**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Уральский государственный университет имени первого президента России

Б. Н. Ельцина»

Институт радиоэлектроники и информационных технологий

Центр ускоренного обучения

**Отчёт по лабораторной работе №9**

**«Наследование. Обработка исключительных ситуаций»**

Руководитель ст. преподаватель Н. А. Архипов

Студент гр. РИЗ-300028у В. А. Кудрявцев

**Екатеринбург 2023**

**Лабораторная работа №9.**

**«Наследование. Обработка исключительных ситуаций»**

*Цель:* знакомство с иерархией классов исключений и получение навыков обработки ошибок.

*Описание задачи:* представить все примеры с описаниями вызываемых команд, а также 6 проектов заданий из задания 2.

**Ход выполнения задач:**

1. ***Выполнить примеры 1-14 лабораторной работы, дав письменно объяснения (в комментариях к коду) последовательности выполняемых команд.***

//Сгенерировано и перехвачено RuntimeException

package job1;

public class example\_18\_91 {

public static void main(String[] args) {

try {

System.out.println("0");

//генерация ошибки

throw new RuntimeException("Непроверяемая ошибка");

//перехватывается созданное исключение

} catch (RuntimeException e) {

//выводится надпись 1 и созданная ошибки

System.out.println("1 "+ e);

}

System.out.println("2");

}

}

//Исключение перехвачено перехватчиком предка.

//IDE не хотела запускать код с выводом 1 в консоль

package job1;

public class example\_18\_92 {

public static void main(String[] args) {

try {

System.out.println("0");

//создается ошибка

throw new RuntimeException("Непроверяемая ошибка");

// System.out.println("1");

//перехватывается ошибка

} catch (Exception e) {

System.out.println("2 "+ e );

}

System.out.println("3");

}

}

//Перехват исключения подходящим классом.

package job1;

public class example\_18\_93 {

public static void main(String[] args) {

try {

System.out.println("0");

//создание ошибки

throw new RuntimeException("Ошибка!");

//ожидание ошибки NullPointerException но сгенерированная ошибка другая

} catch (NullPointerException e) {

System.out.println("1" );

//ожидание (и последующие выполнение) нужной ошибки

}catch (RuntimeException e) {

System.out.println("2" );

//ожидание другой ошибки

}catch (Exception e) {

System.out.println("3" );

}

System.out.println("4");

}

}

// Перехват исключения подходящим классом.

package job1;

public class example\_18\_94 {

public static void main(String[] args) {

try {

System.out.println("0");

//генерирование ошибки RuntimeException

throw new RuntimeException("ошибка");

//проверка на ошибку NullPointerException

} catch (NullPointerException e) {

System.out.println("1" );

//проверка на ошибку группы Exception

//выполнение этого блока, потому что ошибка RuntimeException относится к группе Exception

}catch (Exception e) {

System.out.println("2" );

//проверка на ошибку группы Error

}catch (Error e) {

System.out.println("3" );

}

System.out.println("4");

}

}

//Исключение не перехвачено.

package job1;

public class example\_18\_95 {

public static void main(String[] args) {

try {

System.out.println("0");

//создание ошибки RuntimeException

throw new RuntimeException("Ошибка!");

//перехват другой ошибки

} catch (NullPointerException e) {

System.out.println("1" );

}

System.out.println("2");

//ошибка осталась не перехваченной

}

}

//Последовательность перехвата должна

//соответствовать иерархии классов исключений. Предок не должен

//перехватывать исключения раньше потомков. Указанный пример

//выдает ошибку компилятора. Программу запустить невозможно.

//IDE потребовала поменять местами исключения, вызывающие 2 и 3,

// так как сначала перехватывалась общее, после частности

package job1;

public class example\_18\_96 {

public static void main(String[] args) {

try {

System.out.println("0");

throw new NullPointerException("ошибка");

} catch (ArithmeticException e) {

System.out.println("1" );

} catch (RuntimeException e) {

System.out.println("2" );

} catch (Exception e) {

System.out.println("2" );

}

System.out.println("4");

}

}

/\*Нельзя перехватить брошенное исключение с помощью чужого catch, даже если перехватчик подходит\*/

package job1;

public class example\_18\_97 {

public static void main(String[] args) {

try {

System.out.println("0");

throw new NullPointerException("ошибка");

} catch (NullPointerException e) {

}

System.out.println("1" );

//создан новый обработчик для перехвата

try {

throw new ArithmeticException();

}catch (ArithmeticException e) {

System.out.println("2" );

}

System.out.println("3");

}

}

//Генерация исключения в методе.

package job1;

public class example\_18\_98 {

//метод m

public static int m(){

try {

System.out.println("0");

//генерация ошибки

throw new RuntimeException();

//эта часть всегда будет отрабатываться

} finally {

System.out.println("1");

}

}

public static void main(String[] args) {

//вызов метода m

System.out.println(m());

}

}

//Генерация исключительной ситуации в методе и

//дополнительное использование оператора return.

package job1;

public class example\_18\_99 {

public static int m(){

try {

System.out.println("0");

return 55; // выход из метода

//даже при выходе из метода, этот блок будет обрабатываться

} finally {

System.out.println("1");

}

}

public static void main(String[] args)

{

System.out.println(m());

}

}

//Генерация исключительной ситуации в методе.

//Использование оператора return в секциях try и finally.

package job1;

public class example\_18\_100 {

public static int m(){

try {

System.out.println("0");

return 15;

//этот блок выполнится обязательно и именно этот return сработает

} finally {

System.out.println("1");

return 20;

}

}

public static void main(String[] args) {

System.out.println(m());

}

}

package job1;

public class example\_18\_101 {

public static void main(String[] args) {

try {

System.out.println("0");

//генерируются ошибки

throw new NullPointerException("Ошибка!");

//обрабатываются ошибки

} catch (NullPointerException e) {

System.out.println("1" );

//этот блок все равно выполняется

}finally {

System.out.println("2" );

}

System.out.println("3");

}

}

//Исключение IllegalArgumentException – неверные аргументы.

package job1;

public class example\_18\_102 {

public static void m(String str, double chislo){

if (str==null) {

/\*генерация ошибки и вылет программы, потому что обрабатывать нечем (в прикладной программе) и обработана ошибки стандартным обработчиком\*/

throw new IllegalArgumentException("Строка введена неверно!");

}

if (chislo>0.001) {

throw new IllegalArgumentException("Неверное число!");

}

}

public static void main(String[] args) {

m(null,0.000001);

}

}

//Пример работы с аргументами метода main.

package job1;

public class example\_18\_103 {

public static void main(String[] args) {

try {

int l = args.length;

System.out.println("Размер массива= " + l);

//генерация ошибки более естественным образом

int h=10/l;

args[l + 1] = "10";

//обработка ошибки

} catch (ArithmeticException e) {

System.out.println("Деление на ноль!");

// не выполняемый блок

}catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Индекс не существует!");

}

}

}

/\* Обработка исключения, порожденного одним методом m() в другом (в методе main)\*/.

package job1;

public class example\_18\_104 {

//что бы вызываемый метод не умер оповещаем об исключении через throws ArithmeticException

public static void m(int x) throws ArithmeticException{

int h=10/x;

}

public static void main(String[] args) {

try {

int l = args.length;

System.out.println("Размер массива= " + l);

m(l);

//обработка ошибки

} catch (ArithmeticException e) {

System.out.println("Ошибка !: Деление на ноль!");

}

}

}

1. ***Выполнить все задания из таблицы2:***

***– определить экспериментально, ошибки каких классов будут сгенерированы;***

***– создать обработчики исключительных ситуаций с использованием выявленных классов и всех секций конструкции обработчика с соответствующими сообщениями, позволяющими корректно выполнить программу.***

/\* В программе, вычисляющей среднее значение среди положительных элементов

одномерного массива (тип элементов int), вводимого с клавиатуры, могут

возникать ошибки в следующих случаях:

ввод строки вместо числа;

несоответствие числового типа данных;

положительные элементы отсутствуют.\*/

package job2;

import java.util.Scanner;

public class example\_18\_105 {

public static void main(String[] args)

{

Scanner InCMD = new Scanner(System.in);

System.out.print("Введите размер: ");

int SArr = InCMD.nextInt();

System.out.println();

int[] arr = new int[SArr];

int INT = 0;

int pos = 0;

Scanner scanValue = new Scanner(System.in);

for(int i = 0; i < arr.length; i++) {

try {

while (!scanValue.hasNextInt()) {

System.out.println("int, please!");

scanValue.nextLine();

}

INT = scanValue.nextInt();

if(INT >= 0) {

arr[i] = INT;

pos++;

}

}

catch (Exception e) {

System.out.println("Ошибка: " + e.getMessage());

return;

}

}

try {

if(pos == 0) throw new Exception("В массиве нет положительных элементов.");

}

catch (Exception e) {

System.out.println("Ошибка!: " + e.getMessage());

}

}

}

***Вывод:***

В ходе работы было получено представление о способах обработки ошибок в программах, получено представление о иерархии классов исключений и получение навыков обработки ошибок.

Не всё получилось сделать, более подробно разбирать поставленные задачи не было возможности.

Каждое решение задания сопровождено листингами программ, в архиве с классами будут добавлены соответствующие файлы.