Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»

ЛАБОРАТОРНЫЙ ПРАКТИКУМ «ОРГАНИЗАЦИЯ РАБОТЫ С ФАЙЛАМИ НА ОСНОВЕ JAVA»

Выпускная квалификационная работа
по направлению подготовки 44.03.04 Профессиональное обучение
(по отраслям)
профилю подготовки «Энергетика»
профилизации «Компьютерные технологии автоматизации и управления»

Идентификационный код ВКР: 187

Екатеринбург 2017

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»
Институт инженерно-педагогического образования
Кафедра информационных систем и технологий

К ЗАЩИТЕ ДОПУСКАЮ
Заведующая кафедрой ИС
_____Н. С. Толстова
«____»_____ 2017 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ЛАБОРАТОРНЫЙ ПРАКТИКУМ «ОРГАНИЗАЦИЯ РАБОТЫ С ФАЙЛАМИ НА ОСНОВЕ JAVA»

Исполнитель:
обучающийся группы № КТэ-402 _____ А. Н. Чернышов
(Подпись)
Руководитель:
старший преподаватель _____ Т. П. Телепова
(Подпись)
Нормоконтролер:
старший преподаватель _____ Т. В. Рыжкова
(Подпись)

Екатеринбург 2017

АННОТАЦИЯ

Выпускная квалификационная работа состоит из лабораторного практикума «Организация работы с файлами на основе Java» и пояснительной записки на 58 страницах, содержащей 36 рисунков, 6 таблиц, 26 источников литературы, а также 1 приложение на 2 страницах.

Ключевые слова: JAVA, JAVA DEVELOPMENT KIT, ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, INTELLIJ IDEA.

Чернышов, А. Н. Организация работы с файлами на основе Java: выпускная квалификационная работа/ А. Н. Чернышов; Рос. гос. проф.-пед. ун-т, Ин-т инж.-пед. образования, Каф. информ. систем и технологий. — Екатеринбург, 2017. — 66 с.

В работе рассмотрен процесс обучения основам работы с файлами в ОС Windows на основе языка программирования Java.

Целью работы является разработка лабораторного практикума по работе с файлами на основе языка программирования Java. Для достижения цели были проанализированы теоретический и практический материал по программированию на языке Java процессов работы с файловой системой. Разработана структура лабораторного практикума и сформировано их содержание. Сформирован электронный вариант лабораторного практикума.

Несмотря на стремительное развитие многих языков программирования, Java остается ключевым и наиболее удобным для работы с файлами в операционной системе Windows, и написания программ.

СОДЕРЖАНИЕ

Введение

В современном мире информационные технологии играют огромную роль. Сложно представить нашу жизнь без Интернета и персонального компьютера, они есть практически у каждого человека. Информационные технологии повлияли на развитие многих сфер деятельности человека и существенно сказались ни их развитии. Огромное количество сил и времени отводилось на вычисление экономических процессов, влияющих на экономику, а при необходимости вычислить результаты экономических процессов одного предприятия, на уровне одного региона или всей страны вычисления производились очень долго, да и точность этих вычислений была далеко не идеальной. В настоящее время любые вычисления выполняются с помощью компьютеров. При этом результаты точные, и вычисляются очень быстро.

Язык программирования Java используется сейчас не только в инженерных расчетах, но и в медицине, а также и экономике.

Для создания программ необходимы языки программирования. Язык Java является одним из языков программирования, необходимым для создания программного обеспечения и многих других задач.

В банках программистам очень часто приходится работать с файлами и каталогами. Для этого в основном используется объектно-ориентированный язык программирования Java. Он позволяет в короткие сроки решать максимально сложные задачи.

Java — объектно-ориентированный язык, удобный и надёжный в эксплуатации благодаря таким своим достоинствам, как многозадачность, поддержка протоколов Internet.

Одно из главных преимуществ языка Java — его независимость от платформы, на которой выполняются программы. Таким образом, один и тот же код можно запускать под управлением операционных систем Windows, Linux, FreeBSD, Solaris, Apple Mac и др. Это становится очень важным, когда

программы загружаются посредством глобальной сети интернет и используются на различных платформах.

Java является сравнительно простым языком программирования. Быстрому созданию надежного программного кода способствует автоматическое управление памятью, отсутствие многократного наследования и указателей.

Программист — очень важная и ответственная профессия в современном мире и значимость ее будет только увеличиваться с каждым годом.

Объектом исследования является процесс обучения основам работы с файлами на основе языка программирования Java.

Предметом исследования являются учебные пособия по языку объектно-ориентированного программирования Java.

Целью работы является разработка лабораторного практикума по работе с файлами на основе языка программирования Java.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Проанализировать теоретический и практический материал по программированию на языке Java процессов работы с файловой системой.
2. Разработать структуру лабораторного практикума и сформировать их содержание.
3. Сформировать электронный вариант лабораторного практикума.

1 ТЕОРЕТИЧСКИЕ ОСНОВЫ ОРГАНИЗАЦИИ РАБОТЫ С ФАЙЛАМИ В ОС WINDOWS

1.1 Анализ файловых систем ОС Windows

Любая информация (текстовые данные, программы, фотографии) может сохраняться во внешней памяти компьютера только в виде файлов.

Файл — это упорядоченная совокупность данных, хранимая на диске и занимающая именованную область внешней памяти (это место хранения информации, с которой компьютер работает) [11].

Файл в процессе обработки рассматривается как единое целое, которое имеет собственное имя. Имя файла состоит из двух частей, отделенных друг от друга точкой. Первая часть — это собственно имя файла, вторая часть — расширение (последовательность символов, добавляемых к имени файла и предназначенных для идентификации типа). В таблице 1 ниже приведены примеры типов файлов, а также расширений.

Таблица 1 — Типы файлов и расширения

Тип файла	Расширение
Программы	exe, com
Текстовые файлы	doc, txt, docx
Графические файлы	bmp, jpeg, jpg, gif
Звуковые файлы	wav, mp3
Видеофайлы	avi, mov
Коды программ на языках программирования	pas, bas

Файловая система — регламент, определяющий способ организации, хранения и именования данных на носителях информации. Она определяет формат физического хранения информации, которую принято группировать в виде файлов. Конкретная файловая система определяет размер имени файла (папки), максимальный возможный размер файла и раздела, набор атрибутов

(метаданные, которые описывают файл) файла. Некоторые файловые системы предоставляют сервисные возможности, например, разграничение доступа или шифрование файлов [12].

Файловая система связывает носитель информации и API (набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах) для доступа к файлам. Прикладная программа обращается к файлу, она не имеет никакого представления о том, каким образом расположена информация в конкретном файле, так же, как и на каком физическом типе носителя (CD, жёстком диске, магнитной ленте или блоке флэш-памяти) он записан. Всё, что знает программа — это имя файла, его размер и атрибуты. Эти данные она получает от драйвера файловой системы. Именно файловая система устанавливает, где и как будет записан файл на физическом носителе (например, жёстком диске).

С точки зрения операционной системы, весь диск представляет собой набор кластеров (единица хранения данных на гибких и жёстких дисках компьютеров) размером от 512 байт и выше. Драйверы файловой системы организуют кластеры в файлы и каталоги (реально являющиеся файлами, содержащими список файлов в этом каталоге). Драйверы отслеживают, какие из кластеров в настоящее время используются, какие свободны, какие помечены как неисправные.

Основные ***функции*** любой файловой системы:

1. Именование файлов.
2. Программный интерфейс работы с файлами для приложений.
3. Отображения логической модели файловой системы на физическую организацию хранилища данных.
4. Организация устойчивости файловой системы к сбоям питания, ошибкам аппаратных и программных средств.
5. Содержание параметров файла, необходимых для правильного его взаимодействия с другими объектами системы (ядро, приложения и пр.).

Наиболее распространёнными и часто используемыми файловыми системами являются FAT и NTFS. В приведенной ниже таблице 2 выполнена сравнительная характеристика этих файловых систем, в результате которой выявлены плюсы и минусы каждой системы.

Таблица 2 — Сравнение файловых систем FAT и NTFS

Параметры сравнения	**FAT32**	**NTFS**
Операционные системы, поддерживающие формат файловой системы.	Windows 98, NT5, Windows Xp, Windows 2000, Windows Vista, Windows Seven.	NT4, NT5, NT5.1, Windows Xp, Windows Vista, Windows 7.
Максимальный размер тома	Практически неограничен	Практически неограничен
Безопасность	Нет	Да
Сжатие	Нет	Да
Устойчивость к сбоям	Плохая (средства оптимизации по скорости привели к появлению слабых по надежности мест)	Полная — автоматическое восстановление системы при любых сбоях (не считая физические ошибки записи, когда пишется одно, а на самом деле записывается другое).
Экономичность	Средняя (улучшена за счет уменьшения размеров кластеров)	Максимальная (очень эффективная и разнообразная система хранения данных).
Быстродействие	Полностью, но на дисках большого размера (десятки гигабайт) начинаются серьезные проблемы с общей организацией данных	Система не очень эффективна для малых и простых разделов (до 1 ГБ), но работа с огромными массивами данных и внушительными каталогами организована как нельзя более эффективно и очень сильно превосходит по скорости другие системы.

Основной фактор, от которого зависит быстродействие файловой системы — это, объем оперативной памяти машины (персонального компьютера). Каждая из представленных файловых систем обладает рядом преимуществ и недостатков. Выбирая одну из них, необходимо определиться, для каких целей будет использован компьютер, и какие у него параметры. Если система устанавливается на мощный серверный компьютер, то более подходящим вариантом окажется NTFS. При работе на домашнем компьютере достаточным будет использование FAT32.

Инструменты ОС Windows по работе с файловой системой

Повысить эффективность работы с файловой системой призваны специальные программы — файловые менеджеры ОС Windows. Они помогают пользователю выполнять наиболее распространенные задачи для работы с файлами: создание, редактирование, просмотр, проигрывание, перемещение, копирование, удаление, настройка параметров, свойств и поиск документов.

Файловый менеджер расширяет арсенал средств операционной системы, наполняя ее дополнительными функциями (резервное копирование, работа с сетью, управление оргтехникой). При выборе надежного проводника windows нужно опираться на его объем функциональных возможностей, удобство работы с файловыми системами и перспективы для расширений (поддержка плагинов). Стандартным приложением для работы с файлами в Windows является проводник [24].

Проводник — средство, дающее возможность пользователю видеть в иерархической форме структуру, размещение папок и быстро переходить к какому-либо объекту (папке, файлу, ярлыку), а также выполнять ряд действий с папками и файлами. Программа «Проводник» предназначена для создания и копирования, перемещения, удаления, открытия, переименования папок и файлов (рисунок 1) [26].

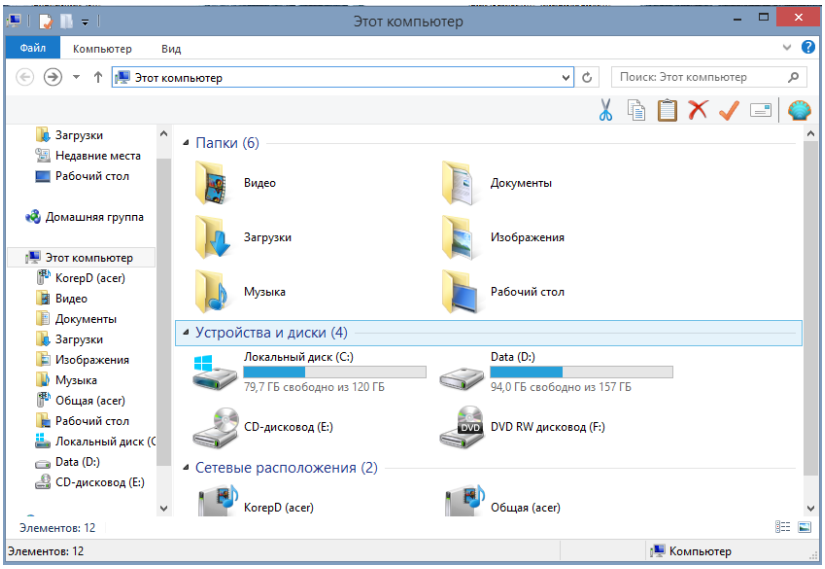


Рисунок 1 — Интерфейс Проводника

Далее представлены наиболее распространённые файловые менеджеры, которые устанавливаются дополнительно и являются личным выбором пользователя персонального компьютера.

Total Commander — наиболее распространенный файловый менеджер для ОС windows, разработанный более 20 лет назад в Швейцарии. Это классический двух панельный файловый менеджер, который заслужил себе репутацию профессионального помощника управления файлами. Он по достоинству оценен опытными пользователями, благодаря высокой стабильности, удобству и расширяемости. Этот мощный менеджер файлов включает в себя обширный пакет встроенных инструментов (рисунок 2) [13].

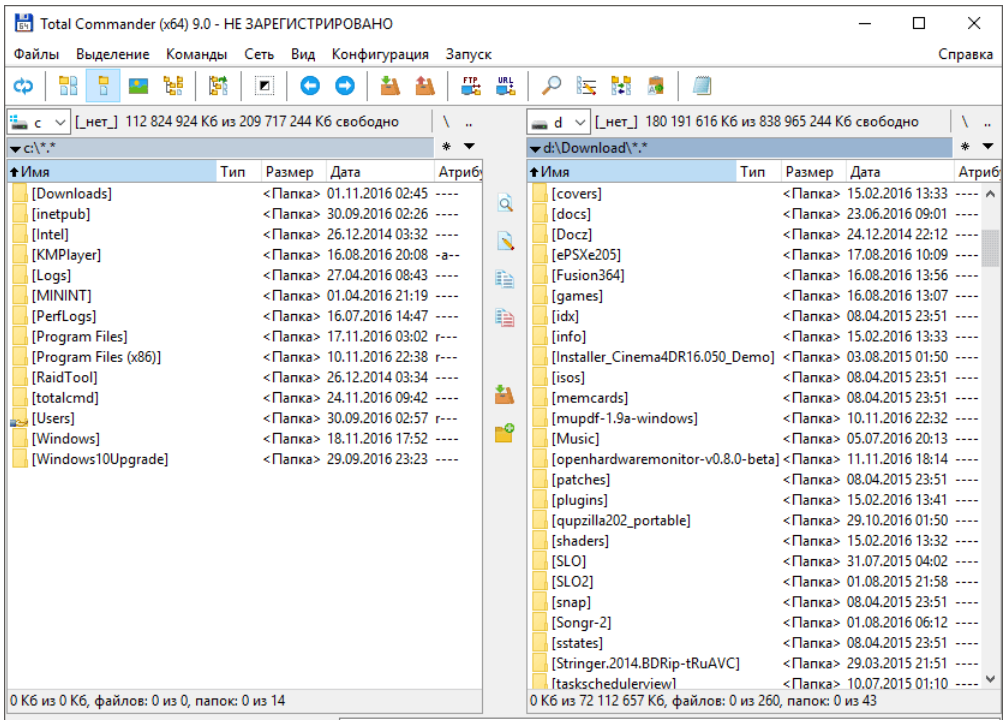


Рисунок 2 — Интерфейс Total Commander

FreeCommander — это перспективный файл менеджер, который постоянно дорабатывается, укомплектовывается новыми инструментами для слаженной работы с файловыми объектами. Легко настраиваемый внешний вид системы, комбинации «горячих клавиш» и отображение папок и файлов. Бесплатная программа привлекает опытных пользователей продуманным до мелочей внешним видом и средствами для тонкой настройки (рисунок 3) [13].

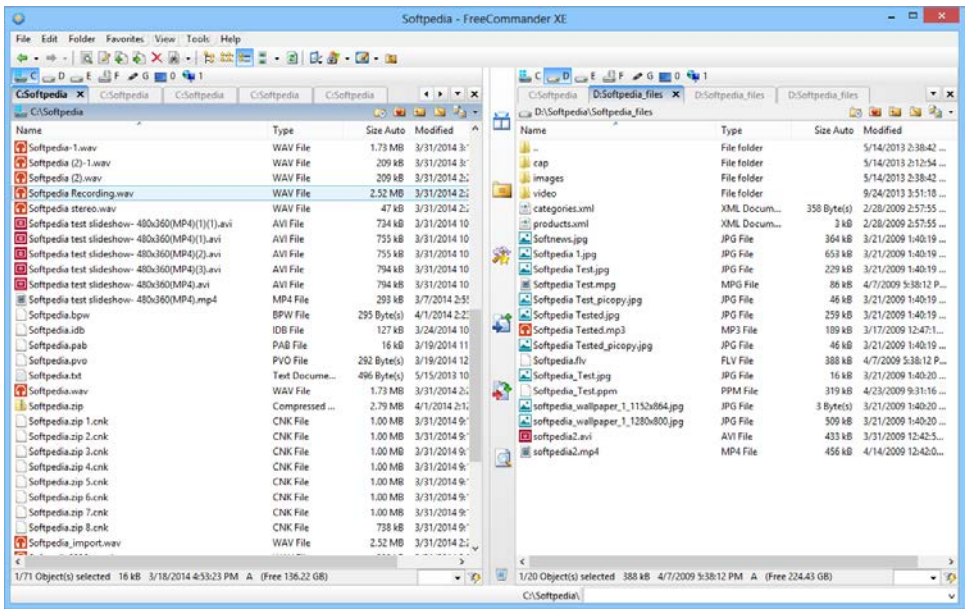


Рисунок 3 — Интерфейс FreeCommander

FAR Manager — бесплатная программа для работы с файлами на платформе windows, которая отличается консольным внешним видом, высокой скоростью и потребляет минимум системных ресурсов. Данный менеджер файлов проводит операции с файловыми системами компактным нажатием пары клавиш. Разработанный на основе Norton Commander, FAR Manager использует стандартные клавиатурные комбинации (рисунок 4) [13].

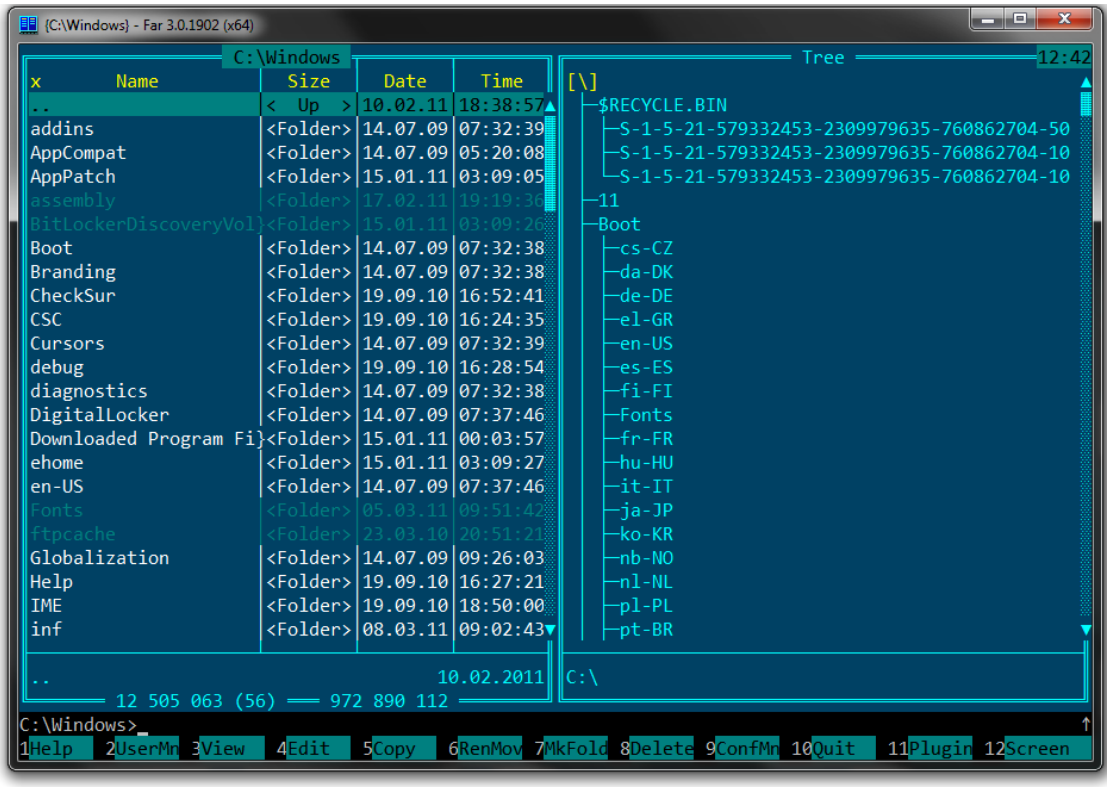


Рисунок 4 — Интерфейс FAR Manager

В таблице 3 ниже представлено сравнение дополнительных файловых менеджеров.

Необходимость использования менеджера файлов очевидна. В операционных системах ОС Windows недостаточно средств для быстрой и эффективной работы с файловыми материалами. Не теряют своих позиций в рейтинге пользователей старые, но продуктивные программы Total Commander и FAR Manager. Имея несколько устаревший двух панельный интерфейс, они предлагают эффективные инструменты для работы с файлами и папками, имеют ряд полезных расширений.

Таблица 3 — Сравнение файловых менеджеров

Параметры сравнения	**FAR Manager**	**FreeCommander**	**Total Commander**
Операционная система	Windows	Windows	Windows, *nix
Создатель	Eugene Roshal (originally) (1996—2000), FAR Group(2000—2012)	Marek Jasinski	Christian Ghisler
Дата первого публичного релиза	1996	2004	1993
Стоимость	Бесплатно	Бесплатно	CHF 40 / €37
Сжатие (архивирование) файлов	Да	Да	Да
Изменение кодировки	Да	Нет	Да
Выбор файлов, удовлетворяющих фильтру	Да	Нет	Да
Управление доступом к файлам	Нет	Нет	Да
Массовое переименование	Да	Да	Да

1.2 Применение программирования для работы с файлами и каталогами

1.2.1 Стандартные инструменты программирования ОС Windows

Командная строка представляет собой одну из возможностей Windows, обеспечивающую ввод команд MS-DOS (дисковая операционная система для компьютеров на базе архитектуры x86) и других компьютерных команд. Важность этой командной строки состоит в том, что она позволяет выполнять задачи без помощи графического интерфейса Windows. Обычно командная строка используется только опытными пользователями.

При работе с командной строкой сам термин командная строка обозначает также и закрывающую угловую скобку (>, иначе: символ больше). Это указывает на то, что интерфейс командной строки может принимать команды. Текущий рабочий каталог (или расположение), где будет выполняться данная команда, может быть также указана в командной строке. Например, если при открытии окна командной строки в этой строке отображается текст «C:\>» и мигающий курсор справа от закрывающей угловой скобки (>), это означает, что введенная команда будет выполняться на всем диске C данного компьютера.

Командная оболочка (CMD) — это отдельный программный продукт, который обеспечивает прямую связь между пользователем и операционной системой. Текстовый пользовательский интерфейс командной строки предоставляет среду, в которой выполняются приложения и служебные программы с текстовым интерфейсом. В командной оболочке программы выполняются, и результат выполнения отображается на экране (рисунок 5) [22].

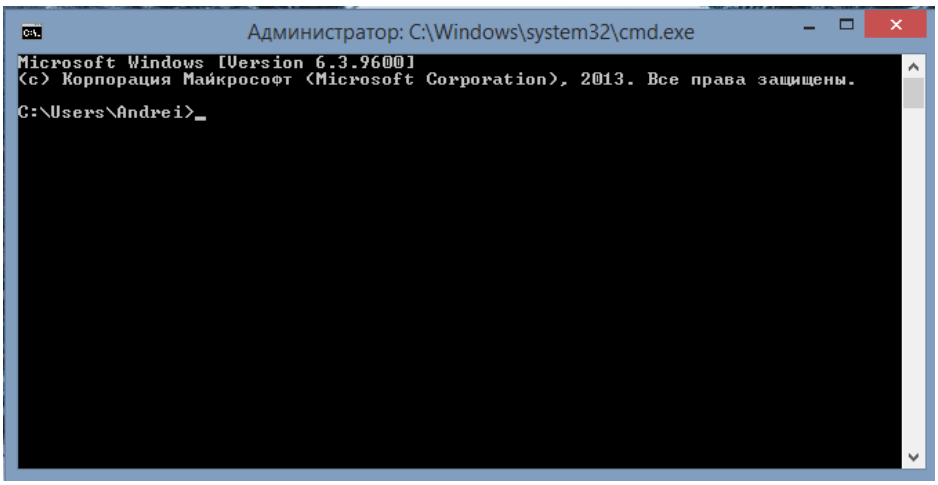


Рисунок 5 — Интерфейс CMD

Командная оболочка ОС Windows использует интерпретатор команд Cmd.exe, который загружает приложения и направляет поток данных между приложениями, для перевода введенной команды в понятный системе вид. Консоль командной строки присутствует во всех версиях операционных систем Windows. Отличием работы из командной строки является полное отсутствие больших и громоздких графических утилит.

Для каждого приложения, поддерживающего командную строку, предусмотрен специальный набор команд, которые может обрабатывать программа. Параметры команд могут иметь самый разный формат. Чтобы передать программе параметры, необходимо ввести в командной строке имя приложения и параметры команд. После нажатия Enter запустится приложение с введенными командами.

Windows PowerShell — это оболочка командной строки на основе задач, а также язык сценариев, предназначенный специально для системного администрирования. Созданная на основе .NET Framework, служба Windows PowerShell помогает ИТ-специалистам и опытным пользователям в управлении и автоматизации администрирования операционной системы Windows, а также приложений, работающих под управлением Windows (рисунок 6) [23].

Рисунок 6 — Интерфейс Windows PowerShell

Команды Windows PowerShell, которые называются командлетами, позволяют управлять компьютерами из командной строки. Поставщики Windows PowerShell позволяют получить доступ к хранилищам данных, например, реестру и хранилищу сертификатов, так же легко, как и к файловой системе. Кроме того, Windows PowerShell включает многофункциональное средство синтаксического анализа выражений и полностью разработанный язык сценариев.

1.2.2 Применение Объектно-ориентированного программирования для автоматизации работы с файлами и каталогами

История развития Объектно-ориентированного программирования (ООП)

Объектно-ориентированное программирование возникло в результате развития идеологии процедурного программирования (программирование на императивном языке, при котором последовательно выполняемые операторы можно собрать в подпрограммы, более крупные целостные единицы кода, с помощью механизмов самого языка), где данные и подпрограммы (процедуры, функции) их обработки формально не связаны. Для дальнейшего развития объектно-ориентированного программирования часто большое значение

имеют понятия события (так называемое событийно-ориентированное программирование) и компонента (компонентное программирование, КОП).

Взаимодействие объектов происходит посредством сообщений. Результатом дальнейшего развития объектно-ориентированного программирования, по-видимому, будет агентно-ориентированое программирование, где агенты — независимые части кода на уровне выполнения. Взаимодействие агентов происходит посредством изменения среды, в которой они находятся.

Субъектно-ориентированное программирование (метод построения объектно-ориентированных систем, как композиции субъектов) расширяет понятие объекта посредством обеспечения более унифицированного и независимого взаимодействия объектов. Может являться переходной стадией между объектно-ориентированного программирования и агентным программированием в части самостоятельного их взаимодействия.

После появления *языков третьего поколения* (1967 г.) ведущие специалисты в области программирования выдвинули идею преобразования постулата фон Неймана: «данные и программы неразличимы в памяти машины». Их цель заключалась в максимальном сближении данных и программы. Решая поставленную задачу, они столкнулись с задачей, решить которую без декомпозиции оказалось невозможно, а традиционные структурные декомпозиции не сильно упрощали задачу. Усилия многих программистов и системных аналитиков, направленные на формализацию подхода, увенчались успехом.

Были разработаны три основополагающих принципа того, что потом стало называться объектно-ориентированным программированием (ООП):

1. Наследование.
2. Инкапсуляция.
3. Полиморфизм.

Результатом их первого применения стал ***язык Симула-1*** (Simula-1), в котором был введен новый тип — объект. В описании этого типа одновременно указывались данные (поля) и процедуры, их обрабатывающие — ме-

тоды. Родственные объекты объединялись в классы, описания которых оформлялись в виде блоков программы. При этом класс можно использовать в качестве префикса к другим классам, которые становятся в этом случае подклассами первого. Взгляд на программирование «под новым углом» (отличным от процедурного) предложили Алан Кэй и Дэн Ингаллс в языке Smalltalk. Здесь понятие класса стало основообразующей идеей для всех остальных конструкций языка (то есть класс в Смолтоке (объектно-ориентированный язык программирования с динамической типизацией, основанный на идее посылки сообщений) является примитивом, посредством которого описаны более сложные конструкции). Именно он стал первым широко распространённым объектно-ориентированным языком программирования.

В настоящее время количество прикладных языков программирования (список языков), реализующих объектно-ориентированную парадигму, является наибольшим по отношению к другим парадигмам. Наиболее распространённые в промышленности языки (C++, Delphi, C#, Java и др.) воплощают объектную модель Симулы. Примерами языков, опирающихся на модель Смолтока (служит для того, чтобы увязать модель, существующую в воображении человека, с той, которая существует в компьютере), являются Objective-C, Python, Ruby.

Объектно-ориентированное программирование оказалось пригодным не только для моделирования (Simula) и разработки графических приложений (SmallTalk), но и для создания большинства других приложений, а его приближенность к человеческому мышлению и возможность многократного использования кода сделали его одной из наиболее бурно используемых концепций в программировании.

Объектно-ориентированный подход помогает справиться с такими сложными проблемами, как *уменьшение* сложности программного обеспечения; *повышение надежности* программного обеспечения; обеспечение возможности *модификации* отдельных компонентов программного обеспечения

без изменения остальных его компонентов; обеспечение возможности *повторного использования* отдельных компонентов программного обеспечения.

Сравнение объектно-ориентированных языков программирования

Рассмотрим наиболее распространенные языки объектно-ориентированного программирования.

C++ — компилируемый строго типизированный язык программирования общего назначения. Поддерживает разные парадигмы программирования: процедурную, обобщённую, функциональную; наибольшее внимание уделено поддержке объектно-ориентированного программирования. Разработка языка началась в 1979 году. Целью создания C++ было дополнение C возможностями, удобными для масштабной разработки ПО, с сохранением гибкости, скорости и портабельности (переносимость программного обеспечения) [15].

C# — язык программирования, сочетающий объектно-ориентированные и контекстно-ориентированные концепции. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как основной язык разработки приложений для платформы Microsoft .NET. Компилятор с C# входит в стандартную установку самой .NET, поэтому программы на нём можно создавать и компилировать даже без инструментальных средств вроде Visual Studio [15].

Java — объектно-ориентированный язык программирования, разрабатываемый компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года. Изначально новый язык программирования назывался Oak (James Gosling) и разрабатывался для бытовой электроники, но впоследствии был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения [15].

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах

как язык сценариев для придания интерактивности веб-страницам [16]. В приведенной ниже таблице 4 происходит сравнение таких языков программирования, как C++, C#, Java, Java Script по критерию «Возможность».

Условные обозначения:

- + — указанная возможность присутствует;
- - — указанная возможность отсутствует;
- +/- — возможность поддерживается не полностью;
- -/+ — возможность поддерживается очень ограниченно.

Таблица 4 — Сравнение языков программирования

Возможность	Языки			
	C++	C#	Java	Java Script
Императивная	+	+	+	+
Объектно-ориентированная	+	+	+	+
Функциональная	–/+	+/–	–/+	+/-
Логическая	–	–	–	–
Распределенная	+/–	–/+	+	–
Статическая типизация	+	+	+	–
Параметрический полиморфизм	–	+	+	–
Возможность компиляции	+	+	+	+
Интерпретатор командной строки	+/–	+	–	+
Условная компиляция	+	+	–/+	–/+
Блок else (исключения)	–	+	+	-
Многомерные массивы	+	+	+/-	+/–
Контроль границ массивов	+/–	+	+	–
Интерфейсы	+	+	+	–
Mixins	–/+	–	+	–
Анонимные функции	+	+	+	+
Лексические замыкания	+	+	+	+
Шаблоны	+	+	+	–
Поддержка Unicode в идентификаторах	+	+	+	+
Перегрузка функций	+	+	+	–/+

В результате сравнения языков программирования, приведенного выше в таблице было выявлено, что **Java** является наиболее подходящим языком программирования для использования его в различных сферах и для различных целей. Используя язык программирования Java можно решить множество конкретных сложных задач, в кротчайшие сроки. Для решения задач, таких как — автоматизация работы с файловой системой, с электронной почтой Java также подходит наиболее целесообразней, нежели другие языки программирования, присутствующие в таблице сравнения. Главный плюс языка Java это его простота для изучения. При разработке Java было уделено большое внимание простоте языка, поэтому программы на Java, по сравнению с программами на других языках, проще писать, компилировать, отлаживать и изучать.

Объектно-ориентированный язык программирования Java

Java — язык программирования, разработанный компанией Sun Microsystems. Приложения Java обычно компилируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине (JVM) независимо от компьютерной архитектуры. Дата официального выпуска — 23 мая 1995 года. Сегодня технология Java предоставляет средства для превращения статических Web-страниц в интерактивные динамические документы и для создания распределенных не зависящих от платформы приложений.

Microsystems — североамериканская компания, производитель программного и аппаратного обеспечения, образована в 1982 году, в период с апреля 2009 года по январь 2010 года была поглощена корпорацией Oracle. Штаб-квартира компании располагалась в Санта-Кларе.

Основные особенности языка Java

Программы на Java *транслируются в байт-код*, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор. Достоинство подобного способа выполнения программ — в полной независимости байт-

кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java *является гибкая система безопасности* благодаря тому, что исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером) вызывают немедленное прерывание.

Часто к *недостаткам* концепции виртуальной машины относят то, что исполнение байт-кода виртуальной машиной может снижать производительность программ и алгоритмов, реализованных на языке Java. В последнее время был внесен ряд усовершенствований, которые несколько увеличили скорость выполнения программ на Java:

1. Применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы (JIT-технология) с возможностью сохранения версий класса в машинном коде.

2. Широкое использование платформенно-ориентированного кода (native-код) в стандартных библиотеках.

3. Аппаратные средства, обеспечивающие ускоренную обработку байт-кода (например, технология Jazelle, поддерживаемая некоторыми процессорами фирмы ARM).

Процесс создания работающего Java-приложения (рисунок 7).

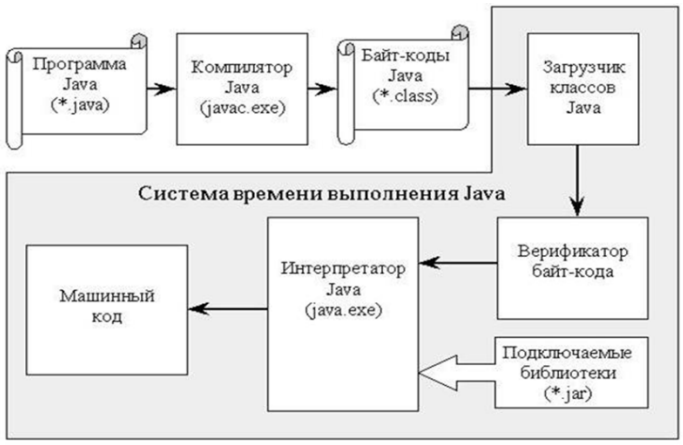


Рисунок 7 — Процесс создания работающего Java-приложения

Транслятор Java предполагает, что исходные тексты программ хранятся в файлах с расширениями *Java*.

Получаемый в процессе трансляции *байт-код* для каждого класса записывается в отдельном выходном файле, с именем совпадающем с именем класса, и расширением *class*. Именно *class-файлы*, содержащие байт-код, интерпретируются системой времени выполнения Java (набор библиотек (модулей) той или иной системы программирования, поставляемых вместе с компилятором, операционной системой или средой разработки программ, обеспечивает поддержку функций, предоставляемых системой программирования, во время выполнения программы от начала до её завершения) в машинный код конкретной системы.

Прежде всего, байт-код Java загружается в систему времени выполнения загрузчиком классов. Загрузчик классов отвечает за то, чтобы были загружены все классы, необходимые для выполнения приложения. Затем байт-код проверяется верификатором байт-кода на отсутствие операций, которые могли бы нарушить безопасность системы или вызвать в ней аварийную ситуацию.

Важно отметить, что загрузчик классов и верификатор байт-кодов не делают никаких предположений относительно происхождения кодов, получены с локальной файловой системы или с другого континента. Верификатор гарантирует, что любой код, прошедший проверку, может быть использован интерпретатором без риска повредить его (интерпретатор), а именно:

1. Не может произойти переполнение или исчерпание стека.
2. Параметры для инструкций байт-машины имеют нужный тип.
3. Доступ к полям и методам объектов не нарушает объявленных в классе правил (public, private, protected).

После проверки на безопасность байт-код *интерпретируется* в машинный код и запускается на выполнение интерпретатором. Классы, полученные локально (заслуживающие безусловного доверия), и классы, присланные по сети из остального мира (и потенциально враждебные), находятся

в разных пространствах имён. При разрешении ссылки на какой-либо класс он ищется, прежде всего, в локальном пространстве. Это не позволяет «внешним» кодам подменить один из базовых классов в системе. Также в процессе интерпретации происходит подключение необходимых библиотек (файлы с расширением jar). Весь описанный процесс исполнения Java программ изображён на (рисунке 7).

Как уже было сказано выше, технология Java предполагает лёгкую переносимость программных продуктов с одной платформы на другую. Такую степень лёгкости переноса не обеспечивает никакой язык программирования.

Основные возможности Java:

- автоматическое управление памятью;
- расширенные возможности обработки исключительных ситуаций;
- богатый набор средств фильтрации ввода/вывода;
- набор стандартных коллекций, таких как массив <#"justify">§на уровне отдельных SQL-запросов — на основе JDBC, SQLJ;
- на уровне концепции объектов, обладающих способностью к хранению в базе данных — на основе Java Data Objects и Java Persistence API;
- поддержка шаблонов (начиная с версии 1.5);
- параллельное выполнение программ.

Компиляция программы на языке Java

Если приложение Java (или апплет) должно работать на нескольких платформах, нет необходимости компилировать его исходные тексты несколько раз. Можно откомпилировать и отладить приложение Java на одной, наиболее удобной платформе. В результате получится байт-код, пригодный для любой платформы, где есть виртуальный процессор Java (рисунок 8).

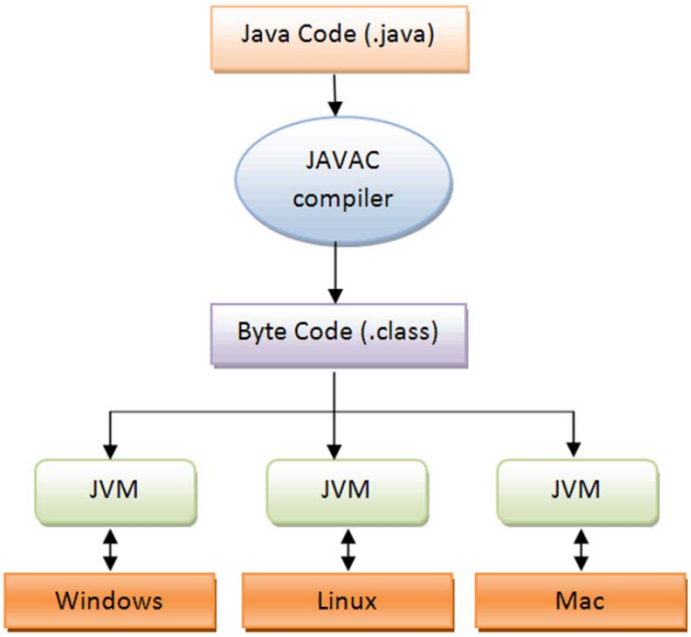


Рисунок 8 — Структура программы на Java для любой платформы

Java Development Kit

Java Development Kit (сокращенно JDK) — бесплатно распространяемый компанией «Oracle Corporation» (ранее «Sun Microsystems») комплект разработчика приложений на языке Java, включающий в себя компилятор Java (javac), стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему Java (JRE). В состав JDK не входит интегрированная среда разработки на Java, поэтому разработчик, использующий только JDK, вынужден использовать внешний текстовый редактор и компилировать свои программы, используя утилиты командной строки [1].

Все современные интегрированные среды разработки приложений на Java, такие, как JDeveloper, NetBeans IDE, Sun Java Studio Creator, IntelliJ IDEA, Borland JBuilder, Eclipse, опираются на сервисы, предоставляемые JDK. Большинство из них для компиляции Java-программ используют компилятор из комплекта JDK. Поэтому эти среды разработки либо включают в комплект поставки одну из версий JDK, либо требуют для своей работы предварительной инсталляции JDK на машине разработчика.

Javac — оптимизирующий компилятор языка java, включенный в состав многих Java Development Kit (JDK).

Компилятор принимает исходные коды, соответствующие спецификации Java language specification (JLS), и возвращает байт-код, соответствующий спецификации Java Virtual Machine Specification (JVMS).

Javac написан на Java. Может вызваться непосредственно из java-программ (JSR 199) [2].

Исполнение Javac разбито на следующие фазы:

Parse — лексический и синтаксический анализ, генерация AST;

Enter — регистрация символов классов, определенных в программе;

Process annotations — обработка аннотаций;

Attribute — проверка типов, разрешение имен классов, свертка констант, вывод типов;

Flow — анализ потока управления (достижимость операций), анализ обработки исключений, проверка обращений к неинициализированным данным, проверка корректности инициализации final переменных;

Desugar — удаление синтаксического сахара (вложенные классы, классовые литералы, assert, foreach);

Generate — создание файла class.

Среда разработки IntelliJ IDEA

IntelliJ IDEA — интегрированная среда разработки программного обеспечения на многих языках программирования, в частности Java, Java Script, Python, разработанная компанией JetBrains [17].

Первая версия появилась в январе 2001 года и быстро приобрела популярность, как первая среда для Java с широким набором интегрированных инструментов для рефакторинга, которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций [17].

2 ОПИСАНИЕ ЛАБОРАТОРНОГО ПРАКТИКУМА

2.1 Педагогический адрес

Лабораторный практикум по теме «Организации работы с файлами на основе объектно-ориентированного языка Java» предназначен для самостоятельного освоения указанной темы, а также для получения основ работы с объектно-ориентированным языком программирования Java. Разработанный лабораторный практикум можно рекомендовать в качестве лабораторного модуля при усвоении дисциплин «Операционные системы» и «Языки и системы программирования» для студентов, обучающихся по профилю «Информатика и вычислительная техника», а также «Прикладная информатика в экономике».

2.2 Общие требования по созданию лабораторных практикумов

Лабораторный практикум является частью теоретической и профессиональной подготовки студентов в Федеральном государственном автономном образовательном учреждении высшего профессионального образования Российского государственного профессионально-педагогического университета [18].

Значение лабораторного практикума заключается в:

- практическом освоении студентами научно-теоретических положений изучаемой дисциплины;
- овладении техникой экспериментирования соответствующей отрасли науки;
- применении полученных знаний для решения учебно-исследовательских, а затем реальных экспериментальных и практических задач.

Целью выполнения лабораторного практикума является приобретение студентами навыков и умений, необходимых для профессиональной деятельности выпускника [19].

Дидактическими целями лабораторного практикума являются:

- экспериментальное подтверждение и проверка существующих теоретических положений;
- формирование практических умений и навыков работы с измерительными приборами, аппаратами, компьютерной техникой, лабораторными установками, технологическим оборудованием, составляющих часть профессиональной подготовки;
- формирование исследовательских умений наблюдать, сравнивать, анализировать, устанавливать зависимости, делать выводы, самостоятельно вести исследования, оформлять результаты;
- повышение познавательной активности и самостоятельности работы студентов в ходе выполнения лабораторного практикума путем организации поэтапного контроля их работы;
- усиление практической направленности образовательного процесса;
- углубление теоретических знаний и освоение приемов, методов и способов исследования объектов изучения [19].

По своему назначению лабораторные практикумы можно ***классифицировать***:

- практикумы вводные или измерительные, которые проводятся по общенаучным и общетехническим дисциплинам. Их цель — проиллюстрировать основные закономерности изучаемой науки, ознакомить студентов с техникой эксперимента, теорией погрешностей и методами обработки экспериментальных данных, с устройством и принципом работы измерительных приборов;
- практикумы, обеспечивающие накопление знаний и практических навыков при усвоении общих курсов и подготовку к изучению специальных дисциплин и методов научных исследований;

• практикумы по специальным дисциплинам и дисциплинам специализации, обеспечивающие практическую подготовку студентов, формирование навыков профессиональных и научных исследований в научной и производственной деятельности [20].

По характеру выполняемых студентами заданий лабораторные занятия подразделяются на:

• ознакомительные, предпринимаемые с целью закрепления и конкретизации изученного теоретического материала;

• аналитические, ставящие своей целью получение новой информации на основе формализованных методов;

• творческие, связанные с получением новой информации путем самостоятельно выбранных подходов решения задач.

Формами организации студентов на лабораторных работах могут быть:

1. Фронтальная — все студенты выполняют одновременно одну и ту же работу.

2. Групповая — одна и та же работа выполняется подгруппами по 25 человек.

3. Индивидуальная — каждый студент выполняет индивидуальное задание [18].

Лабораторный практикум разрабатывается на базе учебно–лабораторного оборудования кафедры, средств компьютерной поддержки, программных моделей изучаемых процессов и явлений.

Компьютерные технологии получения, хранения и преобразования информации при развитых интерфейсных системах ввода-вывода позволяет обеспечить проведение лабораторных практикумов более эффективным. Задача практикума в этом случае не просто научить студента «нажимать кнопки» по определенным правилам, но обеспечить формирование необходимых профессиональных умений, экономя время и позволяя провести опыты, которые трудно или невозможно выполнить стандартными приемами.

Программное и информационное обеспечение лабораторного практикума, особенно в части программного интерфейса, должно корректно функционировать и обладать интуитивно понятным, дружелюбным интерфейсом.

Средства мультимедиа позволяют представить учебный материал в увлекательной, динамичной форме, а инженерные конструкции, устройства, элементы — как движущиеся трехмерные объекты, тем самым в полной мере раскрывая их конструкцию и принцип действия.

Современные специализированные программные пакеты ориентированы, облегчить труд разработчика при работе с большим количеством материала в процессе создания электронного мультимедийного продукта, а также при дополнении материала в уже готовом продукте.

Это позволяет создавать хорошие электронные продукты с точки зрения психологии восприятия в условиях компьютерного обучения, а также в плане минимизации трудовых, временных, финансовых затрат.

В то же время, появляется и список требований, предъявляемых к таким лабораторным практикумам. Помимо требований к оформлению текста, появляются требования к оформлению мультимедийной информации содержимого. Рассмотрим список типовых требований более детально.

Достоверность информации. Это требование является наиболее актуальным, поскольку количество информации, содержащейся в Интернете, растёт в геометрической прогрессии, но при этом, сколько источников, столько и мнений. Весьма сложно становится находить именно достоверную информацию об интересующем направлении. Именно поэтому, при составлении лабораторного, стоит использовать только достоверные источники материала [21].

Наглядность представления. Как ни странно, но чем внешне привлекательнее электронный продукт, тем больший круг пользователей он получает. Много зависит от оформления электронного продукта, будь то фон для текстовых блоков или же часть фрагмент с представлением, скажем, ви-

деоролика. Первую оценку от читателя мультимедийный продукт получает за внешний вид [21].

Удобство использования. Так как лабораторный практикум, как правило, состоит из большого количества страниц, то самое главное на этапе подготовки правильно произвести структуризацию материала, чтобы читатель мог без лишнего труда найти необходимую ему информацию, при этом, не пролистывая страницы и разделы, которые ему конкретно в данный момент не нужны. Для этого необходимо обеспечить максимальное удобство навигации по страницам электронного продукта [21].

2.3 Описание средства реализации практикума и модуля

Для представления лабораторного практикума по теме: «Организация работы с файлами в ОС Windows», был выбран формат PDF.

PDF — это стандартный формат файлов, специально предназначенный для обмена готовыми к печати документами в виде электронных данных, при котором отправителю и получателю не требуется дополнительной договоренности для обработки информации и получения требуемых результатов в тираже. Фактически он является цифровым эквивалентом цвет оделённых фотоформ.

Формат (PDF) представляет собой универсальный файловый формат, документа независимо от того, на какой из множества платформ и в каком из множества приложений такой документ создавался. Формат Adobe PDF считается признанным общемировым стандартом в области тиражирования и обмена надежно защищенными электронными документами и бланками. Файлы Adobe PDF имеют небольшой размер, и они самодостаточны; они допускают совместную работу, просмотр и печать с помощью бесплатной программы Adobe Reader.

DOCX Microsoft Word — текстовый процессор, предназначенный для создания, просмотра и редактирования текстовых документов, с локальным применением простейших форм таблично-матричных алгоритмов.

SunRav BookEditor — представляет собой многофункциональное средство для создания и просмотра электронных книг реагирующих на действия пользователей. С его помощью можно создавать документы в виде EXE файлов, CHM, HTML, PDF форматах, а также в любых других, используя шаблоны. В книгах можно использовать всю мощь современных мультимедийных форматов: аудио и видео файлы, изображения PNG, JPEG, GIF включая анимированные, Flash, любые OLE объекты и т.д. SunRav BookEditor может создавать книги в разных форматах, представленных ниже.

ExeBook — это программа, которая создает электронные книги в формате «exe», также может быть использована для создания фотоальбомов. В нее есть возможность добавлять ссылки, анимации и картинки, используя разные форматы файлов. Также поддерживает такой формат, как HTML. Для создания книги необходимо добавить в нее нужные файлы и настроить параметры.

EBooksWriter LITE — программа, использующаяся для создания книг в электронном виде. В программе присутствуют шаблоны, которые значительно упрощают создание книги, а также можно поворотно использовать картинки и объекты из других проектов. Также есть возможность защитить созданную книгу от копирования пользователей.

HTML — позволяет поместить наработанный материал на сайт. Представляет набор HTML страниц, изображений и несколько служебных файлов, с помощью которых можно создать дерево содержания. Возможно создание 2 типов HTML книг: с фреймами и без фреймов.

Инструменты просмотра PDF формата

Foxit Reader — бесплатное кроссплатформенное прикладное программное обеспечение для просмотра электронных документов в стандарте PDF с любого носителя. Имеет возможность интегрироваться в интернет

браузеры и просматривать документ, а также создавать закладки, а также защищать информацию от копирования. Имеет традиционный интерфейс (рисунок 9).

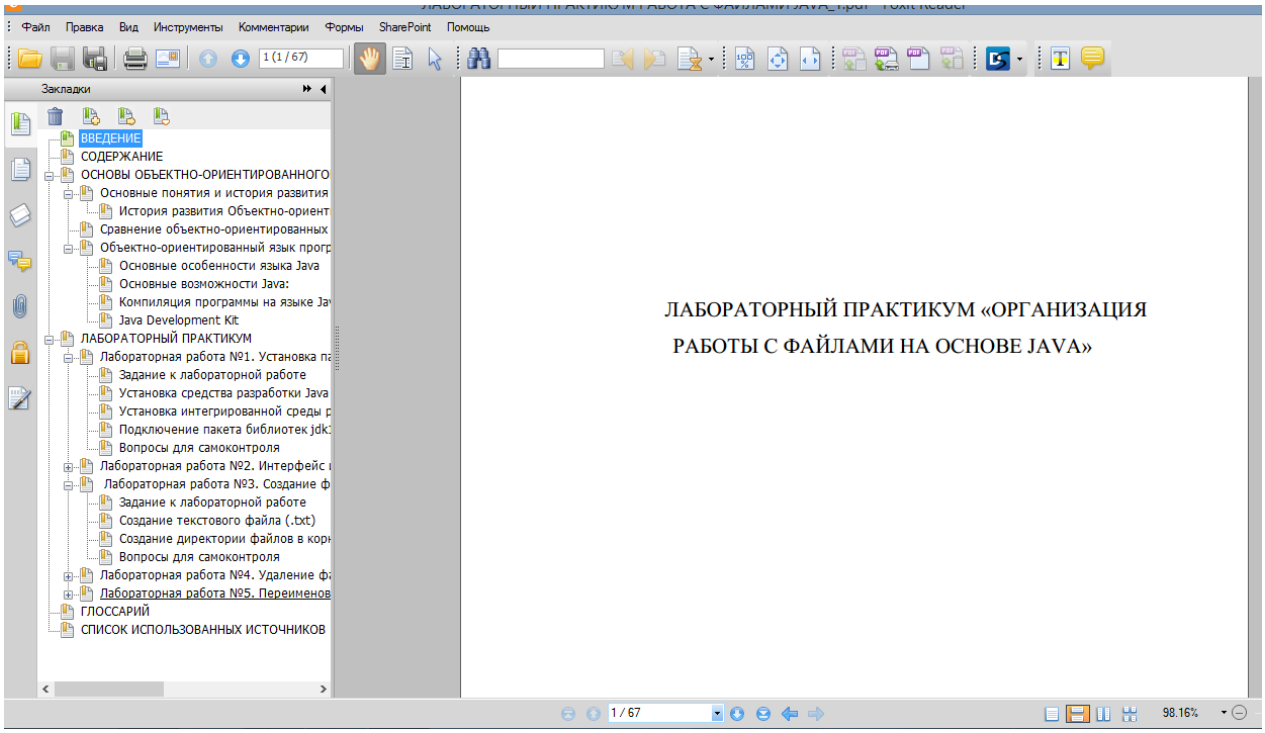


Рисунок 9 — Интерфейс Foxit Reader

Adobe Reader — бесплатная программа для просмотра и печати электронной документации в формате PDF и всегда поддерживает все его модификации. Интерфейс не включает инструментарий для редактирования документов, в окне программы присутствуют минимум кнопок и панелей. Есть возможность просмотра документов в разных окнах, то есть работать с несколькими документами одновременно. Позволяет легко и быстро ознакомиться с руководствами, описанием программных продуктов, рекламными буклетами и презентациями с полным набором применяемых данных (рисунок 10).

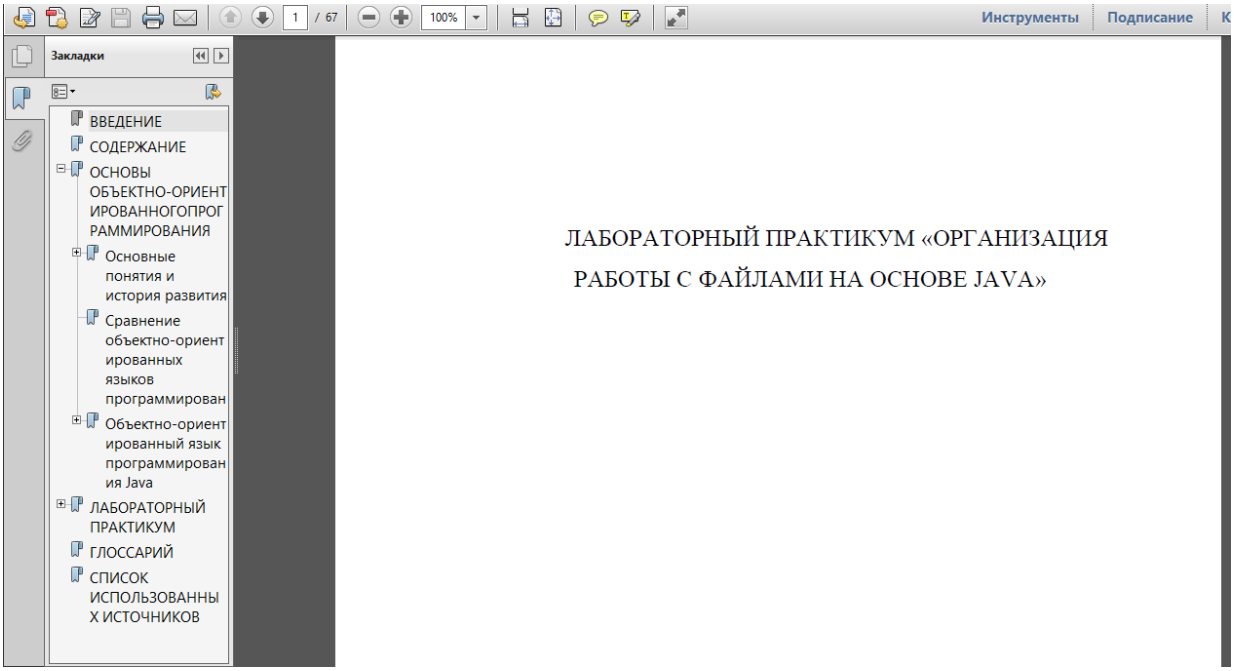


Рисунок 10 — Интерфейс Adobe Reader

PDF-XChange Viewer — бесплатный, быстрый и удобный инструмент для просмотра, редактирования и печати PDF файлов. Программа имеет платную и бесплатную версию. Платная версия имеет расширенные возможности по работе с закладками, редактированию документов. Интерфейс PDF-XChange Viewer оформлен в современном стиле и очень элегантен рисунок (рисунок 11).

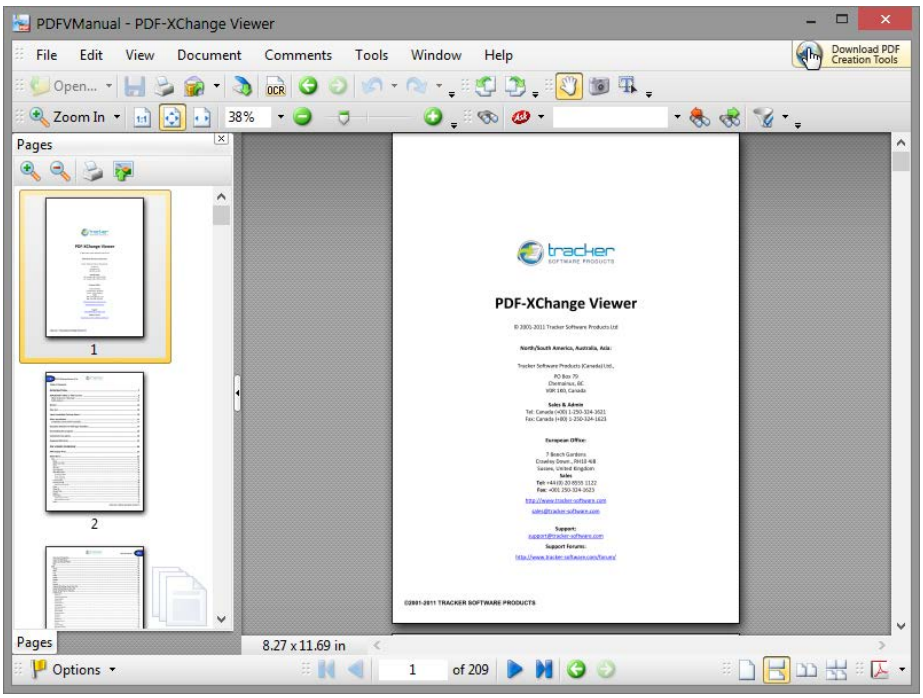


Рисунок 11 — Интерфейс PDF-XChange Viewer

Особенности формата PDF:

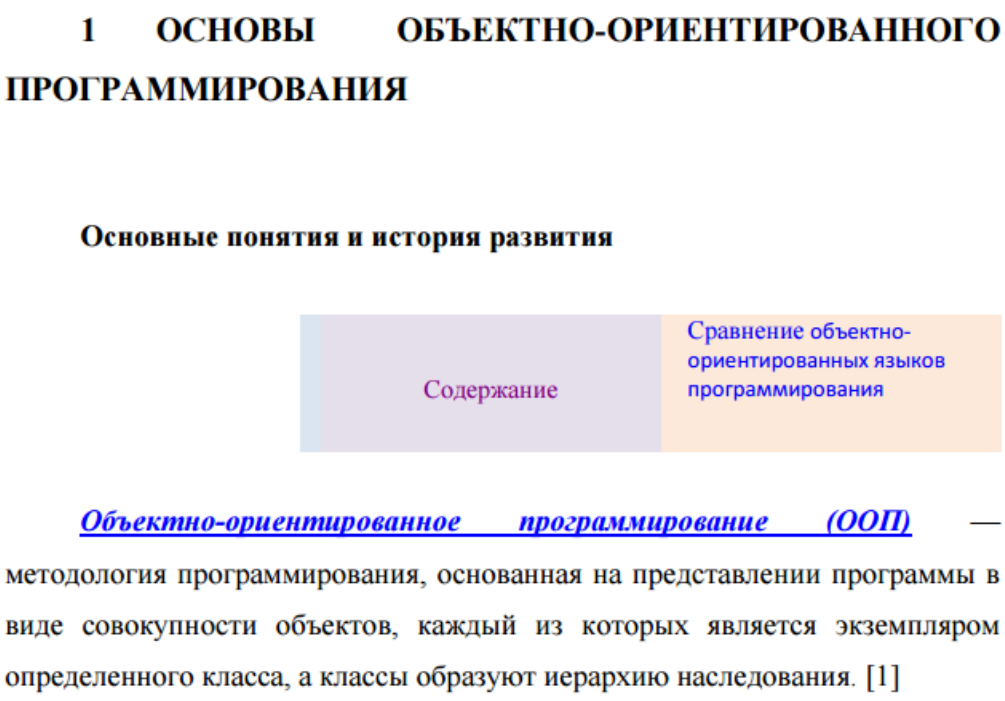
1. Стандартизованность и популярность: открывается на любых устройствах с любыми операционными системами ровно в том виде, в котором был создан;

2. Средство просмотра PDF формата, Adobe Acrobat Reader, часто предустановлено на компьютер, если устройство поставлялось с операционной системой. Если же нет — он доступен для скачивания с сайта разработчика Adobe Systems и это совершенно бесплатно;

3. Занимает мало места на жестком диске, потому что поддерживает множество алгоритмов компрессии;

4. Безопасность: пользователь может настроить параметры безопасности для своего PDF файла, например, запрет печати, запрет редактирования, использование электронной подписи для определения подлинности документа.

В лабораторном практикуме присутствуют гиперссылки (рисунок 12) и закладки (рисунок 13), для удобного использования практикума обучающимися.



1 ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

Основные понятия и история развития

Содержание

Сравнение объектно-ориентированных языков программирования

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования. [1]

Рисунок 12 — Гиперссылки

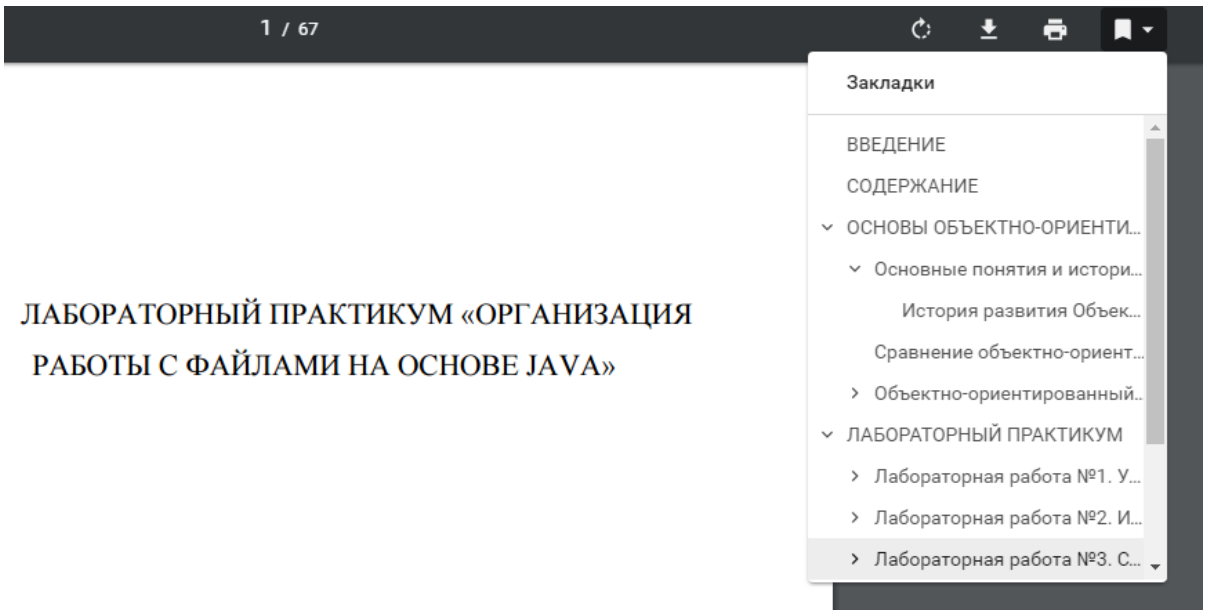


Рисунок 13 — Закладки

2.4 Структура лабораторного практикума и реализация навигации

Лабораторный практикум разработан в качестве учебного материала, который может использоваться в самостоятельной работе студентами профессионально-педагогического вуза, изучающими объектно-ориентированные языки программирования. При прохождении лабораторного практикума студенты знакомятся с базовыми понятиями, используемыми в языке программирования Java.

В результате тщательного сбора, анализа учебного материала, его методической переработки была разработана структура лабораторного практикума, которая делилась на два основных раздела — ***теоретический*** и ***раздел лабораторных работ***.

Выделение теории в отдельный раздел позволяет студентам еще до выполнения лабораторных работ изучить основные понятия и проконтролировать уровень их усвоения. Это повышает эффективность выполнения самих лабораторных работ.

В ***теоретический раздел*** лабораторного практикума входят три темы: «Основные понятия и история развития», «Сравнение объектно-

ориентированных языков программирования», «Объектно-ориентированный язык программирования Java». Материал для лабораторного практикума трансформирован в форму, удобную для его усвоения студентами. Длительность теоретического раздела лабораторного практикума представлена в таблице 5.

Таблица 5 — Время, отведенное на изучение тем теоретического раздела

Темы теоретического раздела	**Время изучения (мин)**
1. Основные понятия и история развития	45
2. Сравнение объектно-ориентированных языков программирования	45
3. Объектно-ориентированный язык программирования Java	90

Лабораторные работы по организации работы с файлами на основе Java включают следующим темы:

1. Лабораторная работа №1: «Установка пакета приложений Java для ОС Windows»;
2. Лабораторная работа №2: «Интерфейс интегрированной среды разработки IntelliJ IDEA»;
3. Лабораторная работа №3: «Создание файлов с помощью языка программирования Java»;
4. Лабораторная работа №4: «Удаление файлов с помощью языка программирования Java»;
5. Лабораторная работа №5: «Переименование и копирование файлов с помощью языка программирования Java».

Длительность каждой лабораторной работы представлена в таблице 6.

Таблица 6 — Время, отведенное на выполнение лабораторных работ

Наименование	**Время изучения (мин)**
1. Установка пакета приложений Java для ОС Windows	45
2. Интерфейс интегрированной среды разработки IntelliJ IDEA	45
3. Создание файлов с помощью языка программирования Java	90
4. Удаление файлов с помощью языка программирования Java	90
5. Переименование и копирование файлов с помощью языка программирования Java	90
Итого:	360

Структура лабораторного практикума представлена на рисунке 14.

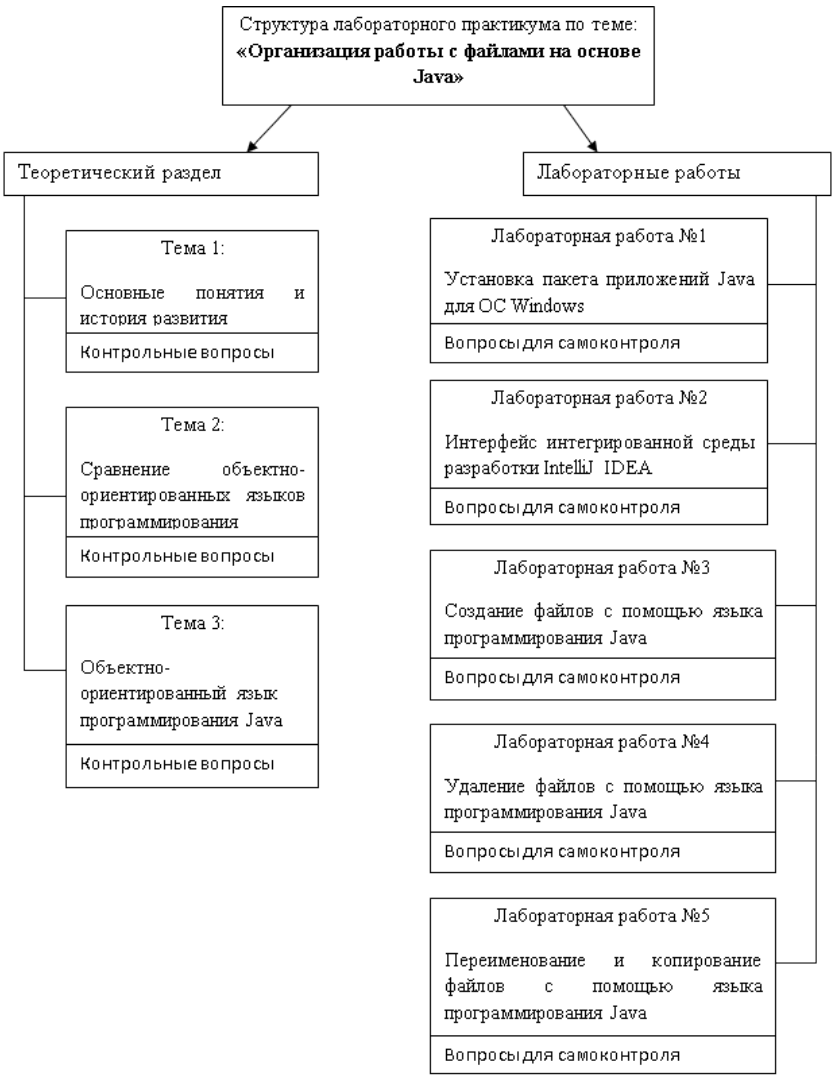


Рисунок 14 — Структура лабораторного практикума

2.5 Описание лабораторных работ

В разделе «Лабораторный практикум» содержатся 5 лабораторных работ:

1. Лабораторная работа №1: «Установка пакета приложений Java для ОС Windows».
2. Лабораторная работа №2: «Интерфейс интегрированной среды разработки IntelliJ IDEA».
3. Лабораторная работа №3: «Создание файлов с помощью языка программирования Java».
4. Лабораторная работа №4: «Удаление файлов с помощью языка программирования Java».
5. Лабораторная работа №5: «Переименование и копирование файлов с помощью языка программирования Java».

Лабораторные работы выполняются последовательно, с первой темы. После завершения выполнения лабораторных работ присутствуют контрольные вопросы, которые направлены на выявления усвоения пройденного материала.

Лабораторная работа №1. Установка пакета приложений Java для ОС Windows

Цель: Установить и настроить приложения: Java Development Kit и интегрированную среду разработки IntelliJ IDEA.

Оборудование и программное обеспечение: Персональный компьютер с установленной операционной системой Windows 7 (Windows 8), имеющий следующие характеристики: процессор Intel Core CPU (или аналогичный), оперативная память – не менее 2048 Mb.

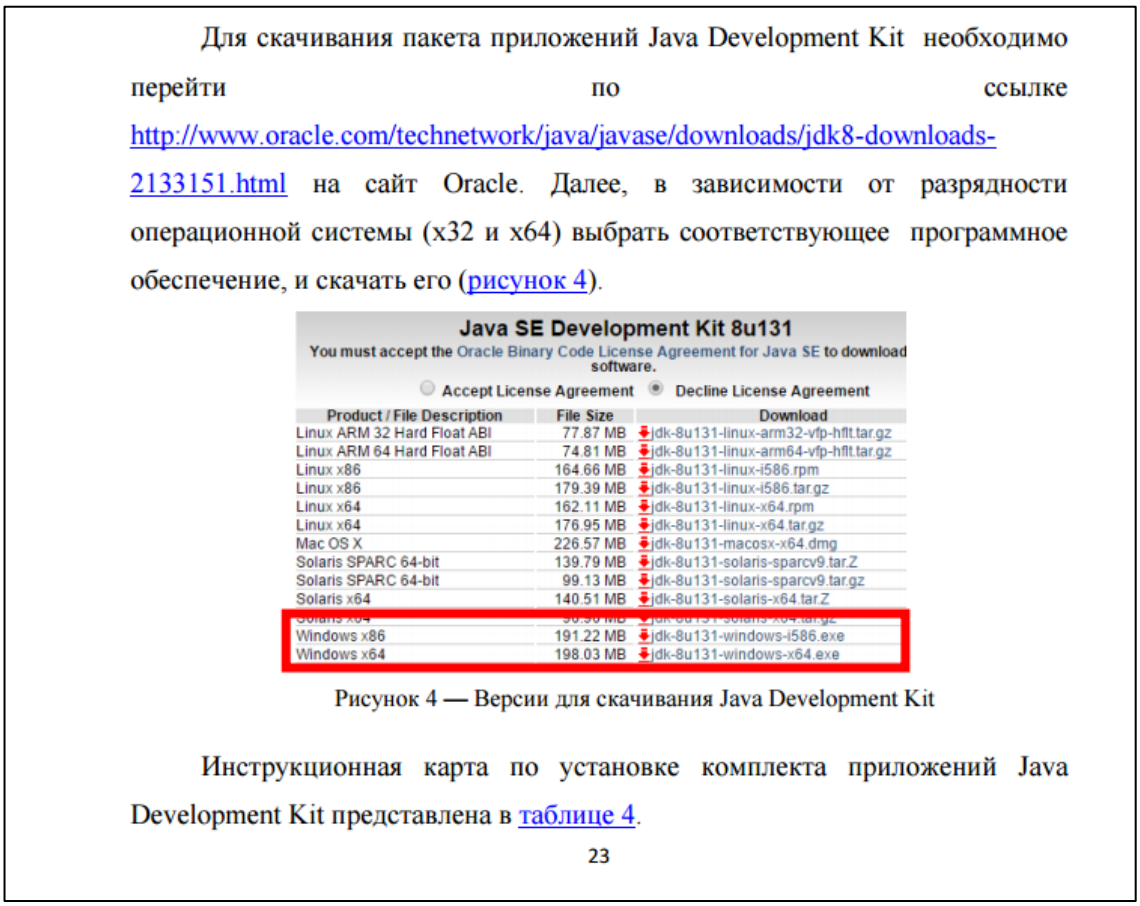
Задание к лабораторной работе

1. Установить комплект приложений Java Development Kit, включающий в себя компилятор Java (javac).

2. Установить на виртуальную машину интегрированную среду разработки IntelliJ IDEA, работающую со многими языками программирования, в том числе и Java.

3. Настроить среду разработки IntelliJ IDEA подключив пакет библиотек jdk1.8.0_121.

В первой лабораторной работе обучаемый должен подготовить свой персональный компьютер к работе с лабораторным практикумом. Перейти по ссылкам, выбрать необходимое программное обеспечение (ПО), скачать Java Development Kit и IntelliJ IDEA, далее используя инструкционные карты установить их на компьютер. Также их нужно проверить после установки на работоспособность. Затем настроить среду разработки IntelliJ IDEA подключив пакет библиотек jdk1.8.0_121 (рисунок 15), (рисунок 16), (рисунок 17), (рисунок 18), (рисунок 19), (рисунок 20).



Для скачивания пакета приложений Java Development Kit необходимо перейти по ссылке http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html на сайт Oracle. Далее, в зависимости от разрядности операционной системы (x32 и x64) выбрать соответствующее программное обеспечение, и скачать его (рисунок 4).

Java SE Development Kit 8u131

You must accept the Oracle Binary Code License Agreement for Java SE to download software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.87 MB	jdk-8u131-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.81 MB	jdk-8u131-linux-arm64-vfp-hflt.tar.gz
Linux x86	164.66 MB	jdk-8u131-linux-i586.rpm
Linux x86	179.39 MB	jdk-8u131-linux-i586.tar.gz
Linux x64	162.11 MB	jdk-8u131-linux-x64.rpm
Linux x64	176.95 MB	jdk-8u131-linux-x64.tar.gz
Mac OS X	226.57 MB	jdk-8u131-macosx-x64.dmg
Solaris SPARC 64-bit	139.79 MB	jdk-8u131-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.13 MB	jdk-8u131-solaris-sparcv9.tar.gz
Solaris x64	140.51 MB	jdk-8u131-solaris-x64.tar.Z
Windows x86	191.22 MB	jdk-8u131-windows-i586.exe
Windows x64	198.03 MB	jdk-8u131-windows-x64.exe

Рисунок 4 — Версии для скачивания Java Development Kit

Инструкционная карта по установке комплекта приложений Java Development Kit представлена в таблице 4.

23

Рисунок 15 — Скачивание Java Development Kit

Таблица 4 — Инструкционная карта по установке Java Development Kit

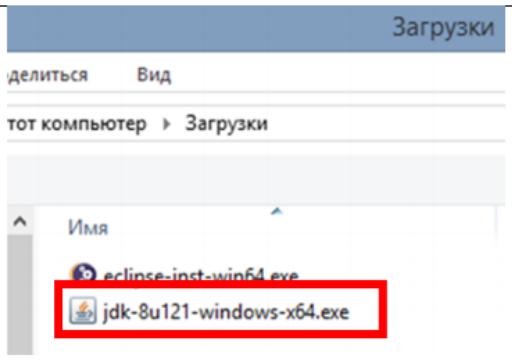
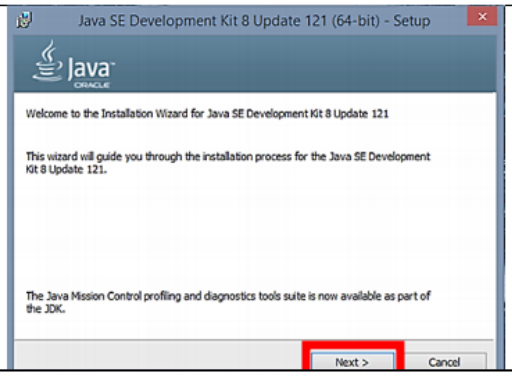
Этапы выполнения	Экранные формы
1. В загрузках находим скаченный ранее пакет Java Development Kit и запускаем его.	
2. После запуска установщика на экране будет отображаться окно установки, продолжим установку нажатием кнопки ***Next***.	

Рисунок 16 — Пошаговый порядок действий при установке Java Development Kit

Далее комбинацией клавиш (win+r) открываем интерпретатор команд Windows CMD и в командной строке пишем ***java –version*** и нажимаем Enter. Таким образом, мы проверяем, работает ли наше средство разработки и правильно ли оно установлено. Если выводится версия Java, и нет сообщений об ошибке, то все было установлено правильно (рисунок 6).

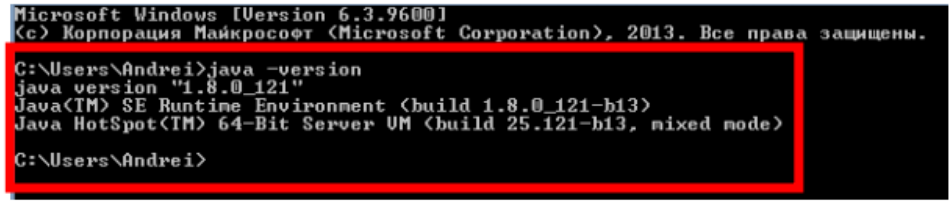


Рисунок 6 — Проверка работы Java Development Kit

Рисунок 17 — Проверяем правильность установки Java Development Kit

Установка интегрированной среды разработки IntelliJ IDEA

Установщик находится на сайте JetBrains. Переходим по ссылке http://www.jetbrains.com/idea/download/#section=windows на сайт и скачиваем бесплатную версию Intellij idea (рисунок 7).

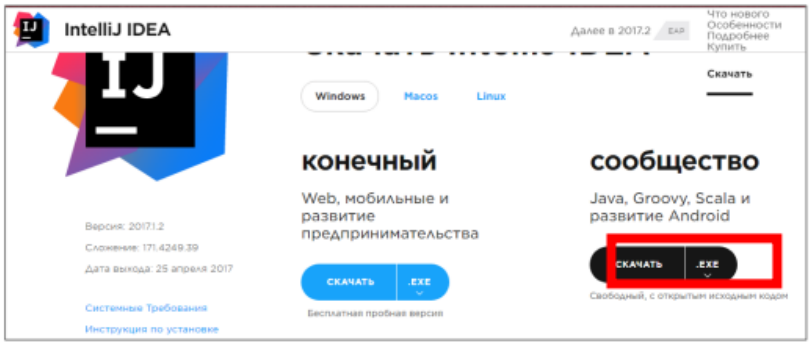


Рисунок 7 — Бесплатная версия Intellij idea

Инструкционная карта по установке интегрированной среды разработки Intellij idea представлена в таблице 5.

Рисунок 18 — Скачивание IntelliJ IDEA

Таблица 5 — Инструкционная карта по установке Intellij idea

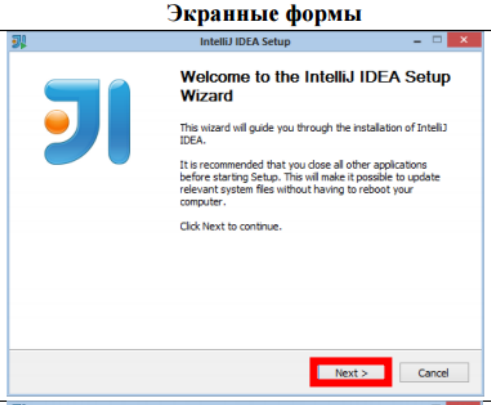
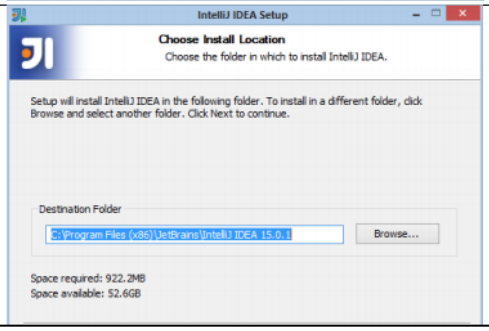
Этапы выполнения	Экранные формы
1. Запускаем установщик Intellij idea. Далее на экране появляется окно установки, продолжим установку нажатием кнопки ***Next***.	
2. На следующем экране нам предложат директорию для установки. Оставим ее неизменной. Продолжим установку нажатием кнопки ***Next***.	

Рисунок 19 — Пошаговый порядок действий при установке IntelliJ IDEA

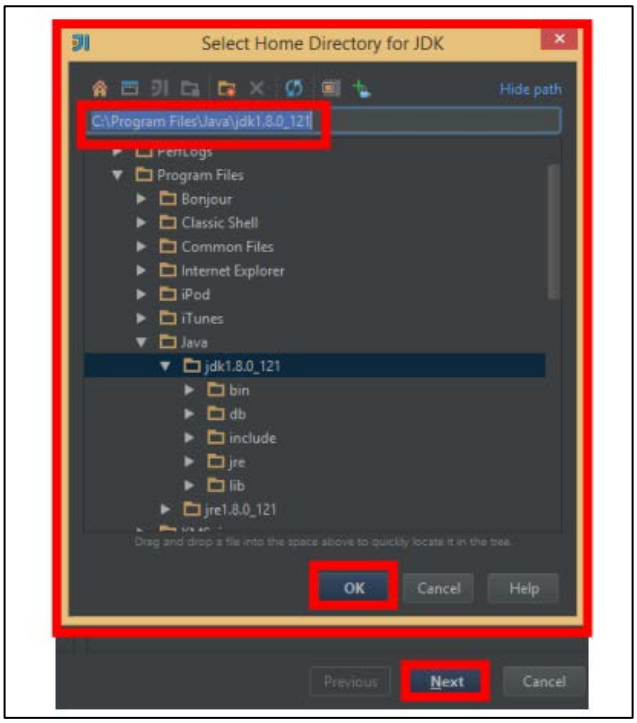


Рисунок 20 — Подключение библиотек в среде разработки IntelliJ IDEA

В конце каждой лабораторной работы присутствуют вопросы для самоконтроля (рисунок 21).

Вопросы для самоконтроля

1. Что такое Java Development Kit?
2. Что такое IntelliJ IDEA?
3. Расскажите, как поэтапно происходит добавление библиотек в Java?

Рисунок 21 — Контроль после завершения Лабораторной работы

Лабораторная работа №2. Интерфейс интегрированной среды разработки IntelliJ IDEA

Цель: Познакомиться с компонентами интегрированной среды разработки IntelliJ IDEA и особенностями ее интерфейса.

Оборудование и программное обеспечение: Персональный компьютер с установленной операционной системой Windows 7 (Windows 8), имеющий следующие характеристики: процессор Intel Core CPU (или аналогичный), оперативная память — не менее 2048 Mb.

Задание к лабораторной работе

1. Изучение интерфейса интегрированной среды разработки IntelliJ IDEA на основе создания программы на языке программирования Java.

2. Изучение интерфейса интегрированной среды разработки IntelliJ IDEA на основе создания одного объекта с полями имя и возраст на языке программирования Java.

Во второй лабораторной работе, обучаемый должен познакомиться с интерфейсом IntelliJ IDEA прочитав теорию, создать Java class и написать первую программу на Java, а также создать объект с заданными полями (рисунок 22).

Интерфейс интегрированной среды разработки IntelliJ IDEA

Основное меню среды IntelliJ IDEA

Главное меню IntelliJ IDEA состоит из следующих пунктов (рисунок 11):

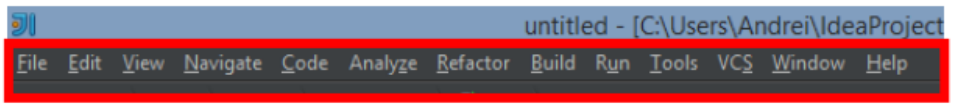


Рисунок 11 — Меню IntelliJ IDEA

File (Файл) — команда для работы с проектами, модулями, файлами.

Edit (Редактирование) — стандартные команды для редактирования.

View (Просмотр) — команды для отображения различных элементов интерфейса.

Navigate (Навигация) — команды навигации по проекту.

Code (Код) — вспомогательные команды для написания кода.

Analyze (Анализ) — команды для анализа кода.

Refactor (Переработка кода) — команды для изменения внутренней структуры программы, не затрагивая ее внешнего поведения.

Build (Сборка) — команды для компиляции и сборки проекта.

Run (Запуск) — запуск и отладка приложений.

Tools (Утилиты) — набор различных вспомогательных утилит.

Windows (Окно) — работа с окнами в IntelliJ IDEA.

Рисунок 22 — Интерфейс программы IntelliJ IDEA

Необходимо зайти в панель навигации IntelliJ IDEA и создать Java class (рисунок 23).

Заходим в панель навигации **IntelliJ IDEA** и выделяем папку **SRC** (папка, в которой будут отображаться, и располагаться все созданные нами проекты). Далее нажимаем на ней правой кнопкой мыши и в появившемся окне выбираем **New** (создать новый проект). В новом окне выбираем **Java Class** (рисунок 16).

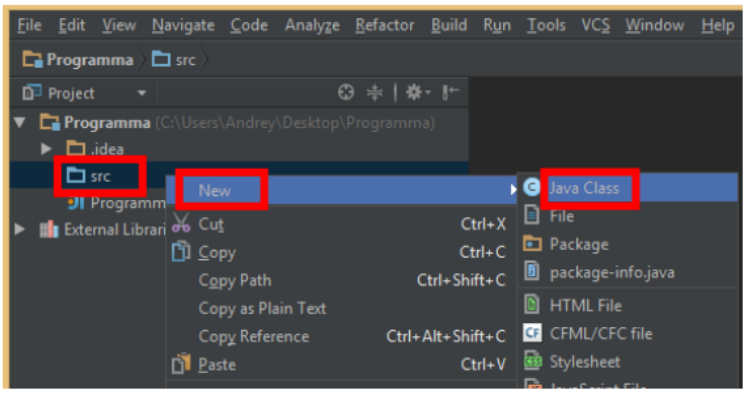


Рисунок 16 — Создаем новый Java class

Рисунок 23 — Создание Java class

Задать имя программы и начать описание кода программы (рисунок 24), (рисунок 25).

Далее появляется окно с именем **Create New Class**, в поле **Name** вписываем имя нашей программы ***«HelloWorld»*** и нажимаем на кнопку **OK** (рисунок 17).

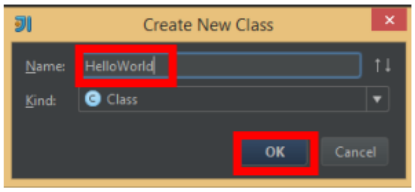


Рисунок 17 —Вписываем имя класса

Начинаем создавать программу с имени class ***«HelloWorld»*** (рисунок 18).

Рисунок 18 — Задаем имя class

Рисунок 24 — Написание имени программы

Также используем метод ***println*** (передается любое значение, как правило, строка, которое необходимо вывести на консоль), для вывода текста программы (рисунок 20).

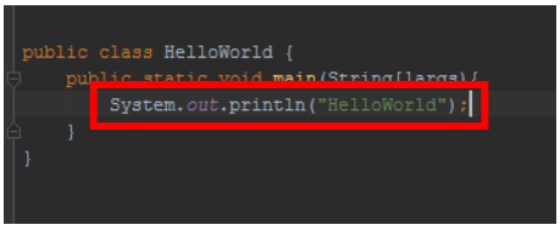


Рисунок 20 — Метод println

Завершив написание кода программы ***«HelloWorld»*** мы должны ее скомпилировать. Переходим в главное меню и выбираем пункт **Run**. В новом окне также выбираем пункт **Run**, таким образом, запустив компиляцию

Рисунок 25 — Описание кода программы

Скомпилировать программу и вывести результат на консоль (рисунок 26).

После того, как мы скомпилировали программу ее нужно запустить. Переходим в главное меню и выбираем пункт **Run *«HelloWorld»*** (рисунок 22).

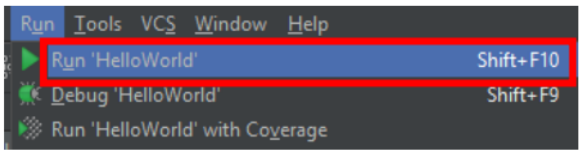


Рисунок 22 — Запуск программы HelloWorld

37

Далее после запуска программы выводится результат (рисунок 23).

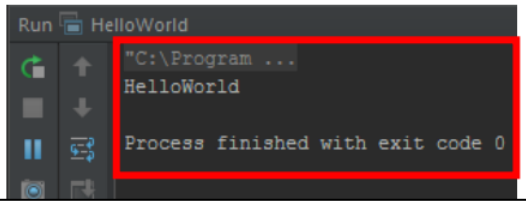


Рисунок 26 — Компиляция и вывод результатов

Лабораторная работа №3. Создание файлов с помощью языка программирования Java

Цель: Научиться создавать файлы на языке программирования Java в среде разработки IntelliJ IDEA.

Оборудование и программное обеспечение: Персональный компьютер с установленной операционной системой Windows 7 (Windows 8), имеющий следующие характеристики: процессор Intel Core CPU (или аналогичный), оперативная память — не менее 2048 Mb.

Задание к лабораторной работе

1. Создать текстовый (file.txt) файл по адресу C:\Users\Пользователь на языке программирования Java в интегрированной среде разработки IntelliJ IDEA.

2. Создать директорию со структурой (папка/папка/папка/.txt) и добавить запись в файле txt.

В третьей лабораторной работе обучаемый должен создавать txt файлы на языке программирования Java в интегрированной среде разработки IntelliJ IDEA, а также директории файлов.

Для успешного создания файла необходимо ознакомиться с теорией по созданию текстового файла (рисунок 27).

Методические указания

Создание текстового файла (.txt)

Для создания нового файла в Java чаще всего используется класс java.io.File (набор методов для операций с файлами). Во время инициализации (создание, активация, подготовка к работе, определение параметров) объекта File мы должны предоставить ему имя файла, а затем создать сам файл вызовом метода ***createNewFile()*** (создание новых файлов в Java).

Этот метод используется для создания новых файлов в Java, он возвращает значение типа boolean: true — если файл был создан успешно, false — если файл создать не удалось.

Создать файл в Java можно одним из трех способов, передав в объект File:

1. Абсолютный путь (полный);
2. Только указать имя файла;
3. Указать относительный путь (в этом случае объект файла

Рисунок 27 — Теория по созданию файла

Зайти в панель навигации, создать новый Java Class и начать описание кода программы (рисунок 28).

Далее появляется окно с именем ***Create New Class***, в поле ***Name*** вписываем имя нашей программы ***«CreateNewFile»*** и нажимаем на кнопку ***OK*** (рисунок 32).

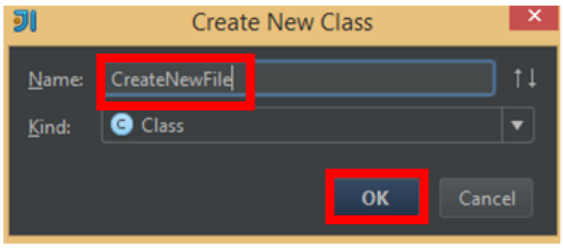


Рисунок 32 — Вписываем имя класса

Начинаем написание кода программы с импортирования библиотек, которые будут использоваться в нашей программе (рисунок 33).

Рисунок 28 — Создание Java Class и описание кода программы

Скомпилировать и запустить программу (рисунок 29).

После завершения написания кода программы ее необходимо скомпилировать. Переходим в главное меню и выбираем пункт ***Run***. В новом окне также выбираем пункт ***Run***, таким образом, запустив компиляцию программы (рисунок 21).

После того, как мы скомпилировали программу ее нужно запустить. Для этого мы используем комбинацию клавиш на клавиатуре ***Shift + F10***.

После запуска программы выводится результат на консоль (рисунок 38).

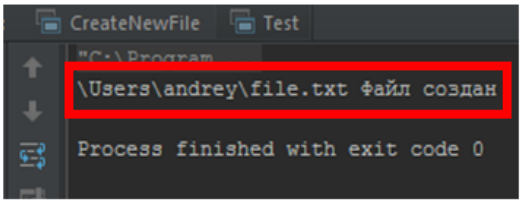


Рисунок 38 — Вывод результата

44

Рисунок 29 — Компиляция и вывод результатов

Лабораторная работа №4. Удаление файлов с помощью языка программирования Java

Цель: Научиться удалять файлы с помощью языка программирования Java.

Оборудование и программное обеспечение: Персональный компьютер с установленной операционной системой Windows 7 (Windows 8), имеющий следующие характеристики: процессор Intel Core CPU (или аналогичный), оперативная память — не менее 2048 Mb.

Задание к лабораторной работе

1. Удалить файл (file.txt), используя прямой полный путь к файлу.
2. Удалить файл, используя только (имя файла) из корневой папки проекта.

В четвёртой лабораторной работе обучаемый должен удалить файлы, используя прямой путь к файлу, а также используя только имя файла.

Для успешного удаления файла необходимо ознакомиться с теорией по удалению текстового файла (рисунок 30).

Методические указания

Удаление (file.txt), используя прямой полный путь к файлу

Перед тем, как приступить к написанию программы ознакомимся с краткой теоритической справкой о методе ***java.io.File delete***() (удаление файла или каталога), который мы будем использовать в нашей программе.

Для удаления файлов или папок в Java используется метод ***java.io.File delete()***. Он возвращает true, если файл удалился успешно и возвращает false, если указанный для удаления файл не существует или не может быть удален.

Если вы пытаетесь удалить папку, то этот метод проверяет указанную папку на пустоту. Если папка пуста, то она удаляется, если в папке что-то есть, то метод ***delete()*** просто возвращает false, то есть папка не удаляется.

Рисунок 30 — Теория, по удалению файлов используя полный путь к файлу

Также необходимо зайти в панель навигации, создать новый Java Class, задать ему имя и начать описание кода программы (рисунок 31).

Заходим в панель навигации ***IntelliJ IDEA*** и выделяем папку ***SRC*** (папка, в которой будут отображаться, и располагаться все созданные нами проекты). Далее нажимаем на ней правой кнопкой мыши и в появившемся окне выбираем ***New*** (создать новый проект). В новом окне выбираем ***Java Class*** (рисунок 16).

Далее появляется окно с именем ***Create New Class***, в поле ***Name*** вписываем имя нашей программы ***«DeleteFileJava»*** и нажимаем на кнопку ***OK*** (рисунок 48).

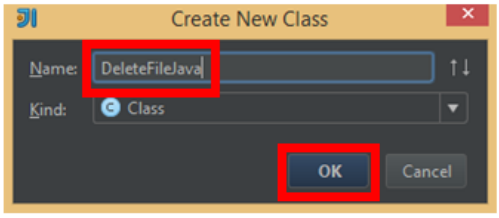


Рисунок 48 — Вписываем имя класса

Начинаем написание кода программы с импортирования библиотеки, которая будет использоваться в нашей программе (рисунок 49).

Рисунок 31 — Создание Java Class

Также нужно описать код программы и скомпилировать ее для запуска (рисунок 32), (рисунок 33).

удален и файла не обнаружено. Используем файл с именем ***(file.txt)***, который мы создавали ранее в Лабораторной работе №3 (Задание 1) (рисунок 51).

```
//удаление с использованием полного пути к файлу
File file = new File("/Users/andrey/file.txt");
if (file.delete()) {
    System.out.println("/Users/andrey/file.txt файл удален");
} else System.out.println("Файла /Users/andrey/file.txt не обнаружено");
}
```

Рисунок 51 — Прописываем путь

После завершения написания кода программы ее необходимо скомпилировать. Переходим в главное меню и выбираем пункт ***Run***. В новом окне также выбираем пункт ***Run***, таким образом, запустив компиляцию программы (рисунок 21).

После того, как мы скомпилировали программу ее нужно запустить. Для этого мы используем комбинацию клавиш на клавиатуре ***Shift + F10***.

После запуска программы выводится результат на консоль (рисунок

Рисунок 32 — Описание кода программы и компиляция

После запуска программы выводится результат на консоль (рисунок 52).

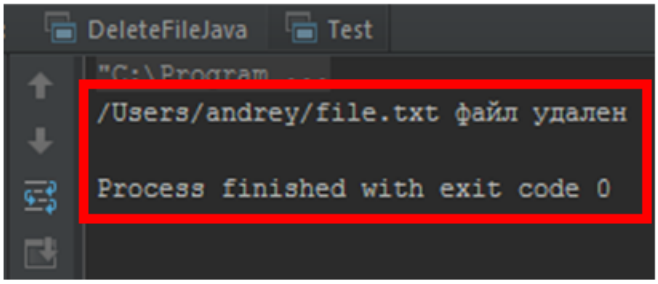


Рисунок 52 — Вывод результата

Рисунок 33 — Вывод результатов

Лабораторная работа №5. Переименование и копирование файлов с помощью языка программирования Java

Цель: Научиться переименовывать и копировать файлы с помощью языка программирования Java.

Оборудование и программное обеспечение: Персональный компьютер с установленной операционной системой Windows 7 (Windows 8), имеющий следующие характеристики: процессор Intel Core CPU (или аналогичный), оперативная память — не менее 2048 Mb.

Задание к лабораторной работе

1. Переименовать файл с именем (lab.txt) в файл с именем (lab6.txt), используя абсолютный путь к файлу.
2. Копировать файл с именем (and.txt).

В пятой лабораторной работе обучаемый должен переименовывать файлы, используя абсолютный путь к файлу (прямой), а также копировать файлы.

Для успешного переименования необходимо ознакомиться с теорией по переименованию текстового файла (рисунок 33).

Переименование файла с именем (lab.txt) в файл с именем (lab6.txt), используя абсолютный путь к файлу

Класс File, определенный в пакете java.io, не работает напрямую с потоками. Его задачей является управление информацией о файлах и

54

каталогах. Хотя на уровне операционной системы файлы и каталоги отличаются, но в Java они описываются одним классом File.

Метод ***renameTo(File dest)*** может быть использован для переименования или перемещения файла в Java. Этот метод возвращает true, если переименование файлов успешно, иначе она возвращает false. Некоторые операции зависят от платформы, например переименование, может потерпеть неудачу, если вы перемещаете файл из одной файловой системы в другую или, если файл с тем же именем уже существует в пункте

Рисунок 33 — Теория, по переименованию текстового файла

Также необходимо зайти в панель навигации, создать новый Java Class, задать ему имя и начать описание кода программы (рисунок 34).

Далее появляется окно с именем ***Create New Class***, в поле ***Name*** вписываем имя нашей программы ***«RenameFileJava»*** и нажимаем на кнопку ***OK*** (рисунок 61).

55

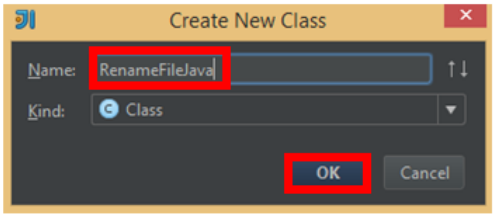


Рисунок 61 — Вписываем имя класса

Рисунок 34 — Создание Java Class

Также нужно описать код программы и скомпилировать ее для запуска (рисунок 35).

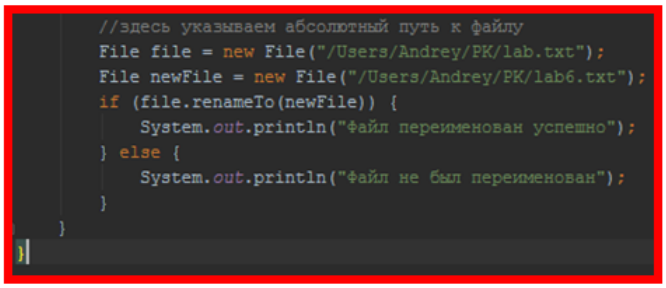


Рисунок 64 — Описанный класс RenameFileJava

После завершения написания кода программы ее необходимо скомпилировать. Переходим в главное меню и выбираем пункт ***Run***. В новом окне также выбираем пункт ***Run***, таким образом, запустив компиляцию программы (рисунок 21).

После того, как мы скомпилировали программу ее нужно запустить. Для этого мы используем комбинацию клавиш на клавиатуре ***Shift*** + ***F10***.

Рисунок 35 — Описание и компиляция программы

Далее идет вывод результатов на консоль рисунок (36).

После запуска программы выводится результат на консоль (рисунок 65).

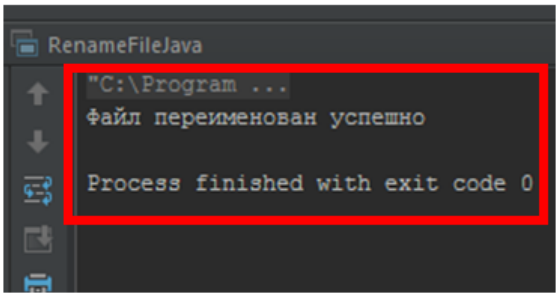


Рисунок 65 — Вывод результата

Рисунок 36 — Вывод результатов

2.6 Рекомендации по самостоятельному обучению с использованием лабораторного практикума

При самостоятельном изучении лабораторного практикума ***«Организация работы с файлами на основе Java»*** рекомендуется перед началом выполнения лабораторных работ скачать и установить на персональный ком-

пьютер виртуальную машину ***VMware Player*** (бесплатный инструмент, который дает пользователям возможность запускать виртуальную машину на компьютерах, работающих под Linux или Windows). Это желательно осуществить для того, чтобы опробовать и протестировать программы, которые потребуется установить при работе с лабораторным практикумом, а также не нагружать и засорять операционную систему своего персонального компьютера.

ЗАКЛЮЧЕНИЕ

Появление языков программирования в современном мире, позволило человеку общаться с персональным компьютером, решать сложные задачи во многих сферах жизни. Язык программирования сыграл огромную роль в развитии современных технологий и продолжает развиваться.

В настоящее время язык программирования Java привлек более девяти миллионов разработчиков по всему миру.

Он объектно-ориентирован и имеет богатую библиотеку классов и в то же время его не сложно осваивать. Его цикл разработки приложений сокращен за счет того, что система построена на основе интерпретатора.

В результате тщательного сбора, анализа учебного материала, его методической переработки была разработана структура лабораторного практикума, которая делилась на два основных раздела — ***теоретический*** и ***раздел лабораторных работ***.

В теоретический раздел лабораторного практикума входят три темы: «Основные понятия и история развития», «Сравнение объектно-ориентированных языков программирования», «Объектно-ориентированный язык программирования Java».

Лабораторные работы включают следующие темы: «Установка пакета приложений Java для ОС Windows», «Интерфейс интегрированной среды разработки IntelliJ IDEA», «Создание файлов с помощью языка программирования Java», «Удаление файлов с помощью языка программирования Java», «Переименование и копирование файлов с помощью языка программирования Java».

Лабораторный практикум был разработан в качестве дополнительного учебного материала, который может использоваться в самостоятельной работе студентами профессионально-педагогического вуза, изучающих дисциплину «Операционные системы».

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Брюс Эккель. Философия Java [Текст] — Санкт-Петербург: Классика Computer Science, 2016 г. — 1168 с.
2. Берт Бейтс. Изучаем Java [Текст] Кэти Сиерра — Москва: «Эксмо», 2016. — 720 с.
3. Джошуа Блох. Java. Эффективное программирование [Текст] — Москва: Лори: Java «из первых рук», 2014. — 310 с.
4. Кей С. Хорстманн Java. Библиотека профессионала. Том 1. Основы [Текст] — Москва: Вильямс, 2017. — 864 с.
5. Иванова Г.С. Объектно-ориентированное программирование [Текст]: учебник / Г. С. Иванова, Т. Н. Ничушкина. — Москва: Изд-во МГТУ,2014. — 456 с.: ил.
6. Финогенов К.Г Основы объектно-ориентированного программирования. Лабораторный практикум [Текст]: Учебное пособие. Москва: МИФИ, 2008. — 92с.
7. Колесов Ю. Б. Моделирование систем. Объектно-ориентированный подход [Текст]: Учебное пособие/ Ю. Б. Колесов, Ю. Б. Сениченков. — СПб.: БХВ-Петербург, 2012. — 192 с.: ил.
8. Грэхем Иан. Объектно-ориентированные методы. Принципы и практика [Текст]: - 3-е изд. — Москва: «Вильямс», 2004.
9. Нотон П. JAVA: Справ.руководство [Текст]: Пер.с англ./Под ред.А.Тихонова.Москва:БИНОМ:Восточ.Кн.Компания,1996:Восточ.Кн.Компания. – 447с. – (Club Computer).
10. Чен М.С. и др. Программирование на JAVA:1001 совет [Текст]: Наиболее полное руководство по Java и Visual J++: Пер.с англ./ Чен М.С., Грифис С.В.,Изи Э.Ф.— Минск:Попурри,1997.-640с.ил.

11. Файлы [Электронный ресурс]. — Режим доступа: http://school-collection.lyceum62.ru/ecor/storage/9ca37130-a040-4d34-9695-394b0c64b664/[INF10_01_09].html#4 (дата обращения: 12.04.2017).

12. Файловая система [Электронный ресурс]. — Режим доступа: http://coolreferat.com/Файловая_система_для_операционной_системы_Windows (дата обращения: 14.04.2017).

13. Файловые менеджеры для windows [Электронный ресурс]. — Режим доступа: http://softcatalog.info/ru/obzor/faylovye-menedzhery-dlya-windows (дата обращения: 10.04.2017).

14. Компиляция [Электронный ресурс]. — Режим па: http://learn.javascript.ru/intro (дата обращения: 11.04.2017).

15. Языки программирования [Электронный ресурс]. — Режим доступа: http://progopedia.ru/language/ (дата обращения: 16.04.2017).

16. JavaScript [Электронный ресурс]. — Режим па: http://ru.wikipedia.org/wiki/JavaScript (дата обращения: 19.04.2017).

17. IntelliJ IDEA [Электронный ресурс]. — Режим па: http://jetbrains.ru/products/idea/ (дата обращения: 8.04.2017).

18. Сумина Т. Г. Общая и профессиональная педагогика. [Текст]: учеб. пособие/ Т. Г. Сумина. — Екатеринбург: Изд-во ГОУ ВПО «Рос. гос. проф.-пед.ун-т», 2008. — 127 с.

19. Эрганова Н. Е. Методика профессионального обучения [Текст]: учеб. пособие / Н. Е. Эрганова. — М.: Издательский центр «Академия», 2008. — 160 с.

20. Эрганова Н. Е. Практикум по методике профессионального обучения [Текст]: учеб. пособие / Н. Е. Эрганова. — Екатеринбург: Изд-во Рос. гос. проф.-пед.ун-та, 2011. — 89 с.

21. Эрганова Н. Е. Практикум по педагогическим технологиям [Текст]: учеб. пособие / Н. Е. Эрганова. — Екатеринбург: Изд-во Рос. гос. проф.-пед. ун-та, 2011. — 50 с.

22. Командная строка [Электронный ресурс]. — Режим доступа: http://www.colorpilot.ru/command_line.html (дата обращения: 12.04.2017).

23. Windows PowerShell [Электронный ресурс]. — Режим па: http://technet.microsoft.com/ru-ru/library/bb978526.aspx (дата обращения: 06.04.2017).

24. Джейсон Мейнджер Java: Основы программирования. — 1996, Издательская группа BHV, Киев,1997. — 150 с.

25. Файловые менеджеры [Электронный ресурс]. — Режим па: http://softcatalog.info/ru/obzor/faylovye-menedzhery-dlya-windows (дата обращения: 09.04.2017).

26. Проводник [Электронный ресурс]. — Режим па: http://computer-lectures.ru/operacionnye-sistemy-i-obolochki/6-6-programma-provodnik/ (дата обращения: 06.04.2017).

ПРИЛОЖЕНИЕ

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»

Институт *Инженерно-педагогического образования*
Кафедра *Информационных систем и технологий*
Направление подготовки *44.03.04 Профессиональное обучение*
Профилизация *Компьютерные технологии автоматизации и управления*

Заведующий кафедрой ИС

_____ *Н.С. Толстова*
(подпись) (Фамилия И.О.)
« _____ » _____ 20 17 г.

ЗАДАНИЕ
на выполнение **выпускной квалификационной работы** бакалавриата
(дипломная работа)

студента (ки) **4** курса группы **КТэ-402**

Чернышова Андрея Николаевича
(фамилия, имя, отчество полностью)

1. Тема **Лабораторный практикум «Организация работы с файлами на основе Java»**

утверждена распоряжением по институту от « ___ » _____ 20 17 г. № ___

2. Руководитель Телепова Татьяна Петровна
(фамилия, имя, отчество полностью)

_____ _____ старший преподаватель каф. ИС
(ученая степень) (ученое звание) (должность) (место работы)

3. Место преддипломной практики «РГППУ»

4. Исходные данные к ВКР Берт Бейтс. Изучаем Java Кэти Сиерра — Москва: «Эксмо», 2016. — 720 с.
(список основной литературы)

5. Содержание пояснительной записки ВКР (перечень подлежащих разработке вопросов)

1)Теоретический раздел.
2) Практический раздел.
3) Методический раздел.
4) Список используемых источников информации

6. Перечень графических и демонстрационных материалов _____

7. Календарный план выполнения выпускной квалификационной работы

№ п/п	Наименование этапа дипломной работы	Срок выполнения этапа	Процент выполнения ВКР	Отметка руководителя о выполнении
1	*Поиск информации по теме ВКР* *Работа над теоретическим разделом ВКР* *Сдача зачета по преддипломной практике*	*02.04.2017–* *04.05.2017*	*40 %*	(подпись)
2	Выполнение работ по разрабатываемым вопросам, их изложение в пояснительной записке ВКР:			(подпись)
	Выполнение и оформление теоретического раздела ВКР	*05.05.2017*	*45 %*	(подпись)
	Работа над практическим разделом ВКР			(подпись)
	Выполнение и оформление практического раздела ВКР	*19.05.2017*	*75 %*	(подпись)
	Работа над заданием методического раздела			(подпись)
	Выполнение и оформление методического раздела	*02.06.2017*	*85 %*	(подпись)
3	*Оформление демонстрационных материалов: электронная презентация (плакаты) и подготовка доклада к предварительной защите*	*17.06.2017*	*90 %*	(подпись)
4	*Подготовка доклада к предварительной защите*	*18. 06.2017*		(подпись)
5	*Нормоконтроль*	*12– 19. 06.2017*	*95%*	(подпись)
6	*Предварительная защита*	*21– 26. 06.2017*	*98 %*	(подпись)
7	*Подготовка к защите*	*24.06.2017- 28. 06.2017*		(подпись)
8	*Защита ВКР*	*30.06.2017*	*100 %*	

8. Консультанты по разделам выпускной квалификационной работы

Наименование раздела	Консультант	Задание выдал		Задание принял	
Методическая часть		(подпись)	(дата)	(подпись)	(дата)
Нормоконтроль		(подпись)	(дата)	(подпись)	(дата)
Предварительная защита		(подпись)	(дата)	(подпись)	(дата)

Руководитель _____ _____ *Задание получил* _____ _____
(подпись) (дата) (подпись) (дата)

9. Пояснительная записка дипломной работы и все материалы проанализированы
Считаю возможным допустить **Чернышова Андрея Николаевича** к защите выпускной квалификационной работы в государственной экзаменационной комиссии

Руководитель _____ _____
(подпись) (дата)

10. Допустить **Чернышова А.Н.** к защите выпускной квалификационной работы
(фамилия и.о. студента)
в государственной экзаменационной комиссии (протокол заседания кафедры
от «_____» 2017 г., № _____)

Заведующий кафедрой _____ _____
(подпись) (дата)