Міністерство освіти і науки України

Київський політехнічний інститут ім. Ігоря Сікорського

Теплоенергетичний факультет

Кафедра АПЕПС

ЗВІТ

про виконання лабораторної роботи №3

з дисципліни «Технології конструювання програмного забезпечення»

**«Работа со строками в Java»**

**Виконала:**

Студентка 2 курсу, групи ТІ-01

Круть Катерина Олександрівна

Дата: 13.06.21

**Перевірив:**

доцент, к.ф.-м.н.

Тарнавський Ю.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Варіант №14

Завдання: 3,5,29,39

**Завдання №9:**

Вывести таблицу преобразований целых десятичных чисел в интервале от min до max с шагом step в 16-ном представлении. Параметры задачи вводятся пользователем в ходе диалога с программой за один раз.

**Код програми:**

package task3;  
  
import java.io.IOException;  
import java.util.InputMismatchException;  
import java.util.Scanner;  
  
public class Task\_3 {  
 public static void PrintTable(int min, int max, int step) {  
 for (int j = 1, i = min; i < max; i += step, ++j) {  
 System.*out*.format("%d. %d->%-10s\n", j, i, i);  
 }  
 }  
  
 public static void main(String[] args) throws IOException {  
 Scanner scan = new Scanner(System.*in*);  
 System.*out*.println("Enter min max step:");  
 int min, max, step;  
 try {  
 min = scan.nextInt();  
 max = scan.nextInt();  
 step = scan.nextInt();  
 if(min < 0 | max < 0 | step < 0 | min < max){  
 throw new IOException("Invalid input");  
 }  
 *PrintTable*(min, max, step);  
 } catch (InputMismatchException e) {  
 e.printStackTrace();  
 } catch (RuntimeException e) {  
 e.printStackTrace();  
 }  
 }  
}

**Unity Test:**

package task3;  
  
import org.junit.Before;  
import org.junit.Test;  
  
import java.io.ByteArrayOutputStream;  
import java.io.PrintStream;  
  
import static junit.framework.TestCase.*assertEquals*;  
  
public class Task\_3Test {  
  
 Task\_3 TestObject = new Task\_3();  
 int min, max, step;  
 String actual;  
 private final ByteArrayOutputStream outputStreamCaptor = new ByteArrayOutputStream();  
  
  
 @Before  
 public void setUp() throws Exception {  
 min = 1; max = 11; step = 2;  
 actual = "1. 1->1 \n" +  
 "2. 3->3 \n" +  
 "3. 5->5 \n" +  
 "4. 7->7 \n" +  
 "5. 9->9 \n";  
 System.*setOut*(new PrintStream(outputStreamCaptor));  
 }  
  
 @Test  
 public void printTable() {  
 TestObject.*PrintTable*(min, max, step);  
 *assertEquals*(actual, outputStreamCaptor.toString());  
 }  
}

**Завдання №5:**

Создайте программу для шифрования\расшифровки текста методом Цезаря. В нем ключом является целое число, а шифрование\расшифровка заключается в суммировании\ вычитании кодов символов открытого текста\криптотекста с ключом.

**Код програми:**

package task5;  
import java.util.MissingFormatArgumentException;  
import java.util.MissingResourceException;  
import java.util.Scanner;  
  
public class Task\_5 {  
  
 public static String Ceaser(String text, int shift) {  
 String result = "";  
  
 for (int i = 0; i < text.length(); ++i) {  
 if (Character.*isUpperCase*(text.charAt(i))) {  
 char ch = (char) (((int) text.charAt(i) + shift - 65) % 26 + 65);  
 result += ch;  
 } else {  
 char ch = (char) (((int) text.charAt(i) + shift - 97) % 26 + 97);  
 result+= ch;  
 }  
 }  
 return result;  
 }  
  
 public static void main(String...s){  
 Scanner scan = new Scanner(System.*in*);  
 try {  
 System.*out*.println("Enter a message: ");  
 String message = scan.nextLine();  
 if(message.length() <= 0){  
  
 }  
 System.*out*.println("Enter key: ");  
 int key = scan.nextInt();  
 try {  
 System.*out*.println(*Ceaser*(message, key));  
 System.*out*.println(*Ceaser*(message, 26 + key));  
 } catch (MissingResourceException | MissingFormatArgumentException e) {  
 System.*out*.println(e);  
 }  
 } catch (NumberFormatException e) {  
 System.*out*.println(e);  
 }  
 }  
}

**Unity Test:**

package task5;  
  
import org.junit.Before;  
import org.junit.Test;  
import static junit.framework.TestCase.*assertEquals*;  
  
public class Task\_5Test {  
  
 Task\_5 TestObject = new Task\_5();  
 String strOriginal;  
 Integer key;  
  
 @Before  
 public void setUp() throws Exception {  
 strOriginal = "Message";  
 key = 3;  
 }  
  
 @Test  
 public void ceaser() {  
 String expected = TestObject.*Ceaser*(strOriginal, key);  
 String actual = "Phvvdjh";  
 *assertEquals*(expected, actual);  
 }  
}

**Завдання №**29**:**

Анализ аргументов, задаваемых при запуске программы. Программа определяет, какие из введенных аргументов содержат строку, задаваемую в качестве первого параметра. Шаблон аргумента: строка либо цифр, либо латинских букв, либо букв кириллицы. Программа выводит количество заданных аргументов (без учета первого аргумента) и аргументы, содержащие заданную подстроку  или сообщение о том, что данная строка не содержится во введенных аргументах.

**Код програми:**

package task29;  
  
public class Task\_29 {  
 public static void checking(String args[]){  
 for(int i = 1; i < args.length; ++i) {  
 System.*out*.format("%s %s\n", args[i], args[i].contains(args[0]));  
 }  
 }  
 public static void main(String args[]) {  
 System.*out*.println(args.length - 1);  
 *checking*(args);  
 }  
}

**Unity Test:**

package task29;  
  
import org.junit.Before;  
import org.junit.Test;  
import java.io.ByteArrayOutputStream;  
import java.io.PrintStream;  
import static junit.framework.TestCase.*assertEquals*;  
  
public class Task\_29Test {  
  
 Task\_29 TestObject = new Task\_29();  
 String args[] = {"h", "hi", "ап", "87", "lo", "o", "hello"};  
  
 String actual;  
 private final PrintStream standardOut = System.*out*;  
 private final ByteArrayOutputStream outputStreamCaptor = new ByteArrayOutputStream();  
   
 @Before  
 public void setUp() throws Exception {  
 actual = "hi true\n"+  
 "ап false\n"+  
 "87 false\n"+  
 "lo false\n" +  
 "o false\n" +  
 "hello true\n";  
  
 System.*setOut*(new PrintStream(outputStreamCaptor));  
 }

@Test  
 public void checking() {  
 TestObject.*checking*(args);  
 *assertEquals*(outputStreamCaptor.toString(), actual);  
 }  
}

**Завдання №39:**

Программа определяет тип аргумент – шестнадцатеричное число без знака (шаблон: шестнадцатеричным числом без знака считается строка, которая содержит цифры от 0 до 9 и буквы A(a), B(b), C(c), D(d), E(e),F(f)) или строка. Введенные аргументы-числа преобразуются в двоичные числа. Программа выводит количество заданных аргументов, их значения, а также количество аргументов-чисел и их двоичные значения.

**Код програми:**

package task39;

import java.util.\*;  
  
  
public class Task\_39 {  
 public static List<String> ConvertingToList(String str){  
 List<String> list = new ArrayList<>(Arrays.*asList*(str.split("")));  
 return list;  
 }  
 public static boolean Cheking(String strHex) {  
 List<String> list = *ConvertingToList*("0123456789ABCDEFabcdef");  
 List<String> listTemp = *ConvertingToList*(strHex);  
 return list.containsAll(listTemp);  
 }  
 public static int ConvertingDemical(String strHex) {  
 return Integer.*parseInt*(strHex, 16);  
 }  
 public static void main(String[] args) {  
 Scanner scan = new Scanner(System.*in*);  
 String str = scan.nextLine();  
 if(str.length() <= 0){  
 throw new IndexOutOfBoundsException("Input Exception");  
 }  
 if(*Cheking*(str)) {  
 System.*out*.println(*ConvertingDemical*(str));  
 } else {  
 throw new InputMismatchException("Invalid input");  
 }  
 }  
}

**Unity Test:**

package task39;  
  
import org.junit.Before;  
import org.junit.Test;  
import java.util.\*;  
import static junit.framework.TestCase.*assertEquals*;  
  
public class Task\_39Test {  
 Task\_39 TestObject = new Task\_39();  
 List<String> listB = new ArrayList<>();  
 String str2;  
  
 @Before  
 public void setUp() throws Exception {  
 str2 = "2F6";  
 listB = Arrays.*asList*("2", "F", "6");  
 }  
  
 @Test  
 public void convertingToList() {  
 List<String> expected = TestObject.*ConvertingToList*(str2);  
 List<String> actual = new ArrayList<>(Arrays.*asList*("2", "F", "6"));  
 *assertEquals*(expected, actual);  
 }  
 @Test  
 public void cheking() {  
 boolean expected = TestObject.*Cheking*(str2);   
 *assertEquals*(expected, true);  
 }  
  
 @Test  
 public void convertingDemical() {  
 Integer expected = TestObject.*ConvertingDemical*(str2);  
 Integer actual = 758;  
 *assertEquals*(expected, actual);  
 }  
}