфейса пользователя Eclipse. Компоновки, представления. Создание проекта, создание и операции с классами.



Задания к работе

1. Установите последовательно компоновки на вашем рабочем столе: Java, Java Browsing, Debug. Укажите функции и назначение этих компоновок.

2. Работа с представлениями. В компоновку Java добавьте новые представления: Problems, Members и затем их закройте. В компоновку Java Browsing добавьте новые представления: Debug, Display, Memory и затем их закройте.

3. Создайте новый проект newProject1. Добавьте в него три пустых класса: wnclass1, wnclass2, wnclass3.

4. Переименуйте классы, созданные в задании 3: fqrs1, fqrs2, fqrs3.

5. Измените место расположение на диске классов поместив их в новый предварительно созданный каталог New.

6. Удалите проект созданный в задании 3.

7. Создайте новый проект Eclipse с именем newClock из примера апплета JDKX.X.X./Demo/Clock.

Лабораторная работа №2

Тема работы. Введение в язык программирования Java.

Цель работы. Первое знакомство с языком программирования Java.

Содержание работы. В данной работе подробно рассматриваются программы, на основе которых учащиеся познакомятся со структурой простейших программ Java.

Программа 1. Простейшая программа.

```
01 public class Hello {
02   public static void main(String[] args) {
03     System.out.print("Добро пожаловать в мир Java!");
04   }
05 }
```

Внимание: с целью облегчения комментирования текста программ практикума в листинги программ добавлена нумерация строк, которую вы не найдете в редакторе кода Eclipse.

Данная программа выводит сообщение на консоль (или в командную строку). Приводится описание класса с именем `Hello` в строке (01), в котором имеется только один метод `main()`(02). При запуске программы управление передается этому методу и сразу же происходит вывод сообщения на консоль.

В самом начале не нужно пытаться понять, как устроен процесс вывода, он достаточно сложный, нужно просто помнить, что вывод производится конструкцией, называемой *объектом стандартного вывода* `System.out` с использованием метода `print`. Метод `println` позволяет после вывода строки осуществить переход курсора на новую строку.

Программа 2. Вывод текста несколькими строками.

```
01 public class Hello2 {
02 // Выполнятся приложение Java начинает с метода main
03   public static void main(String[] args) {
04    System.out.println("Добро\n пожаловать\n в мир\n Java!");
05    } // окончание метода main
06 } // окончание класса Hello2
```

Управляющие Escape последовательности: \n-новая строка, \t-перемещение на следующую позицию табуляции, \r-в начало текущей строки, \\-обратный слеш, \» -двойные кавычки.

Задания к работе

1. Вывести на консоль следующий набор символов:

```
a=1, c=a+1    m=2, n=3  решение уравнений
b=6, d=b*2-1   p=4, g=5  нахождение корней
```

2. Вывести на консоль следующий набор символов:

```
      1
    1 2 3
  1 2 3 4 5
1 2 3 4 5 6 7
```

Лабораторная работа №3

Тема работы. Алгоритмизация и использование управляющих структур в Java.

Цель работы. Использование циклов в приложении Java.

Содержание работы. Рассмотрены программы, в которых используются операторы циклов `For`, `While` для многократного выполнения определенных операторов.

Программа 1. Вычисление суммы и произведения последовательности из 10 случайных чисел.

```
01 public class ForCycle {
02   public static void main(String[] args) {
03     int tmp = 0;
04     long tmp2 = 1;
05     System.out.print("Сумма\t\tПроизведение \n");
06     for(int i=0;i<10;i++){
07       tmp += (int)Math.round(Math.random()*10);
08       tmp2 = tmp2*tmp;
09       System.out.print(tmp+"\t\t"+tmp2+"\n");
10     }
11   }
12 }
```

В строке 02 объявляем главный метод `main()`. В объявлении переменной `tmp` типа `int` присваиваем ей начальное значение (`tmp=0`, инициализация переменной `tmp` в строке 03). В переменной `tmp` будет храниться сумма 10 членов последовательности из случайных чисел. Далее объявляем переменную `tmp2` типа `long`, в которой будем хранить значения произведения членов последовательности (04). В цикле `FOR` (06-10) находится сумма и произведение элементов последовательности. Генерация случайного члена последовательности осуществляется с

использованием стандартной функции `Math.random()`, которая возвращает случайное число в диапазоне от 0 до 1. Далее умножаем случайное число на 10 и округляем `Math.round()` Печать результатов производится в строке 09.

Программа 2. Вычисление квадратного корня числа с помощью итерационной формулы Герона.

```
01 public class Sqrt {
02   static void sqrt(long a){
03     double b=a;
04     int i=0;
05     while ((b*b>a)&&(i<200)){
06       b=(b+a/b)/2;
07       i++;
08     }
09     System.out.print(b);
10   }
11   public static void main(String[] args) {
12     sqrt(45);
13   }
14 }
```

Для вычисления квадратного корня числа используем итерационную формулу Герона $X_{n+1}=\frac{1}{2}(X_n+\frac{a}{X_n})$. Для этого используем цикл `while` с предусловием `((b*b>a)&&(i<200))`.

В строке 02 объявляем новый метод `sqrt()` с одним параметром `a` типа `long`. В объявлении переменной `b` типа `double` присваиваем ей начальное значение (`b=a`, инициализация переменной `b` в строке 03). В переменной `b` будем хранить промежуточные значения корня, вычисляемые в цикле по формуле Герона `b=(b+a/b)/2`. Далее объявляем переменную `i` типа `int` и инициализируем ее значением `0`. Данная переменная понадобится в качестве счетчика цикла.

Далее следует цикл `While` (05-08) с предусловием `((b*b>a)&&(i<200))`. Данное условие состоит из двух условий: квадрат `b` должен быть больше начального значения `a` и значение счетчика не должно превышать 200, то есть всего производится не более 200 итераций. В данном цикле производится вычисление очередного значения `Xn`, которое хранится в переменной `b` (06). Переменная `b` на каждом шаге цикла изменяется, причем новое значение данной переменной зависит от предыдущего значения. В конце цикла увеличиваем счетчик `i` на 1 в строке 07.

В конце метода выводим полученное значение `b` в консольное окно.

В методе `main()` вызывается метод `sqrt(45)`. В качестве параметра можно взять произвольное целое число (45), которое будет присвоено переменной `a`. (Помните, вызов каких-либо методов напрямую из метода, имеющего модификатор `static`, возможен только при условии, что вызываемый метод тоже является статическим).

Задания к работе

1. С помощью цикла вычислите значение выражения 2^n.
2. Составьте программу расчета факториала для произвольного числа `n<10`.
3. Даны два действительных числа. Необходимо получить их сумму, разность и произведение. Результат вывести на консоль.
4. Определить время свободного падения материального тела с заданной высоты `h`. Результат вывести в консоль.
5. Составьте программу для нахождения длины катета прямоугольного треугольника (`b`), если известны длины гипотенузы (`c`) и катета (`a`). Результат вывести на консоль.

Лабораторная работа №4

Тема работы. Алгоритмизация и использование управляющих структур в Java.

Цель работы. Использование элементов организации ветвления в Java.

Содержание работы. В данной работе рассматриваются программы, использующие операторы ветвления. Ветвление используется для организации различных направлений вычислительного процесса.

Программа 1. Вычисления числа Пи.

```
01 public class Pi {
02   static double pi;
04   static void leibnic(){
05     for(double i=1;i<100000000;i+=1){
06       if (i%2==0){
07         pi-=1/(2*i-1);
09       }else{
10         pi+=1/(2*i-1);
12       }
13     }
14     pi*=4;
15     System.out.print("\nЧисло Пи, подсчитанное по методу
Лейбница равно: "+pi);
16   }
17   static void vallis(){
18     double pi1=1, pi2=1;
19     for(double i=2;i<100000000;i+=2){
```

```
20        pi1*=i/(i+1);
21        pi2*=i/(i-1);
22      }
23      pi=pi1*pi2*2;
24      System.out.print("\nЧисло Пи, подсчитанное по методу Валлиса
равно: "+pi);
25    }
26    public static void main(String[] args) {
27      leibnic();
28      vallis();
29    }
30  }
```

В данной программе описаны два метода нахождения числа Пи: метод Лейбница (ряд Лейбница) и метод Валлиса (формула Валлиса).

В данной программе в строках 02 и 03 описываются `static` переменная класса `pi` типа `double`. Расчеты методом Лейбница вызываются в строке 27 благодаря использованию имени метода `leibnic()`. Аналогично в строке 28 вызываем метод Валлиса `vallis()`.

В методе Лейбница для вычисления числа Пи используется ряд:

$$\frac{1}{1}-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}\ldots=\frac{\pi}{4}$$

Для вычисления данного ряда мы использовали цикл с условным оператором, где проверяется условие на четность `i%2==0`, если оно верно, то дробь имеет положительный знак `pi+=1/(2*i-1)`, если нет, то ставим знак минус. В конце, для получения приближенного значения числа Пи, переменная `pi` умножается на 4 и выводится на консоль.

Расчет по формуле Валлиса

$$\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\frac{8}{7}\cdot\frac{8}{9}\ldots=\frac{\pi}{2}$$

происходит немного по-другому. В методе объявляются две переменные `pi1` и `pi2` типа `double` в строке 18, первая хранит произведение четных дробей, вторая — нечетных, затем их удвоенное произведение присваивается переменной `pi` и выводится на консоль.

Программа 2. Нахождение решения уравнения $A^3+B^3+C^3=\overline{ABC}$

```
01  public class Sol {
02    public static void work(){
03      long x=0;
04      for(int i=1;i<10;i++){
05      for (int j=0;j<10;j++){
06      for (int k=0;k<10;k++){
07      x=i*100+j*10+k;
08      if (x==(Math.pow(i,3)+Math.pow(j, 3) + Math.pow(k,3))){
09
    System.out.println(i+"^3+"+j+"^3+"+k+"^3+"+" = "+x);
```

```
10      }
11      }
12      }
13      }
14    }
15    public static void main(String[] args) {
16      work();
17    }
18  }
```

Данная программа находит такие цифры A, B и C, сумма кубов которых равна числу, составленному из этих цифр.

В программе имеются три цикла, вложенные друг в друга, каждый цикл выполняется десять раз, таким образом тело цикла 07-09 выполняется одну тысячу раз (10x10x10). Переменная `x`, составленная из кубов счетчиков циклов, где первый счетчик играет роль сотен числа, второй десятков, третий единиц. Если полученное `x` равно сумме кубов счетчиков, то на консоль выводится информация о полученных цифрах и числе `x`. Нахождение кубов чисел осуществляется с помощью метода `pow()` класса `Math`, в качестве первого параметра выступает само число, а вторым параметром служит показатель степени `Math.pow(i,3)`. Можно использовать более знакомую и привычную запись `i*i*i`, `j*j*j` и `k*k*k`.

Задания к работе

1. Создайте приложение, которое покажет, что для выражения $a^n+b^n=c^n$ (теорема Ферма) нет натуральных решений от `1` до `100` и `n>2`. Убедитесь, что есть решения для `n=2`, и выведите их в консоль.

2. Вычислить выражение $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+...+\frac{1}{9999}-\frac{1}{10000}$, используя оператор условия.

Лабораторная работа №5

Тема работы. Алгоритмизация и использование управляющих структур в Java.

Цель работы. Организация ветвления в Java.

Содержание работы. Рассматривается программа, в которой используется оператор множественного выбора, выполняющий задачу множественного ветвления.

Программа 1. Перевод чисел из десятичной системы счисления в двоичную и шестнадцатеричную.

```
01  public class Trans {
02    public static String fromdectobin(int x){
03      String res="";
```

```
04      short q = 0;
05      while(x>0){
06        q=(short)(x%2);
07        x/=2;
08        res=q+res;
09      }
10      return res;
11    }
12    public static String fromdectohex(float x){
13      String res="";
14      short q = 0;
15      int c = 2;
16      while(c>0){
17        q=(short)(16*(x/16-Math.floor(x/16)));
18        c=(int) Math.floor(x/16);
19        x=c;
20        if(q<10){
21          res=q+res;
22        }else{
23        switch(q){
24        case 10:
25          res="A"+res;
26          break;
27        case 11:
28          res="B"+res;
29          break;
30        case 12:
31          res="C"+res;
32          break;
33        case 13:
34          res="D"+res;
35          break;
36        case 14:
37          res="E"+res;
38          break;
39        case 15:
40          res="F"+res;
41          break;
42        }
43        }
44      }
45      return res;
46    }
47    public static void main(String[] args) {
48      System.out.println(fromdectobin(90367));
49      System.out.println(fromdectohex(90367));
50    }
51  }
```

В данной программе приведены два метода, один переводит десятичное число в двоичную систему счисления, второй — в шестнадцатеричную.

Для перевода десятичного числа в двоичное представление используется стандартный метод — деление исходного числа на 2 с сохранением остатка, в нашем примере остаток прибавляется к строке `res` с левой стороны (08). В данном методе используется цикл с предусловием `while()`, который каждый раз получает остаток от деления на 2, затем делит исходное число на 2 и заносит полученный в первом действии остаток слева в строку. В конце, после цикла, метод возвращает полученную строку.

Способ перевода из десятичного в шестнадцатеричное представление использует аналогичный прием, производя последовательно деление на 16, за одним исключением — остаток в нашей программе вычисляется с помощью функции `floor()`, которая возвращает целую часть дробного числа. В результате получим целочисленный остаток от деления. Процесс записи остатков в строку более сложный, чем в случае двоичной системы счисления, так как в шестнадцатеричной для представления числа недостаточно базового набора цифр от 0 до 9 и необходимо использовать буквы A-F. Для того чтобы в строку результата добавлять буквы, в программе используется оператор множественного выбора `switch()`, в котором производится проверка остатка. В зависимости от величины остатка, в строку результата добавляется необходимая буква (строки 23-42).

Методы `fromdectobin`, `fromdectohex` возвращают значения типа `String`, которые можно непосредственно использовать в качестве аргумента метода `print()` для вывода на консоль (строки 48, 49). В качестве аргументов этих методов берутся числа, которые требуется перевести в заданную систему счисления.

Задания к работе

1. Для произвольной цифры от 0 до 9 вывести на консоль ее значение прописью. Например, для цифры 9 на консоли должна быть напечатана строка «Девять».

Лабораторная работа №6

Тема работы. Массивы и строки.

Цель работы. Научиться использовать массивы и строки в Java.

Содержание работы. Работа с массивами: поиск минимального элемента и сортировка массива, а также простейшие приемы работы со строками.

Программа 1. Работа с массивами

```
01 public class Arrays {
02   public static short minX(){
03     short x=0;
04     short[] array = new short[10];
05     System.out.print("Исходный массив: ");
06     for (int i=0;i<10;i++){
07     array[i]=(short)Math.round(50*Math.random());
08       System.out.print(array[i]+", ");
09     }
10     x=array[0];
11     for(int i=1;i<10;i++){
12       if(x>array[i]){
13         x=array[i];
14       }
15     }
16     return x;
17   }
18   public static void sort(){
19     short temp;
20     short[] array = new short[10];
21     System.out.print("\nИсходный массив: ");
22     for (int i=0;i<10;i++){
23     array[i]=(short)Math.round(50*Math.random());
24       System.out.print(array[i]+", ");
25     }
26     System.out.print("\nМассив отсортированный: ");
27     for(int i=0;i<9;i++){
28       for(int j=9;j>i;j--){
29         if(array[j-1]>array[j]){
30           temp=array[j];
31           array[j]=array[j-1];
32           array[j-1]=temp;
33         }
34       }
35       System.out.print(array[i]+", ");
36     }
37 }
38 public static void main(String[] args) {
39 System.out.print("\nМинимальный элемент: "+minX());
40 sort();
41 }
```

```
42 }
```

В данной программе используются два метода — `minX()` и `sort()`. В каждом из представленных методов описывается по одному массиву, выделяя память на десять элементов для каждого массива (04, 20). Затем происходит заполнение массивов случайными числами с помощью метода `Math.random()` (06-09, 22-25). Явным преобразованием `(short)` приводим полученные значения типа `double` к значениям типа `short` (7,23).

После того, как массивы готовы, можно производить над ними различные действия. Метод `minX()` ищет минимальный элемент в полученном массиве (11-15).

Второй метод производит сортировку полученного массива (27-34) методом пузырька. Затем выводит значения его элементов (35).

В методе `main()` вызываем `minX()` и `sort()`. Поскольку метод `minX()` возвращает значение строкового типа, его можно использовать в качестве аргумента конструкции `System.out.print()`.

Программа 2. Работа со строками

```
01 public class Strings {
02   public static String compare(String s1, String s2){
03     String s3="";
04     if (s1.equals(s2)){
05       s3="Строки \""+s1+"\" и \""+s2+"\" равны";
06     } else {
07       s3="Строки \""+s1+"\" и \""+s2+"\" не равны";
08     }
09     return s3;
10   }
11   public static String add(String s1, String s2){
12     System.out.print("\nРезультат сложения строк \""+s1+"\" и
"+"\""+s2+"\": ");
13     s1+=" "+s2;
14     return s1;
15   }
16   public static void main(String[] args) {
17     System.out.println(compare("АБВГ","АБВ"));
18     System.out.print(compare("АБВ","АБВ"));
19     System.out.print(add("Hello","World"));
20   }
21 }
```

В приложении имеется два метода, которые сравнивают и склеивают две строки. Первый метод `compare()` производит сравнение двух строк и выводит результат на консоль. Сравнение производится с помощью функции `equals()` в строке (04). Если строки совпадают, данная функция возвращает значение `true`. Если строки равны (имеют одина-

ковый набор символов), выводится соответствующее сообщение (05), иначе сообщение (07). В конструкции присвоения строке `s3` (05, 07) были использованы символы Escape-последовательности, приведенные в лабораторной работе №2.

Во втором методе происходит простое склеивание строк с помощью операции «+»(13).

Задания к работе

1. Дан массив из целых чисел `A(n)`, где `n=1,25`. Необходимо поменять местами его максимальный и минимальный элемент.
2. Дан массив из целых чисел `B(n)`, где `n=1,25`. Необходимо упорядочить массив по возрастанию.
3. Дан массив из целых чисел `C(n)`, где `n=1,20`. Необходимо найти среднее значение и вывести его на консоль.
4. Дан массив из целых чисел `D(n)`, где `n=1,30`. Посчитайте сумму четных и нечетных элементов массива.
5. Напишите программу, выводящую на консоль таблицу `3x5` случайных элементов `(a(i,j)< 10)`.
6. Измените программу 1 так, чтобы она выводила на консоль еще и максимальный элемент (с помощью описания нового метода, например `maxX()`).
7. Даны 5 строк `s1`, `s2`, `s3`, `s4` и `s5`, на основе условия: если строка `s4` равна строке `s5`, нужно сложить строки `s1` и `s2`, иначе нужно сложить строки `s1` и `s3`.

Лабораторная работа №7

Тема работы. Введение в объектно-ориентированное программирование.

Цель работы. Изучение основ объектно-ориентированного программирования на языке Java.

Содержание работы. В работе рассматриваются приложения, демонстрирующие основные принципы объектно-ориентированного программирования.

Программа 1. Инкапсуляция полей и наследование.

```
01 public class Encapsulate {
02   public int field1 = 100;
03   protected int field2 = 150;
04   int field3 = 200;
05   private int field4 = 250;
06   public int getField4(){
07     return this.field4;
```

```
08    }
09    public static void main(String[] args) {
10      new Class2().method1();
11    }
12  }
13  class Class2 extends Encapsulate {
14    public void method1(){
15      System.out.println("Значение открытого public поля
- field1: "+this.field1);
16      System.out.println("Значение защищенного protected
поля - field2: "+this.field2);
17      System.out.println("Значение поля без модификатора
- field3: "+this.field3);
18      System.out.println("Значение инкапсулированного закрытого
private поля - field4: "+getField4());
19    }
20  }
```

В приведенном примере демонстрируются два основных принципа объектно-ориентированного программирования: инкапсуляция и наследование. Создается класс `Encapsulate` (01-12), в котором объявляются и инициализируются поля `filed1`, `filed2`, `filed3` и `filed4`. Каждое поле имеет различный уровень доступа, поскольку при описании использованы модификаторы доступа к элементам класса: `public`, `protected`, `private`. Затем описывается новый класс `Class2`, который наследуется от класса `Encapsulate` с помощью зарезервированного слова `extends`. В данном классе описывается новый метод, расширяющий суперкласс, в котором выводятся значения полей, описанных в суперклассе. Вы видите, что в классе `Class2` нет описания этих полей, они наследуются из суперкласса `Encapsulate`, так же как и метод `getField4()`. Поскольку метод `main()` находится в классе `Encapsulate`, то для проверки работы метода `method1()` нужно создать экземпляр класса `Class2` и вызвать этот метод, что и происходит в строке 10.

Интересно отметить особенность инкапсуляции, когда поле `filed4`, объявленное с ключевым словом `private`, не доступно в классах потомках, хотя и наследуется. Для проверки этого факта измените строку 18, заменив вызов `getField4()` на прямой запрос значения поля `this.filed4`. В данном случае Eclipse сообщит об ошибке, а именно, что поле `Encapsulate.field4` не доступно. Принцип инкапсуляции применяется для сокрытия переменных или методов класса от внешних клиентов (классов, методов).

Программа 2. Полиморфизм.

```
01  public class Polymorph {
02    void method1(){
03      System.out.println("Это был вызван метод суперкласса
```

```
Polymorph");
04    }
05    public static void main(String[] args) {
06      new NewClass1().method1();
07      new NewClass2().method1();
08    }
09  }
10  class NewClass1 extends Polymorph {
11    void method1(){
12      System.out.println("Это был вызван метод класса потомка
13  NewClass1");
14    }
15  }
16  class NewClass2 extends Polymorph {
17    void method1(){
18      System.out.println("Это был вызван метод класса потомка
    NewClass2");
19    }
20  }
```

В приложении демонстрируется простейший пример полиморфизма. Имеется три класса `Polymorph`, `NewClass1`, `NewClass2`, причем `Polymorph` является суперклассом по отношению к `NewClass1`, `NewClass2`. В классе `Polymorph` описан метод `method1()` (02), который выводит сообщение в консоль. Классы `NewClass1`, `NewClass2` наследуют этот метод и переопределяют его (override), осуществляя тем самым перегрузку метода. Далее в методе `main()` суперкласса создается два объекта экземпляра классов.

В результате оба класса наследуют один и тот же метод, но выполняется он у каждого из них по-разному или в разной форме. Таким образом, вызывая один и тот же метод, с помощью разных объектов можно получить разные по форме результаты, тогда более понятным становится смысл термина «полиморфизм».

Задания к работе

1. Создайте приложение, в котором имеются три класса: `fclass1`, `fclass2`, `fclass3`. В классе `fclass1` содержится метод `main()`. Класс `fclass3` наследуется от `fclass1`, а `fclass2` — от `fclass3`.

2. Создайте приложение, в котором имеются два класса: `fclass1`, `fclass2`. В классе `fclass1` содержится метод `main()`. Создайте методы в `fclass2` для доступа к закрытым переменным класса `fclass1`.

Лабораторная работа №8

Тема работы. Работа с графическим интерфейсом пользователя. Обработка событий.

Цель работы. Знакомство с элементами разработки графического интерфейса пользователя. Создание приложения из нескольких файлов.

Содержание работы. В данной работе рассматриваются программы, демонстрирующие основы работы с графическим интерфейсом пользователя и обработкой событий компонентов ГИП.

В данной работе для создания программы графопостроителя нам понадобится четыре класса, которые будут находиться в разных файлах. Для того, чтобы в дальнейшем избежать путаницы, необходимо данные файлы поместить в отдельную папку (или, в интерпретации платформы Eclipse, в отдельный подпакет). Создайте новую папку, щелкнув правой кнопкой мыши на имени проекта в представлении Package explorer, и в раскрывшемся меню (рис. 15), выбрав пункт «Folder». Назовите ее «MyGraph». Затем в данной папке аналогично создайте четыре новых класса:

- «Graphic» — главный класс, в котором будет метод `main()`.
- «Sinus» — класс графика синуса.
- «Cosinus» — класс графика косинуса.
- «X2» — класс графика параболы.

Помните, если классы описаны с любым из модификаторов видимости `public`, `protected` или `private`, то имена классов должны совпадать с именами файлов, в которых хранятся данные классы.

После того, как классы будут созданы, введите предложенный ниже код программы.

Программа 1. Построение графиков элементарных функций.

Класс Graphic

```
01 package MyGraph;
02 import java.awt.*;
03 import java.awt.event.*;
04 import javax.swing.*;
05 public class Graphic extends JFrame{
06   Graphic(String s){
07     super(s);
08     setLayout(null);
09     setSize(120,200);
10     setVisible(true);
11     this.setDefaultCloseOperation(EXIT_ON_CLOSE);
12     this.setResizable(false);
13     Button sin = new Button("Sin");
14     sin.setBounds(5, 20, 100, 25);
```

```
15      add(sin);
16      Button cos = new Button("Cos");
17      cos.setBounds(5, 70, 100, 25);
18      add(cos);
19      Button x2 = new Button("Парабола");
20      x2.setBounds(5, 120, 100, 25);
21      add(x2);
22      sin.addActionListener(new ActionListener(){
23        public void actionPerformed(ActionEvent event){
24          new Sinus("Синус");
25        }
26      });
27      cos.addActionListener(new ActionListener(){
28        public void actionPerformed(ActionEvent event){
29          new Cosinus("Косинус");
30        }
31      });
32      x2.addActionListener(new ActionListener(){
33        public void actionPerformed(ActionEvent event){
34          new X2("Парабола");
35        }
36      });
37    }
38    public static void main(String[] args) {
39      new Graphic("Графопостроитель");
40    }
41  }
```

Класс Sinus

```
01  package MyGraph;
02  import java.awt.Color;
03  import java.awt.Graphics;
04  import javax.swing.JFrame;
05  public class Sinus extends JFrame{
06    Sinus(String s){
07      super(s);
08      setLayout(null);
09      setSize(600,300);
10      setVisible(true);
11      this.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
12      this.setResizable(false);
13      this.setLocation(100, 100);
14    }
15    public void paint(Graphics gr){
16      int y; int j=0; int k=0;
17      gr.setColor(Color.WHITE);
18      gr.fillRect(0, 0, 600, 300);
19      gr.setColor(Color.lightGray);
20      while(j<600){
```

```
21        gr.drawLine(j, 0, j, 300);
22        j+=30;
23      }
24      while(k<300){
25        gr.drawLine(0, k, 600, k);
26        k+=30;
27      }
28      gr.setColor(Color.BLACK);
29      gr.drawLine(300, 0, 300, 300);
30      gr.drawLine(0, 150, 600, 150);
31      gr.drawLine(120, 140, 120, 160);
32      gr.drawLine(480, 140, 480, 160);
33      gr.drawString("0", 305, 165);
34      gr.drawString("-"+"\u03c0", 125, 140);
35      gr.drawString("\u03c0", 485, 140);
36      gr.setColor(Color.RED);
37      for(double i=0;i<1000;i++){
38        y=(int)(80*Math.sin(Math.PI*i/180));
39        gr.drawLine((int)i-240, y+150, (int)i-240, y+150);
40      }
41      gr.dispose();
42    }
43  }
```

Класс Cosinus

```
01  package  MyGraph;
02  import java.awt.Color;
03  import java.awt.Graphics;
04  import javax.swing.JFrame;
05  public class Cosinus extends JFrame{
06    Cosinus(String s){
07      super(s);
08      setLayout(null);
09      setSize(600,300);
10      setVisible(true);
11      this.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
12      this.setResizable(false);
13      this.setLocation(200, 200);
14    }
15    public void paint(Graphics gr){
16      int y; int j=0; int k=0;
17      gr.setColor(Color.WHITE);
18      gr.fillRect(0, 0, 600, 300);
19      gr.setColor(Color.lightGray);
20      while(j<600){
21        gr.drawLine(j, 0, j, 300);
22        j+=30;
23      }
24      while(k<300){
```

```
25        gr.drawLine(0, k, 600, k);
26        k+=30;
27      }
28      gr.setColor(Color.BLACK);
29      gr.drawLine(300, 0, 300, 300);
30      gr.drawLine(0, 150, 600, 150);
31      gr.drawLine(120, 140, 120, 160);
32      gr.drawLine(480, 140, 480, 160);
33      gr.drawString("0", 305, 165);
34      gr.drawString("-"+"\u03c0", 125, 140);
35      gr.drawString("\u03c0", 485, 140);
36      gr.setColor(Color.RED);
37      for(double i=0;i<1000;i++){
38        y=(int)(80*Math.cos(Math.PI*i/180));
39        gr.drawLine((int)i-240, y+150, (int)i-240, y+150);
40      }
41      gr.dispose();
42    }
43  }
```

Класс X2

```
01  package MyGraph;
02  import java.awt.Color;
03  import java.awt.Graphics;
04  import javax.swing.JFrame;
05  public class X2 extends JFrame{
06    X2(String s){
07      super(s);
08      setLayout(null);
09      setSize(600,300);
10      setVisible(true);
11      this.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
12      this.setResizable(false);
13      this.setLocation(300, 300);
14    }
15    public void paint(Graphics gr){
16      int y; int j=0; int k=0;
17      gr.setColor(Color.WHITE);
18      gr.fillRect(0, 0, 600, 300);
19      gr.setColor(Color.lightGray);
20      while(j<600){
21        gr.drawLine(j, 0, j, 300);
22        j+=50;
23      }
24      while(k<300){
25        gr.drawLine(0, k, 600, k);
26        k+=50;
27      }
28      gr.setColor(Color.BLACK);
```

```
29      gr.drawLine(300, 0, 300, 300);
30      gr.drawLine(0, 150, 600, 150);
31      gr.drawString("0", 305, 165);
32      gr.setColor(Color.RED);
33      for(double i=0;i<1000;i++){
34        y=-(int)(i*i/300)+150;
35        gr.drawLine((int)i+300, y, (int)i+300, y);
36        gr.drawLine(-(int)i+300, y, -(int)i+300, y);
37      }
38      gr.dispose();
39    }
40  }
```

В классе Graphics описано окно размером 120x200, в котором расположены три кнопки. Нажатие кнопки обслуживается (обрабатывается) специальной процедурой, которая называется процедурой обработки события `ActionEvent`. При нажатии кнопки происходит создание новой формы, в которой будет построен соответствующий график элементарной функции (22-26, 27-31, 32-36).

Классы обработчиков событий описаны внутри параметра метода добавления слушателя `addActionListener()` (22, 27, 32). Интересно, что параметром этого метода является новый объект интерфейса `ActionListener` с одним методом `actionPerformed()`, который и отвечает за поведение программы, в случае если будет вызвано данное событие (нажатие на кнопки).

В классах, которые производят построение графиков, в строках 20-27 с использованием двух циклов происходит построение координатной сетки светло-серым цветом, затем черным цветом строятся оси абсцисс и ординат.

Наибольший интерес представляют строки 37-40 (для синуса и косинуса) и 33-37 (для параболы) в которых происходит рисование графиков. Синус и косинус рисуются по аналогии: вначале вычисляется значение `y` и методом `drawString()` рисуется «линия» длиной в один пиксел. Координата `y` вычисляется путем явного преобразования типа `double` в тип `int`, полученного в результате выполнения выражения `80*Math.cos(Math.PI*i/180)`.

Парабола строится аналогичным способом, в два этапа. На первом этапе строится положительная часть параболы и далее отрицательная.

Программа 2. Надписи на кнопках.

```
01  import java.awt.*;
02  import java.awt.event.*;
03  import javax.swing.*;
04  public class ExtEventHandler extends JFrame {
```

```
05    ExtEventHandler(String s){
06      super(s);
07      setLayout(null);
08      setSize(100,200);
09      setVisible(true);
10      setResizable(false);
11      setDefaultCloseOperation(EXIT_ON_CLOSE);
12      Button b1 = new Button("Первая кнопка");
13      b1.setBounds(2, 5, 96, 22);
14      add(b1);
15      Button b2 = new Button("Вторая кнопка");
16      b2.setBounds(2, 100, 96, 22);
17      add(b2);
18      b1.addActionListener(new Handler(b1, b2));
19      b2.addActionListener(new Handler(b1, b2));
20    }
21    public static void main(String[] args) {
22      new ExtEventHandler("");
23    }
24  }
25  class Handler implements ActionListener{
26    private Button ba;
27    private Button bb;
28    String temp;
29    Handler(Button b1, Button b2){
30      this.ba=b1;
31      this.bb=b2;
32    }
33    public void actionPerformed(ActionEvent e) {
34      temp = ba.getLabel();
35      ba.setLabel(bb.getLabel());
36      bb.setLabel(temp);
37    }
38  }
```

В классе `ExtEventHandler` (04-24) приводится основное описание интерфейса пользователя: форма размером 100x200 и две кнопки с надписями: первая кнопка (12-14) и вторая кнопка (15-17).

В приложении для обработки событий, которые возникают при нажатии на кнопки, описан отдельный класс `Handler` (25-38), который использует интерфейс `ActionListener`.

Поскольку в описании `ActionListener` нельзя напрямую использовать компоненты, описанные в конструкторе класса `ExtEventHandler`, то нужно создать ссылки на объекты, которые будут использованы. Создание ссылочных объектов производится в строках 26, 27, а сами ссылки создаются в конструкторе (30, 31). Когда ссылки готовы, в процессе добавления слушателей (18, 19), в параметрах указываем имена реальных компонентов.

Таким образом, при нажатии на любую из двух кнопок меняются местами их надписи (34-36).

Задания к работе

1. Создайте форму размером 500x500 и нарисуйте на ней домик.

2. Сумматор. Создайте приложение, которое представляет собой форму 200x150. На форме разместите три текстовых поля и одну кнопку с надписью «Расчет». При нажатии на кнопку значения, введенные в первые два текстовых поля, складываются и результат записывается в третье поле.

Лабораторная работа №9

Тема работы. Апплеты.

Цель работы. Знакомство с элементами разработки Java-апплетов.

Содержание работы. Разработка апплетов.

Программа 1. Простейший апплет.

```
01 import java.awt.*;
02 import java.applet.*;
03 public class My_Applet extends Applet{
04   public void paint(Graphics g){
05     g.drawString("Hello, World!", 5, 30);
06   }
07 }
```

Приложение представляет собой простейший Java-апплет, в котором выводится строка с использованием ранее рассмотренного метода (05) `drawString()`.

Для запуска и просмотра апплетов в Eclipse используется специальное программное средство AppletViewer.

Программа 2. Использование элементов графического интерфейса пользователя в апплетах.

```
01 import java.awt.*;
02 import java.awt.event.ActionEvent;
03 import java.awt.event.ActionListener;
04 import java.applet.*;
05 public class My_Applet extends Applet{
06   public void init(){
07     setLayout(null);
08     final TextField tf1 = new TextField(15);
09     tf1.setBounds(1, 1, 100, 20);
10     add(tf1);
11     final TextField tf2 = new TextField(15);
12     tf2.setBounds(1, 25, 100, 20);
13     add(tf2);
```

```
14      final TextField tf3 = new TextField(15);
15      tf3.setBounds(1, 50, 100, 20);
16      add(tf3);
17      Button b1 = new Button("Кнопка 1");
18      b1.setBounds(1, 100, 100, 20);
19      add(b1);
20      b1.addActionListener(new ActionListener(){
21        public void actionPerformed(ActionEvent event){
22          try{
23    tf3.setText(String.valueOf((Long.valueOf(tf1.getText())
+Long.valueOf(tf2.getText()))));
24          }catch(Exception e){
25            tf3.setText("Введите числа");
26          }
27        }
28      });
29    }
30  }
```

В данной программе описан апплет, в котором расположены три текстовых поля и одна кнопка. При нажатии на кнопку введенные численные значения первых двух полей складываются и результат записывается в третье поле (вторая задача из лабораторной работы №8).

В отличие от обычных приложений Java, в апплетах инициализация и добавление компонентов графического интерфейса пользователя осуществляется не в конструкторе, а в методе `init()`.

Задания к работе

1. Выполните задание 1 из лабораторной работы №8 в форме апплета.
2. Создайте апплет, в котором при нажатии на кнопку выводится матрица 3x3. Используйте метод `drawString()`.

Лабораторная работа №10

Тема работы. Приложение «Калькулятор».

Цель работы. Знакомство со сложным приложением Java.

Содержание работы. Анализ и подробный разбор кода. Запуск и тестирование приложения.

Программа 1. Калькулятор.

```
01  import java.awt.*; // Импорт пакета awt
02  import java.awt.event.*; // Импорт пакета обработки событий
03  import javax.swing.*; // Импорт пакета swing
04  public class Calc extends JFrame{ // Начало класса Calc, наслед.
от класса JFrame
05  double temp = 0; // Объявление переменной temp типа double
06      Char op = ''; // Объявление переменной op типа char
07      Calc(String s){ // Начало определения конструктора Calc
```

```
08        super(s); // Вызов конструктора суперкласса (JFrame)
09        setLayout(null); // Отказ от менеджера размещения
10        setSize(250,250); // Установка размеров окна
11        setVisible(true); // Установка видимости на экране
12        this.setDefaultCloseOperation(EXIT_ON_CLOSE); //
Установка по умолчанию для закрытия окна
13        final TextField display = new TextField(""); // Создание
объекта типа TextField с именем display (Поле ввода)
14        display.setEditable(false); // Запрещение редактирования
15        display.setBounds(2, 2, 238, 22);// Установка
расположения и размеров поля ввода
16        add(display); // Добавление поля ввода в контейнер окна
17        Button b1 = new Button("1");
18        b1.setBounds(2,30,40,40);
19        add(b1);
20        Button b2 = new Button("2");
21        b2.setBounds(52,30,40,40);
22        add(b2);
23        Button b3 = new Button("3");
24        b3.setBounds(102,30,40,40);
25        add(b3);
26        Button b4 = new Button("4");
27        b4.setBounds(2,80,40,40);
28        add(b4);
29        Button b5 = new Button("5");
30        b5.setBounds(52,80,40,40);
31        add(b5);
32        Button b6 = new Button("6");
33        b6.setBounds(102,80,40,40);
34        add(b6);
35        Button b7 = new Button("7");
36        b7.setBounds(2,130,40,40);
37        add(b7);
38        Button b8 = new Button("8");
39        b8.setBounds(52,130,40,40);
40        add(b8);
41        Button b9 = new Button("9");
42        b9.setBounds(102,130,40,40);
43        add(b9);
44        Button b0 = new Button("0");
45        b0.setBounds(2,180,40,40);
46        add(b0);
47        Button beq = new Button("=");
48        beq.setBounds(52,180,90,40);
49        add(beq);
50        Button bplus = new
Button("+");
51        bplus.setBounds(152,30,80,40);
52        add(bplus);
53        Button bminus = new
```

Строки 17-61 описывают создание 15 кнопок: 10 кнопок для ввода цифр, 4 кнопки для основных арифметических операций и одна кнопка для расчета результата. После создания каждой из кнопок устанавливаются их местоположения и размеры. Затем кнопки добавляются в контейнер окна.

```
Button("-");

  bminus.setBounds(152,80,80,40);
      add(bminus);
      Button bmul = new Button("*");
      bmul.setBounds(152,130,80,40);
      add(bmul);
      Button bdiv = new Button("/");
      bdiv.setBounds(152,180,80,40);
      add(bdiv);
  b1.addActionListener(new ActionListener()
{
  public void actionPerformed(ActionEvent
event) {
  display.setText(display.getText()+"1");
  }
  });
  b2.addActionListener(new ActionListener()
{
  public void actionPerformed(ActionEvent
event) {
  display.setText(display.getText()+"2");
  }
  });
  b3.addActionListener(new ActionListener()
{
  public void actionPerformed(ActionEvent
event) {
   display.setText(display.getText()+"3");
  }
  });
  b4.addActionListener(new ActionListener()
{
  public void actionPerformed(ActionEvent
event) {
   display.setText(display.getText()+"4");
  }
  });
  b5.addActionListener(new ActionListener()
{
  public void actionPerformed(ActionEvent
event) {
   display.setText(display.getText()+"5");
  }
  });
  b6.addActionListener(new ActionListener()
{
  public void actionPerformed(ActionEvent
event) {
   display.setText(display.getText()+"6");
```

В строках 62-140 для каждой из кнопок производится обработка событий на нажатие. Для цифровых кнопок — добавление к тексту поля ввода соответствующей цифры, для арифметических операций — установка «флага» op соответствующей операции, сохранение в переменной temp введенного в поле display числа, путем преобразования строки в число и очистка дисплея.

```
90    }
91    });
92    b7.addActionListener(new ActionListener()
{
93    public void actionPerformed(ActionEvent
event) {
94     display.setText(display.getText()+"7");
95    }
96    });
97    b8.addActionListener(new ActionListener()
{
98    public void actionPerformed(ActionEvent
event) {
99     display.setText(display.getText()+"8");
100   }
101   });
102   b9.addActionListener(new ActionListener()
{
103   public void actionPerformed(ActionEvent
event) {
104    display.setText(display.getText()+"9");
105   }
106   });
107   b0.addActionListener(new ActionListener()
{
108   public void actionPerformed(ActionEvent
event) {
109    display.setText(display.getText()+"0");
110   }
112   });
113   bplus.addActionListener(new
ActionListener() {
114   public void actionPerformed(ActionEvent
event) {
115     op = '+';
116     temp = Double.valueOf(display.getText());
117     display.setText("");
118   }
119   });
120   bminus.addActionListener(new
ActionListener() {
121   public void actionPerformed(ActionEvent
event) {
122     op = '-';
123     temp = Double.valueOf(display.getText());
124     display.setText("");
125   }
126   });
127   bdiv.addActionListener(new
ActionListener() {
```

```
128   public void actionPerformed(ActionEvent
event) {
129     op = '/';
130     temp = Double.valueOf(display.getText());
131     display.setText("");
132   }
133   });
134   bmul.addActionListener(new
ActionListener() {
135   public void actionPerformed(ActionEvent
event) {
136     op = '*';
137     temp = Double.valueOf(display.getText());
138     display.setText("");
139   }
140   });
141   beq.addActionListener(new ActionListener()
{
142   public void actionPerformed(ActionEvent
event) {
143     switch(op){
      case '+':
144 display.setText(String.valueOf(temp+Double.v
alueOf(display.getText()))); break;
      case '-':
145 display.setText(String.valueOf(temp-
Double.valueOf(display.getText()))); break;
      case '*':
146 display.setText(String.valueOf(temp*Double.v
alueOf(display.getText()))); break;
      case '/':
147 display.setText(String.valueOf(temp/Double.v
alueOf(display.getText()))); break;
148     }
149   }
150   });
151
152     } // Завершение конструктора Calc
153 public static void main(String[] args) { // Начало метода main()
154 new Calc("Calculator"); // Создание объекта Calc
155   } // Завершение метода main()
156 } // Завершение класса Calc
```

В строках 141-151 происходит обработка события нажатия на кнопку «=». Путем перебора значений флага op, определяется, какая операция была выбрана и происходит сложение значения переменной temp со значением, введенным в поле после нажатия одной из кнопок арифметических операций.

Задания к работе

1. Измените данную программу так, чтобы числа можно было бы вводить в поле с клавиатуры, при этом перехватывая возможные исключения типа NumberFormatException.

2. Добавьте к данному калькулятору две кнопки — «Память» и «Вызов из памяти», с помощью которых можно будет сохранить в память числовое значение, находящееся в поле, и вызвать из памяти ранее сохраненное значение.

Глоссарий

А

AWT — Abstract Windowing Toolkit — пакет (библиотека) базовых компонентов графического интерфейса пользователя.

Абстракция — определение, характеризующее уровень обобщенности описания того или иного объекта или процесса.

Абстрактный класс (abstract class) — класс, который содержит один или более абстрактных методов, вследствие чего нельзя создавать экземпляры данного класса. Абстрактные классы определены таким образом, чтобы другие классы могли расширять и конкретизировать их, реализуя абстрактные методы.

Абстрактный метод (abstract method) — метод, не имеющий реализации.

Аргумент — элемент некоторых данных, указываемый при вызове метода.

Array (массив) — см. массив.

Б

Байт-код (bytecode) — машинно-независимый код, генерируемый Java-компилятором и исполняемый Java-интерпретатором.

Байт (byte) — последовательность из восьми битов. В языке программирования Java определен соответствующий тип byte.

В

Виртуальная машина JAVA — программный "механизм выполнения", который безопасно выполняет байт-коды файлов классов Java на микропроцессоре (компьютера или другого электронного устройства).

Возврат (return) — оператор возвращения одного из типов значений методом класса.

Выражение — конструкция, состоящая из данных и операторов, возвращающее некоторое значение в качестве результата.

Г

GUI — см. ГИП.

ГИП — графический интерфейс пользователя — средство организации программного графического интерфейса средствами библиотек AWT и SWING.

Д

Динамическая переменная — переменная, наследуемая в потомках, становящаяся отдельным членом каждого экземпляра, не зависящая от тех же переменных других экземпляров.

Декларация — процесс объявления новых членов класса.

Декремент — оператор, увеличивающий значение операнда на единицу.

E

Event — см. событие.

И

Инкапсуляция — локализация части данных в пределах класса. Поскольку объекты инкапсулируют данные и реализацию, пользователь может рассматривать объект как черный ящик, предоставляющий услуги. Переменные и методы экземпляров класса могут добавляться, удаляться или изменяться, но до тех пор, пока услуги, предоставляемые объектом, не изменяются, нет необходимости переписывать код, использующий данный объект. См. также переменная экземпляра класса и метод экземпляра класса.

Инкремент — оператор, уменьшающий значение операнда на единицу.

Inheritable — см. наследование.

Иерархия — классификация соотношений, в которой каждый элемент, кроме верхнего (называемого корнем), является специализированным видом элемента, расположенного над ним. Каждый элемент может иметь один или несколько элементов, находящихся ниже него в иерархии. В иерархии классов Java, образуемой при наследовании, корнем является класс Object.

Имя — символьная или символьно-цифровая идентификация какого-либо члена класса или самого класса, используемая в качестве вызова.

Идентификатор — имя объекта для компилятора, используемое для связывания в процессе компиляции программы.

Интерфейс — понятие, используемое для определения набора методов и постоянных значений (класса специального вида). Интерфейс в дальнейшем может реализовываться классами, которые определяют этот интерфейс с ключевым словом implements.

ИЛИ — логический оператор, возвращающий значение «истина» при условии наличия хотя бы одного из операндов выражения, имеющего значение «истина».

Исключающее ИЛИ — логический оператор, возвращающий значение «истина» при условии различных логических значений двух операндов выражения.

Исключительная ситуация — ситуация, возникающая во время работы программы вследствие неудачного, или в случае невозможности выполнения операции, или запрограммированная с использованием оператора throw. Если в программе не предусмотрена реакция на возникшую ситуацию, дальнейшее исполнение программы невозможно.

Источник события — объект, чаще компонент ГИП, являющийся возбудителем некоторого события.

К

Класс — тип в языке программирования Java, определяющий реализацию особого вида объекта. Описание класса определяет экземпляр класса, его переменные и методы.

Компонент — элемент библиотеки графического интерфейса пользователя AWT или SWING.

Код — исходный текст программы или класса.

Комментарий — часть кода, предваряемая специальным символом (в JAVA это символ //), которую компилятор при интерпретации игнорирует.

Константа — член класса, который на протяжении всей программы не может быть изменен, в том числе и в потомках и экземплярах класса. Задается модификатором final.

Конструктор — особый метод, имеющийся в каждом классе, осуществляющий процесс создания экземпляров данного класса. В случае отсутствия явного описания, используется конструктор суперкласса.

Контейнер — сущность, обеспечивающая управление, безопасность, разработку и сервисы выполнения компонент.

Л

Логический оператор — оператор, выполняющий действие с операндами логического типа.

Логическая переменная/константа — член класса, имеющий логический тип.

Логическое выражение — совокупность операторов и операндов, возвращающая в результате логическое значение.

Логическая ошибка — ошибка программиста, характеризующаяся неверной постановкой вычислительной задачи в программе.

Локальная переменная — переменная класса, которая не участвует в наследовании и конструкции экземпляров. Задается модификатором static.

M

Массив (array) — совокупность элементов данных одного типа, в которой позиция каждого элемента однозначно определена целым числом (индексом массива).

Модульность — один из принципов объектно-ориентированного программирования, предполагающий, что каждый класс должен составлять отдельный модуль, а члены класса, обращение к которым не планируется в дальнейшем, должны быть *инкапсулированы*.

Метод — функция, определенная в классе. Пока не оговорено обратное, метод не является статическим. Различают методы экземпляров и методы классов. Методы классов — локальные методы, как и переменные, задаются модификатором `static`.

Модификатор — элемент языка объектно-ориентированного программирования, задающий параметры декларируемых членов.

H

Наследование — концепция классов, автоматически включающих все переменные и методы, определенные в супертипе.

Name — см. имя.

O

Объект — основной компоновочный блок объектно-ориентированных программ. Каждый объект программного модуля состоит из данных (переменные экземпляра) и функциональных возможностей (методы экземпляра).

Объектно-ориентированное программирование — метод проектирования программного обеспечения, позволяющий моделировать абстрактные или реальные объекты при помощи классов и объектов.

Окно — элемент графического интерфейса пользователя, являющийся контейнером, имеющий возможность помещать в себя другие компоненты.

Отрицание — логический процесс изменения состояния логического выражения, переменной или константы на обратное.

Оператор — элемент, использующийся в процессе вычисления выражений.

Объявление — процесс присваивания начального значения переменной или константе.

Обработка *события* — некоторый набор операторов, процедур и функций, выполняющийся при наступлении события.

П

Переменная — элемент данных, имеющий идентифицирующее его имя.

Подкласс — класс, наследованный от некоторого суперкласса.

Поле — элемент класса. Пока не определено обратное, поле не является статическим.

Полиморфизм — один из базовых принципов ООП, опирающийся на наследование и предполагающий наличие нескольких объектов, наследованных от одного и того же класса, но имеющих разное содержание.

Р

Рабочий стол — часть графического интерфейса пользователя оболочки программирования, отображающая совокупность элементов управления.

С

Символ — элемент алфавита, а также элемент данных текстового типа.

Событие — элемент класса Event.

Ссылка — объект, являющийся сопоставлением некоторому идентификатору некоторой области памяти, содержащей данные.

Строка — элемент данных, представляющий собой набор символов.

Суперкласс — класс, являющийся родителем одного или нескольких подклассов. Класс Object является суперклассом для любого класса.

Т

Тип — параметр данных, характеризующий их принадлежность к определенному множеству.

У

Условие — некоторый блок программы, содержащий выражение, переменную или константу, имеющий логический тип и являющийся основой выполнения какого-либо блока задач. Используется, в основном, в циклах и операторе ветвления.

Управляющие конструкции — разновидность операторов языка программирования, функции которых заключаются в управлении ходом программы.

Ф

Фокус — определение, характеризующее текущее состояние компонента. Компоненты, имеющие фокус, могут получать команды от пользователя с клавиатуры.

Ч

Член класса — метод, константа или переменная, описанная в данном классе.

Э

Экземпляр — объект, созданный на основе некоторого класса.

Список литературы

Использованная:

1. *Барнет Э.* Eclipse IDE Карманный справочник: Пер. с англ. — М.:КУДИЦ-ОБРАЗ, 2006. — 160 с.

2. *Хабибуллин И. Ш.* Самоучитель Java 2. — СПб.: БХВ-Петербург, 2007. — 720 с.

3. *Шилдт Г.* Полный справочник по Java. — М.: Вильямс, 2007. — 1040 с.

4. *Ноутон П., Шилдт Г.* Java 2. — СПб:, BHV-Санкт-Петербург, 2008. — 1072 с.

Электронные издания

5. *Монахов В.В.* Материалы курсов, разработанных в рамках программы Sun Microsystems Teaching Grants в 2006 году — «Язык программирования Java». СПбГУ, 2006.

6. Материалы интернет-сайта http://ru.sun.com

Рекомендуемая:

7. *Фишер Т.* Java. Карманный справочник. — М.: Вильямс, 2008. — 224 с.

8. *Хемраджани А.* Гибкая разработка приложений на Java с помощью Spring, Hibernate и Eclipse. — М.: Вильямс, 2008. — 352 с.

9. *Хабибуллин И. Ш.* Самоучитель Java 2. — СПб.: БХВ-Петербург, 2007. — 720 с.

10. *Шилдт Г.* Swing. Руководство для начинающих. — М.: Вильямс, 2007. — 704 с.

11. *Барнет Э.* Eclipse IDE Карманный справочник: Пер. с англ. — М.:КУДИЦ-ОБРАЗ, 2006. — 160 с.