**Министерство науки и высшего образования Российской Федерации**

ФГАОУ ВО «УрФУ имени первого Президента России Б.Н.Ельцина»

ИРИТ-РТФ

Центр ускоренного обучения

**Отчет**

**о лабораторной работе №5**

**«Введение в классы, часть 1»**

Руководитель преподаватель Н.А. Архипов

Студент гр. РИЗ-100028у В.А. Кудрявцев

**Екатеринбург 2021**

ЦЕЛЬ РАБОТЫ

Целью работы является введение в работу с классами Java.

ОПИСАНИЕ ЗАДАЧИ

Составить 6 программ по представленным задачам, представить листинги программ, во вложении к отчёту приложить файлы готовых классов.

ХОД ВЫПОЛНЕНИЯ

1. Напишите программу с классом, в котором есть закрытое

символьное поле и три открытых метода. Один из методов позволяет присвоить значение полю. Еще один метод при вызове возвращает результатом код символа. Третий метод позволяет вывести в консольное окно символ (значение поля) и его код.

import java.util.Scanner;

public class example\_18\_01 {

private static char First;

private static void InChar (){

Scanner InCMD = new Scanner (System.in);

System.out.print("Введите строку: ");

First = InCMD.next().charAt(0);

System.out.println("Символ = " + First);

}

private static int Second() {

return (int) First;

}

public static void main(String[] args) {

InChar();

System.out.println("Код символа = " + Second());

}

}

2. Напишите программу с классом, у которого есть два символьных поля и метод. Он возвращает результат, и у него нет аргументов. При вызове метод выводит в консольное окно все символы из кодовой таблицы, которые находятся «между» символами, являющимися значениями полей объекта (из которого вызывается метод). Например, если полям объекта присвоены значения ‘A’ и ‘D’, то при вызове метода в консольное окно должны выводиться все символы от ‘A’ до ‘D’ включительно.

import java.util.Scanner;

public class example\_18\_02 {

private static char First;

private static char Second;

public static void Third () {

Scanner InCMD = new Scanner (System.in);

System.out.print("Введите первый символ: ");

First = InCMD.next().charAt(0);

System.out.print("Введите второй символ: ");

Second = InCMD.next().charAt(0);

System.out.println("---");

for (int i = (int) First; i<= (int) Second; i++ ) {

System.out.println("Символ = " + (char) i );

}

}

public static void main(String[] args) {

Third ();

}

}

3. Напишите программу с классом, у которого есть два целочисленных поля. В классе должны быть описаны конструкторы, позволяющие создавать объекты без передачи аргументов, с передачей одного аргумента и с передачей двух аргументов.

import java.util.Scanner;

public class example\_18\_03 {

public static void main(String[] args)

{

Integer\_Test it = new Integer\_Test();

System.out.print("Введите значение для конструктора с одним аргументом: ");

Integer\_Test itDouble = new Integer\_Test(vvod());

System.out.println("Введите два значения для конструктора с двумя аргументами: ");

Integer\_Test itDoubleTwo = new Integer\_Test(vvod(), vvod());

}

public static double vvod()

{

Scanner input\_value = new Scanner(System.in);

var value = input\_value.nextDouble();

return value;

}

}

class Integer\_Test

{

double width, height;

Integer\_Test() {

this.width = this.height = -1.0;

System.out.println("Конструктор без аргументов: " + this.width + " " + this.height);

}

Integer\_Test(double A) {

this.width = this.height = A;

System.out.println("Конструктор с одним аргументом: " + this.width + " " + this.height);

}

Integer\_Test(double B, double C) {

this.width = B;

this.height = C;

System.out.println("Конструктор с двумя аргументами: " + this.width + " " + this.height);

}

}

4. Напишите программу с классом, у которого есть символьное и целочисленное поле. В классе должны быть описаны версии конструктора с двумя аргументами (целое число и символ – определяют значения полей), а также с одним аргументом типа double. В последнем случае действительная часть аргумента определяет код символа (значение символьного поля), а дробная часть (с учетом десятых и сотых) определяет значение целочисленного поля. Например, если аргументом передается число 65.1267, то значением символьного поля будет символ ‘A’ с годом 65, а целочисленное поле получит значение 12 (в дробной части учитываются только десятые и сотые).

import java.util.Scanner;

public class example\_18\_04

{

public static void main(String[] args)

{

CharAndInteger cai = new CharAndInteger( inputChar(), (int) inputDouble() );

cai = new CharAndInteger(inputDouble());

System.out.println(cai.result());

}

public static double inputDouble ()

{

Scanner input\_value = new Scanner(System.in);

var value = input\_value.nextDouble();

return value;

}

public static char inputChar()

{

Scanner input\_value = new Scanner(System.in);

var value = input\_value.next();

return value.charAt(0);

}

}

class CharAndInteger

{

char first;

int second;

CharAndInteger(char a, int b) {

this.first = a;

this.second = b;

}

CharAndInteger(double s) {

this.first = (char)((int)s);

double res = (s - (double)((int)s)) \* 100.0D;

this.second = (int)res;

}

String result() {

char var10000 = this.first;

return var10000 + Integer.toString(this.second);

}}

5. Напишите программу с классом, у которого есть закрытое

целочисленное поле. Значение полю присваивается с помощью открытого метода. Методу аргументом может передаваться целое число, а также метод может вызываться без аргументов. Если методы вызывается без аргументов, то поле получает нулевое значение. Если метод вызывается с целочисленным аргументом, то это значение присваивается полю. Однако если переданное аргументом методу значение превышает 100, то значением полю присваивается число 100. Предусмотрите в классе конструктор, который работает по тому же принципу что и метод для присваивания значения полю. Также в классе должен быть метод, позволяющий проверить значение поля.

import java.util.Scanner;

public class example\_18\_05

{

public static void main(String[] args)

{

System.out.print("Введите целочисленное значение переменной: ");

test\_01 t = new test\_01();

t = new test\_01(inputInteger());

System.out.println(t.result());

}

public static int inputInteger()

{

Scanner input\_chislo = new Scanner(System.in);

var value = input\_chislo.nextInt();

return value;

}

}

class test\_01

{

private int num;

public test\_01() {

this.num = 0;

}

public test\_01(int i) {

if (i <= 100) {

this.num = i;

} else {

this.num = 100;

}

}

int result() {

return this.num;

}

}

6. Напишите программу с классом, в котором есть два закрытых целочисленных поля (назовем их max и min). Значение поля max не может быть меньше значения поля min. Значения полям присваиваются с помощью открытого метода. Метод может вызываться с одним или двумя целочисленными аргументами. При вызове метода значения полям присваиваются так: сравниваются текущие значения полей и значения аргументов, переданных методу. Самое большое из значений присваивается полю max, а самое маленькое из значений присваивается полю min. Предусмотрите конструктор, который работает по тому же принципу, что и метод для присваивания значений полям. В классе также должен быть метод, отображающий в консольном окне значения полей объекта.

import java.util.Scanner;

class example\_18\_06

{ public static void main(String[] args)

{

Test t1 = new Test(inputInteger(), inputInteger());

t1 = new Test(inputInteger());

System.out.println(t1.result());

}

public static int inputInteger()

{

Scanner input\_value = new Scanner(System.in);

var value = input\_value.nextInt();

return value;

}

}

class Test

{

private int min = 0;

private int max = 0;

public Test(int i, int k) {

int min, max;

min = (i < k)? i: k;

max = (i < k)? k: i;

this.max = (max > this.max)? max : min;

this.min = (min < this.min)? min : max;

}

public Test(int i) {

min = (i < min)? i: 0;

max = (i > max)? i: 0;

}

String result() {

String var10000 = Integer.toString(this.min);

return var10000 + " / " + Integer.toString(this.max);

}

}

ВЫВОДЫ

В Java можно создавать классы и методы, а также поля, в которые могут вноситься значения с клавиатуры и вычисляемые виртуальной машиной, значения могут быть выведены на экран или использоваться другими вложенными методами.