Министерство образования и науки РФ

ФГБОУ ВО «Омский государственный технический университет»

Кафедра «Автоматизированные системы обработки информации и управления»

**ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ**

по дисциплине «Объектно-ориентированное программирование»

студентки Шоколовой Анастасии Александровны группы МО-171

Проверил,

старший преподаватель

\_\_\_\_\_\_\_\_\_\_\_Д.Н. Шлома

«\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2019 г.

Выполнила студентка

\_\_\_\_\_\_\_\_\_А.А.Шоколова

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Омск 2019

АННОТАЦИЯ

Для решения задач, поставленных в данных лабораторных работах, использовался язык программирования Java, версии 1.8.

Набор исходного кода осуществлялся в интегрированной среде разработки Intellij IDEA Community Edition 2018.

Запуск откомпилированной программы, требует наличия на целевой ПЭВМ среды выполнения для Java, т.е. JavaRuntimeEnvironment(JRE).

В целях упрощения процедуры запуска, в директории c программой, соответствующей каждой лабораторной работе, находится скриптовый файл – для ОС, основанных на ядре Linux это\_start.sh, а для ОС семейства Microsoft Windows – \_start.bat. Запуск скриптового файла инициирует запуск программы.

Исходный код доступен по ссылке https://github.com/crysatros?tab=repositories

|  |  |
| --- | --- |
| СОДЕРЖАНИЕ  1 Лабораторная работа №15  1.1 Описание структуры исходного кода программы5  1.2 Скриншоты запущенной программы5  1.3 Вывод6  2Лабораторная работа №27  2.1 Описание структуры исходного кода программы7  2.2 Скриншоты запущенной программы7  2.3 Вывод8  3 Лабораторная работа №39  3.1 Описание структуры исходного кода программы9  3.2 Скриншоты запущенной программы9  3.3 Вывод11  4 Лабораторная работа №412  4.1 Описание структуры исходного кода программы12  4.2 Скриншоты запущенной программы12  4.3 Вывод13  5 Лабораторная работа №514  5.1 Описание структуры исходного кода программы14  5.2 Скриншоты запущенной программы14  5.3 Вывод15  6 Лабораторная работа №616  6.1 Описание структуры исходного кода программы16  6.2 Скриншоты запущенной программы16  6.3 Вывод17  7 Лабораторная работа №718  7.1 Описание структуры исходного кода программы18  7.2 Скриншоты запущенной программы18  7.3 Вывод19  8Лабораторная работа №7.120  8.1 Описание структуры исходного кода программы20  8.2 Скриншоты запущенной программы20  8.3 Вывод21  Список использованных источников22  Приложение А1. Лабораторная работа №1. Исходный код23  Приложение А2. Лабораторная работа №2. Исходный код24  Приложение А3. Лабораторная работа №3. Исходный код26  Приложение А4. Лабораторная работа №4. Исходный код28  Приложение А5. Лабораторная работа №5. Исходный код30  Приложение А6. Лабораторная работа №6. Исходный код31  Приложение А7. Лабораторная работа №7. Исходный код33  Приложение А7.1. Лабораторная работа №7.1. Исходный код35 |  |

1. **Лабораторная работа №1**

Условие задачи: ввести n строк с консоли без использования Scanner. Удалить лишние пробелы.

* 1. **Описание структуры программы**

В общем виде структура исходного кода программы для каждой лабораторной работы, представляет собой совокупность классов, основным из которых является класс Main.

Класс Main, содержит точку входа в Java программу в виде статического метода main, осуществляет запросы к пользователю, организует взаимодействие между классами, содержащими алгоритмы, и выводит результат работы программы.

Метод System.out.println позволяет выводить текст в консоль.

Класс Test в данной работе заменяет класс Main и является основным. Он используется для получения (считывания) данных введенных пользователем в виде char.

Метод System.in.read() в классе позволяет читать до конца текущей строки, а метод toCharArray() позволяет создать массив символов, и метод str.length() позволяет узнать размер получившегося массива.

* 1. **Скриншоты запущенной программы**

После запуска программы, на консоль выводится запрос о вводе предложения. Общий вид окна запущенной программы, приведен на рисунке 1.1.

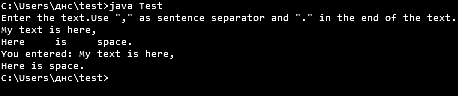


Рисунок 1.1 – Результат работы программы

По окончании работы программы на консоль выводится введённый текст, но уже без лишних пробелов.

* 1. **Вывод**

В результате выполнения лабораторной работы разработана программа выполняющая ввод n строк с консоли и выводящую введённый пользователем текст без лишних пробелов. Исходный код программы приведен в приложении А1.

1. **Лабораторная работа №2**

Условие задачи: Создать генератор паролей, который будет генерировать пароли различной длины в виде строк из случайных символов.

* 1. **Описание структуры программы**

В данной работе используются 2 класса: Main и PassGenerator.

Класс PassGenerator содержит в себе генератор паролей с помощью метода getPass, позволяющим генерировать с помощью цикла строку определенной длины.

Метод rand в классе PassGenerator позволяет использовать цикл для генерации строк различных последовательностей из случайных английских букв.

Метод NextInt позволяет считывать число из класса Main, чтобы разбить строки на строки длиной, равной данному числу, и также использовать 26 букв английского алфавита для генерации последовательности символов.

Класс Main использует класс PassGenerator, и при помощи метода getPass получает случайно сгенерированные строки определенной длины для создания пароля.

Метод System.out.println позволяет выводить текст в консоль

* 1. **Скриншоты запущенной программы**

После запуска программы, на консоль выводятся пароли из случайных символов длиной 10, 3 и 1 символ. Общий результат работы программы, приведен на рисунке 2.1.

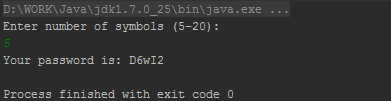


Рисунок 2.1 – Результат работы программы

Для получения других паролей данной длины программу можно перезапустить и получить новые комбинации паролей из символов.

* 1. **Вывод**

В результате выполнения лабораторной работы разработана программа выполняющая генерацию паролей различной длины в виде строк из случайных символов. Исходный код программы приведен в приложении А2.

1. **Лабораторная работа №3**

Условие задачи: Реализовать работу с текстовыми файлами, по указанному адресу внутри программ найти и открыть текстовый файл, скопировать текст из файла в буфер обмена, создать копию данного файла и вставить туда скопированный текст из первого файла, вывести скорость и время копирования файла.

* 1. **Описание структуры программы**

В данной работе используются 2 класса: Main и FileManager.

Класс FileManager содержит в себе все методы, необходимые для работы с текстовыми файлами.

Метод FileInputStream означает входящий поток чтения, а FileOutputStream - выходящий поток записи, size4MB используется для копирования данных из файлов, и его размер задан в 4 Мб, stat хранит данные о времени копирования и скорости копирования в виде одномерного массива.

С помощью write мы записываем в буфер обмена копируемую информацию в копию файла. Благодаря currentTimeMillis мы можем рассчитать время копирования файла и закрыть оба файла с помощью close по завершению их работы. Метод System.out.println позволяет выводить текст в консоль.

Класс Main использует класс FileManager, и при помощи метода copy делает копию файла, указанного по адресу.

* 1. **Скриншоты запущенной программы**

После запуска программы, программа ищет файл testo.txt в директории D:\\testo.txt. Если он обнаружен, программа в той же директории создает его копию testc.txt и выводит в консоль время и скорость копирования. Общий результат работы программы, приведен на рисунках 3.1-3.3.

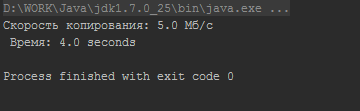


Рисунок 3.3 – Вывод в консоль

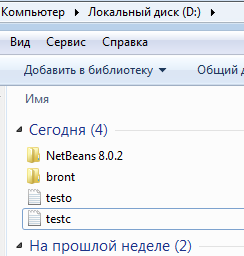
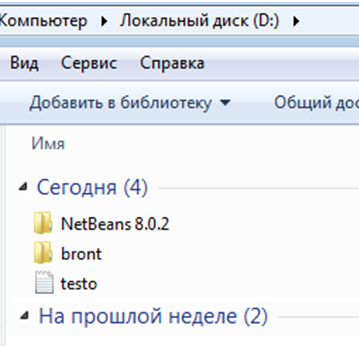


Рисунок 3.2 – Результат работы программы

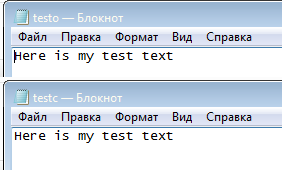


Рисунок 3.3 – Результат работы программы

По окончании работы программы мы видим копию текстового файла со всем его скопированным текстом, а также выведенные в консоль скорость и время копирования этого файла.

* 1. **Вывод**

В результате выполнения лабораторной работы разработана программа для работы с файлами, находящая текстовый файл в данной директории и создающей его копию. Исходный код программы приведен в приложении А3.

1. **Лабораторная работа №4**

Условие задачи: В текстовом файле при помощи программы заменить выбранную строку новой строкой и выбранное слово в строке на новое слово.

* 1. **Описание структуры программы**

В данной работе используются 2 класса: Main и TextEditor.

Класс TextEditor содержит в себе все методы, необходимые для работы с текстовым файлом.

Метод позволяет получить разделитель пути в текущей операционной системе.

При помощи filePath мы находим нужный нам файл, чтобы его изменить.

BufferedReader необходим для считывания данных из файла.

При помощи метода writeFile мы будем изменять файл так, как нам нужно и закрывать с помощью flush.

При помощи метода changeWord мы находим слово, которое необходимо заменить, и вместо него записываем новое.

При помощи метола changeString мы находим строку, которую нужно заменить, и вместо неё пишем уже новую.

* 1. **Скриншоты запущенной программы**

После запуска программы после открытия текстового файла мы статус программы: либо сообщение о завершении процесса, либо ошибку.

Общий результат работы программы, приведен на рисунке 4.1.

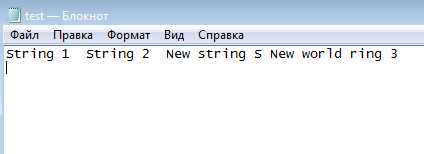


Рисунок 4.1 – Результат работы программы

По окончании работы программы мы видим изменённый текстовый файл с новой строкой и словом.

* 1. **Вывод**

В результате выполнения лабораторной работы разработана программа для редактирования файла строк и слов. Исходный код программы приведен в приложении А4.

1. **Лабораторная работа №5**

Условие задачи: Реализовать запись текста на жесткий диск, USB и принтер. В программе должна быть использованы инкапсуляция, наследование и полиморфизм.

* 1. **Описание структуры программы**

В данной работе используются 5 классов: Main, USB, HDD, Print и Super.

Класс Super в данной работе является супер-классом, от которого наследуют свойства классы USB, HDD и Print, используя оператор extends. Все эти классы-наследники при помощи оператора выводят информацию о том, куда был записан файл.

В классе Super содержится текст файла в строковой переменной str, который он может вернуть при помощи метода getText. Также класс Super реализует сразу несколько методов от дочерних классов, что является реализацией полиморфизма. Класс Main использует дочерние классы. Каждый дочерний класс получает введённую строку и может выводить сообщение о том, куда мы её записали.

Переключатель switch позволяет нам обратиться к блоку инструкций из четырех case, чтобы записать файл на тот или иной носитель, а также выйти из программы.

* 1. **Скриншоты запущенной программы**

После запуска программы, на консоль выводится сообщение «Enter your text:». Мы вводим текст и далее нам предлагают список действий на выбор: мы можем записать текст на любой носитель или выйти из программы.

Общий результат работы программы, приведен на рисунке 5.1.

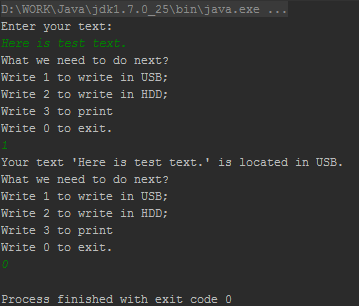


Рисунок 5.1 – Результат работы программы

По окончании работы программы мы видим уведомление об успешном открытии файла произвольного формата.

* 1. **Вывод**

В результате выполнения лабораторной работы разработана программа просмотра для записи текста на жесткий диск, USB и принтер, а также были использованы инкапсуляция, наследование и полиморфизм. Исходный код программы приведен в приложении А5.

1. **Лабораторная работа №6**

Условие задачи: Создать многопоточное приложение с независимым выполнением двух потоков, и один из них должен завершиться раньше другого.

* 1. **Описание структуры программы**

В данной работе используются 4 класса: Main, Log, Thread1, Thread2.

Класс Main в данной работе использует классы Thread1 и Thread2, наследующие свойства от Thread.

Метод System.out.println позволяет выводить текст в консоль.

Метод toConsole класса Log синхронизирует потоки Thread1 и Thread2 с текущими датой и временем и позволяет выводить информацию о времени, когда они начали работу.

Метод run запускает потоки на исполнение.

* 1. **Скриншоты запущенной программы**

После запуска программы, на консоль выводится сообщение «Сравним, какой же из двух потоков завершит работу раньше?». Поток 1 и 2 выполняются до тех пор, пока какой либо поток из них не завершит работу раньше.

Общий результат работы программы, приведен на рисунке 6.1.

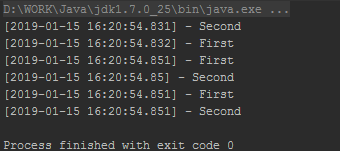


Рисунок 6.1 – Результат работы программы

По окончании работы программы мы видим список, который показывает, в какое время каждый поток выводил своё имя.

* 1. **Вывод**

В результате выполнения лабораторной работы разработана многопоточная программа с независимым выполнением двух потоков, один из которых завершает свою работу раньше другого. Исходный код программы приведен в приложении А6.

1. **Лабораторная работа №7**

Условие задачи: Создать переводчик из одной величины в другую.

* 1. **Описание структуры программы**

В данной работе используются 2 класса: JavafxConverter и FXMLDocumentController.

Класс JavafxConverter в данной работе использует класс FXMLDocumentController и разметку интерфейса fxml .

Класс FXMLDocumentController является основным классом в данной программе на JavaFX, где записан основной выполняющийся код программы. Использует методы, аналогичные классу Main.

TextField – текстовые поля, использующиеся для ввода и вывода значений.

Метод lengthConverAction обрабатывает данные в полях ichesField и centimetersField.

Метод massConverAction обрабатывает данные в полях poundsField и kilogramsField.

Оба метода обрабатывают данные по нажатию кнопки Convert.

Расположение объектов записано в fxml-файл.

* 1. **Скриншоты запущенной программы**

После запуска программы, мы видим форму с 2 полями для ввода, рядом с ними поясняющий текст и кнопка Convert, также есть 2 фкладки для перехода между полем длинны к полю массы и наоборот. Если ввести значение в верхнее поле ввода, например в Inches – то после нажатия на кнопку Convert результат будет выведен в нижнее поле Centimeters, показывая то же значение в сантиметрах и наоборот. Аналогично программа работает и с массой тела, переводя из фунтов в килограммы и наоборот.

Общий результат работы программы, приведен на рисунке 7.1.

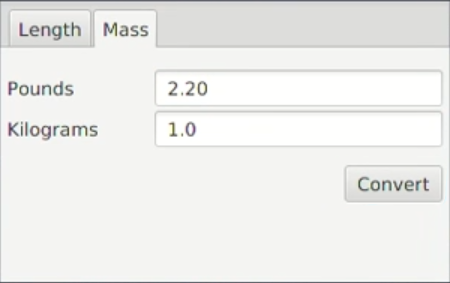
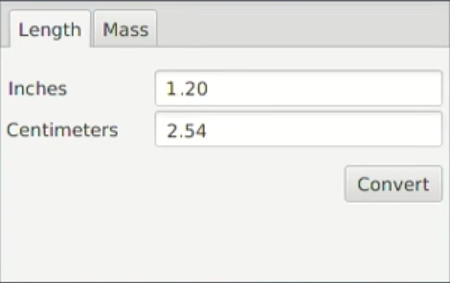


Рисунок 7.1 – Результат работы программы

* 1. **Вывод**

В результате выполнения лабораторной работы в JavaFX разработана программа, при нажатии кнопки «Convert» конвертирует дюймы в сантиметры, а фунты в килограммы. Исходный код программы приведен в приложении А7.

1. **Лабораторная работа №7.1**

Условие задачи: Написать переводчик из одной величины в другую с использованием Combo box.

* 1. **Описание структуры программы**

В данной работе используется 1класс SwingConverter, наследующий JApplet.

JtextField используется для того, чтобы получить введённое значение.

JComboBox используется для выбора из какой величины в какую перевести.

Jlabel используется для пояснения полей и вывода значения после конвертации.

При помощи метода actionPerformed мы получаем информацию с поля ввода при помощи getText, обрабатываем и сравниваем данные с обоих ComboBox и умножаем значение переменной на нужную константу, затем ответ выводится в отдельный Label ниже с помощью setText.

При помощи метода GUI мы задаём активность, её параметры.

С помощью GridBagLayout и GridBagConstraints мы создаём гибкую сетку для компонентов, где мы описываем расположения объектов в активности при помощи этой сетки.

С помощью add мы добавляем уже настроенный элемент.

При помощи методов init и run мы запускаем метод GUI и саму программу.

* 1. **Скриншоты запущенной программы**

После запуска программы, мы видим форму с 1 полем для ввода значения, 2 ComboBox, предназначенных для выбора интересующих величин, а также пояснения к полям в виде нескольких объектов Label. После ввода значения и выбора величин программа выводит в нижний Label результат конвертации.

Общий результат работы программы, приведен на рисунке 8.1.

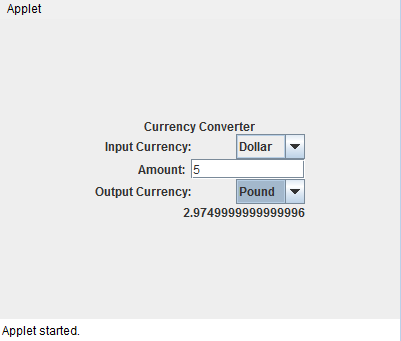


Рисунок 8.1 – Результат работы программы

* 1. **Вывод**

В результате выполнения лабораторной работы программа, которая конвертирует значение из одной величины в другую при помощи ComboBox и TextField, выводящая значения в отдельный Label. Исходный код программы приведен в приложении А7.1.

**Список использованных источников**

1. Шилдт Г.Java 8. Полное руководство, 9-е изд. Пер. с англ. – М.: ООО «И.Д. Вильямс», 2015. – 1376 с.
2. Давыдов С. IntelliJ IDEA. Профессиональное программирование на Java / Станислав Давыдов, Алексей Ефимов. – М.: БХВ-Петербург, 2015. – 800 c.;
3. Руководство по языку программирования Java: [Сайт] https://metanit.com/java/tutorial/ дата обращения: (03.09.2018).

**Приложение А1**

**Лабораторная работа №1. Исходный код**

import java.io.IOException;

public class Test {

public static void main(String[] args) throws IOException {

String str ="";

char inChar, buf;

int i, j, k;

System.out.println("Enter the text. Use \",\" as sentence separator and \".\" in the end of the text.");

do {

inChar = (char) System.in.read();

str += inChar;

} while(inChar != '.');

char[] MyStr = str.toCharArray();

int strlen = str.length();

System.out.print("You entered: ");

for(k = 0; k < strlen-1; k++) for( i= 0; i< strlen-1; i++){

if(MyStr[i] == ' '){

if(MyStr[i] == MyStr[i+1])

for(j = i; j < strlen-1; j++) {

buf = MyStr[j];

MyStr[j] = MyStr[j + 1];

MyStr[j + 1] = buf;

}}

}

for (i = 0; i < strlen; i++) {

System.out.print(MyStr[i]);

}

}

}

**Приложение А2**

**Лабораторная работа №2. Исходный код**

PassGenerator.java

import java.util.Random;

public class PassGenerator {

private int size;

Random random = new Random();

public String getPass(int size){

this.size = size;

String pass = "";

for (int i = 0; i < size; i++){

int buf = random.nextInt(4);

switch(buf){

case 0: pass += (char)rand('0', '9'); break;

case 1: pass += (char)rand('a', 'z'); break;

case 2: pass += (char)rand('A', 'Z'); break;

case 3: pass += '\_'; break;

}

}

return pass;

}

private int rand(int nach, int kon){

int val = nach + random.nextInt(kon-nach+1);

return val;

}

}

Main.java

import java.util.Scanner;

public class Main {

public static void main(String[] args){

int size = 0;

System.out.println("Enter number of symbols (5-20):");

Scanner sc = new Scanner(System.in);

size = sc.nextInt();

if(size < 5 || size > 20 ) System.out.println("Incorrect number.");

else {

PassGenerator pasGen = new PassGenerator();

String genPass = pasGen.getPass(size);

System.out.format("Your password is: ").println(genPass);

}

}

}

**Приложение А3**

**Лабораторная работа №3. Исходный код**

FileManager.java

import java.io.\*;

public class FileManager {

final int size4Mb = 4\*1024\*1024;

public double[] copy(String orig, String copy) throws IOException{

double[] stat = new double[2];

try{

FileInputStream is = new FileInputStream(orig);

FileOutputStream os = new FileOutputStream(copy);

byte[] buf = new byte[size4Mb];

int len;

long CopyTime = System.currentTimeMillis();

long totalSize = 0;

while((len = is.read(buf))>0){

os.write(buf);

totalSize += len;

}

long endCopy = System.currentTimeMillis();

long copyTime = endCopy - CopyTime;

try {

stat[1] = ((double) copyTime);

stat[0] = totalSize / copyTime;

}

catch(ArithmeticException e){

stat[0] = 0;

}

finally{

is.close();

os.close();

}

}catch (FileNotFoundException e){

}

catch (IOException e){

}

return stat;

}

}

Main.java

public class Main {

public static void main(String[] args) {

String origFile = "D:\\testo.txt";

String copyFile = "D:\\testc.txt";

try {

FileManager fileManager = new FileManager();

double[] copyStat = new double[2];

copyStat = fileManager.copy(origFile, copyFile);

System.out.println("Скорость копирования: " + copyStat[0] + " Мб/с\n Время: " + copyStat[1] + " seconds");

}catch(Exception e){

System.out.println(e.getMessage());

}

}

}

**Приложение А4**

**Лабораторная работа №4. Исходный код**

Main.java

public class Main {

public static void main(String[] args) {

String filePath = "C:\\Users\\днс\\test.txt";

TextEditor txt = new TextEditor();

txt.changeWord(filePath,3,2,"New world");

txt.changeString(filePath,2,"New string");

}

}

TextEditor.java

import java.io.\*;

import java.util.\*;

public class TextEditor extends IOException{

Vector<String> file = new Vector<String>();

boolean readFile(String filePath){

InputStream in = null;

try {

in = new FileInputStream(filePath);

} catch(FileNotFoundException e){}

BufferedReader bufread = new BufferedReader(new InputStreamReader(in));

String str;

try {

while ((str = bufread.readLine()) != null){

file.add(str);

}

in.close();

} catch (IOException e){}

return true;

}

boolean writeFile(String filePath){

OutputStream out;

try{

out = new FileOutputStream(filePath,false);

} catch(FileNotFoundException e){

return false;

}

BufferedWriter bufwrit = new BufferedWriter(new OutputStreamWriter(out));

try {

for (String s:file) {

bufwrit.write(s+" \n");

}

bufwrit.flush();

}catch (IOException e){

return false;

}

file.clear();

return true;

}

boolean changeWord(String filePath,int Snum, int Wnum, String newW){

readFile(filePath);

String[] word = file.elementAt(Snum).split("");

word[Wnum] = " " + newW + " ";

StringBuilder builder = new StringBuilder();

for(String wb:word){

builder.append(wb);

}

file.set(Snum,builder.toString());

writeFile(filePath);

return true;

}

boolean changeString(String filePath,int num, String newS){

readFile(filePath);

file.set(num,newS);

writeFile(filePath);

return true; }}

**Приложение А5**

**Лабораторная работа №5. Исходный код**

Super.java

public class Super {  
 private String str = new String();  
 public Super(String str){  
 this.str = str; }  
 public String getText(){  
 return str;  
 }}

Main.java

import java.util.Scanner;

public class Main {

public static void main(String args[]){

String text = new String();

int num;

Scanner sc = new Scanner(System.in);

System.out.println("Enter your text:");

text = sc.nextLine();

for(;;){

System.out.println("What we need to do next?\nWrite 1 to write in USB;\nWrite 2 to write in HDD;\nWrite 3 to print\nWrite 0 to exit.");

num = sc.nextInt();

USB usb = new USB(text);

HDD hdd = new HDD(text);

Print print = new Print(text);

switch(num){

case 1: usb.writeUSB(); break;

case 2: hdd.writeHDD(); break;

case 3: print.printerPrint(); break;

case 0: System.exit(0);

default: System.out.println("Try again."); break;

}}

}

}

**Приложение А6**

**Лабораторная работа №6. Исходный код**

Main.java

public class Main {

public static void main(String[] args) {

Thread1 t1 = new Thread1("First");

Thread2 t2 = new Thread2("Second");

t1.start();

t2.start();

}

}

Thread1.java

public class Thread1 extends Thread {

private String name;

public Thread1(String name){

this.name = name;

}

public void run() {

for(int i = 0;i < 3;i++){

Log.toConsole(this.name);

}

}

}

Thread2.java

public class Thread2 extends Thread {

private String name;

public Thread2(String name){

this.name = name;

}

public void run() {

for(int i = 0;i < 3;i++){

Log.toConsole(this.name);

}

}

}

Log.java

import java.sql.Timestamp;

public class Log {

static public void toConsole(String msg){

System.out.format("[%s] - %s \n", new Timestamp(System.currentTimeMillis()), msg);

}

}

**Приложение А7**

**Лабораторная работа №7. Исходный код**

JavafxConverter.class

package converter;  
  
import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

public class JavafxConverter extends Application{

@Override

public void start(Stage stage) throws Exception{

Parent root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));

Scene scene = new Scene(root);

stage.setScene(scene);

stage.setTitle("Converter");

stage.show();

}

}

Controller.class

package converter;

import java.net.URL;

import java.util.ResourceBundle;

import javafx.fxml.Initializable;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

public class FXMLDocumentController implements Initializable {

@FXML

private TextField ichesField;

@FXML

private TextField centimetersField;

@FXML

private TextField poundsField;

@FXML

private TextField kilogramsField;

boolean operationLength = true;

@Override

public void initialize(URL url, ResourceBundle rb) {

}

@FXML

private void lengthConverAction (ActionEvent event){

if(!ichesField.getText().equals("")){

double iches = Double.parseDouble(ichesField.getText());

double centimeters = iches \* 2.54;

centimetersField.setText("" + centimeters);

} else{

double centimeters = Double.parseDouble(centimetersField.getText());

double iches = centimeters / 2.54;

ichesField.setText("" + iches);

}

}

@FXML

private void massConverAction (ActionEvent event){

if (!poundsField.getText().equals("")){

double pounds = Double.parseDouble(poundsField.getText());

double kilograms = pounds / 2.2;

kilogramsField.setText("" + kilograms);

} else{

double kilograms = Double.parseDouble(kilogramsField.getText());

double pounds = kilograms \* 2.2;

poundsField.setText("" + pounds);

}

}

}

**Приложение А7.1**

**Лабораторная работа №7.1. Исходный код**

import java.awt.\*;  
import java.awt.event.\*;  
import javax.swing.\*;  
public class SwingConverter extends JApplet implements ActionListener{  
 JTextField t1;  
 JComboBox c1,c2;  
 JLabel name, l1, l2, l3, l4;  
 String[] currency={"Dollar","Rupee","Pound","Euro","Driham","Yaun"};  
 public void init(){  
 try{SwingUtilities.invokeAndWait(new Runnable(){  
 public void run(){  
 GUI();  
 }  
 });  
 }catch(Exception e){}  
 }  
 public void actionPerformed(ActionEvent e2) {  
 double zn, mnoz = 0;  
 zn = Double.valueOf(t1.getText());  
 try {  
 if (c1.getSelectedIndex() == 0 & c2.getSelectedIndex() == 1) mnoz = zn \* 60.335;  
 if (c1.getSelectedIndex() == 0 & c2.getSelectedIndex() == 2) mnoz = zn \* 0.595;  
 if (c1.getSelectedIndex() == 0 & c2.getSelectedIndex() == 3) mnoz = zn \* 0.723;  
 if (c1.getSelectedIndex() == 0 & c2.getSelectedIndex() == 4) mnoz = zn \* 3.673;  
 if (c1.getSelectedIndex() == 0 & c2.getSelectedIndex() == 5) mnoz = zn \* 6.221;  
 if (c1.getSelectedIndex() == 1 & c2.getSelectedIndex() == 0) mnoz = zn \* 0.017;  
 if (c1.getSelectedIndex() == 1 & c2.getSelectedIndex() == 2) mnoz = zn \* 0.01;  
 if (c1.getSelectedIndex() == 1 & c2.getSelectedIndex() == 3) mnoz = zn \* 0.012;  
 if (c1.getSelectedIndex() == 1 & c2.getSelectedIndex() == 4) mnoz = zn \* 0.06;  
 if (c1.getSelectedIndex() == 1 & c2.getSelectedIndex() == 5) mnoz = zn \* 0.103;  
 if (c1.getSelectedIndex() == 2 & c2.getSelectedIndex() == 0) mnoz = zn \* 1.679;  
 if (c1.getSelectedIndex() == 2 & c2.getSelectedIndex() == 1) mnoz = zn \* 101.251;  
 if (c1.getSelectedIndex() == 2 & c2.getSelectedIndex() == 3) mnoz = zn \* 1.215;  
 if (c1.getSelectedIndex() == 2 & c2.getSelectedIndex() == 4) mnoz = zn \* 6.17;  
 if (c1.getSelectedIndex() == 2 & c2.getSelectedIndex() == 5) mnoz = zn \* 10.45;  
 if (c1.getSelectedIndex() == 3 & c2.getSelectedIndex() == 0) mnoz = zn \* 1.382;  
 if (c1.getSelectedIndex() == 3 & c2.getSelectedIndex() == 1) mnoz = zn \* 83.333;  
 if (c1.getSelectedIndex() == 3 & c2.getSelectedIndex() == 2) mnoz = zn \* 0.823;  
 if (c1.getSelectedIndex() == 3 & c2.getSelectedIndex() == 4) mnoz = zn \* 5.079;  
 if (c1.getSelectedIndex() == 3 & c2.getSelectedIndex() == 5) mnoz = zn \* 8.601;  
 if (c1.getSelectedIndex() == 4 & c2.getSelectedIndex() == 0) mnoz = zn \* 0.909;  
 if (c1.getSelectedIndex() == 4 & c2.getSelectedIndex() == 1) mnoz = zn \* 54.795;  
 if (c1.getSelectedIndex() == 4 & c2.getSelectedIndex() == 2) mnoz = zn \* 0.541034;  
 if (c1.getSelectedIndex() == 4 & c2.getSelectedIndex() == 3) mnoz = zn \* 0.658;  
 if (c1.getSelectedIndex() == 4 & c2.getSelectedIndex() == 4) mnoz = zn \* 3.339;  
 if (c1.getSelectedIndex() == 4 & c2.getSelectedIndex() == 5) mnoz = zn \* 5.655;  
 if (c1.getSelectedIndex() == 5 & c2.getSelectedIndex() == 0) mnoz = zn \* 0.272;  
 if (c1.getSelectedIndex() == 5 & c2.getSelectedIndex() == 1) mnoz = zn \* 16.409;  
 if (c1.getSelectedIndex() == 5 & c2.getSelectedIndex() == 2) mnoz = zn \* 0.162;  
 if (c1.getSelectedIndex() == 5 & c2.getSelectedIndex() == 3) mnoz = zn \* 0.197;  
 if (c1.getSelectedIndex() == 5 & c2.getSelectedIndex() == 4) mnoz = zn \* 0.299;  
 if (c1.getSelectedIndex() == 5 & c2.getSelectedIndex() == 6) mnoz = zn \* 1.695;  
 if (c1.getSelectedIndex() == 6 & c2.getSelectedIndex() == 0) mnoz = zn \* 0.161;  
 if (c1.getSelectedIndex() == 6 & c2.getSelectedIndex() == 1) mnoz = zn \* 9.689;  
 if (c1.getSelectedIndex() == 6 & c2.getSelectedIndex() == 2) mnoz = zn \* 0.096;  
 if (c1.getSelectedIndex() == 6 & c2.getSelectedIndex() == 3) mnoz = zn \* 0.116;  
 if (c1.getSelectedIndex() == 6 & c2.getSelectedIndex() == 4) mnoz = zn \* 0.177;  
 if (c1.getSelectedIndex() == 6 & c2.getSelectedIndex() == 5) mnoz = zn \* 0.59;  
 l4.setText(String.valueOf(mnoz));  
 } catch (Exception e3){}  
 }  
 private void GUI(){  
 GridBagLayout bagLayout = new GridBagLayout();  
 GridBagConstraints constraints = new GridBagConstraints();  
 setLayout(bagLayout);  
 c1=new JComboBox(currency);  
 c1.setSelectedIndex(0);  
 c2=new JComboBox(currency);  
 c2.setSelectedIndex(1);  
 t1=new JTextField(10);  
 name = new JLabel("Currency Converter");  
 l1 = new JLabel("Input Currency:");  
 l2 = new JLabel("Amount: ");  
 l3 = new JLabel("Output Currency:");  
 l4 = new JLabel(" ");  
 constraints.gridwidth = GridBagConstraints.REMAINDER;  
 constraints.anchor = GridBagConstraints.NORTH;  
 bagLayout.setConstraints(name, constraints);  
 constraints.anchor = GridBagConstraints.EAST;  
 constraints.gridwidth = GridBagConstraints.RELATIVE;  
 bagLayout.setConstraints(l1, constraints);  
 constraints.gridwidth = GridBagConstraints.REMAINDER;  
 bagLayout.setConstraints(c1, constraints);  
 constraints.gridwidth = GridBagConstraints.RELATIVE;  
 bagLayout.setConstraints(l2, constraints);  
 constraints.gridwidth = GridBagConstraints.REMAINDER;  
 bagLayout.setConstraints(t1, constraints);  
 constraints.gridwidth = GridBagConstraints.RELATIVE;  
 bagLayout.setConstraints(l3, constraints);  
 constraints.gridwidth = GridBagConstraints.REMAINDER;  
 bagLayout.setConstraints(c2, constraints);  
 constraints.gridwidth = GridBagConstraints.REMAINDER;  
 bagLayout.setConstraints(l4, constraints);  
 add(name);  
 add(l1);  
 add(c1);  
 add(l2);  
 add(t1);  
 add(l3);  
 add(c2);  
 add(l4);  
 t1.addActionListener(this);  
 c1.addActionListener(this);  
 c2.addActionListener(this);  
 }  
}