Министерство образования и науки Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРО-ВОСТОЧНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ ИМЕНИ М.К. АММОСОВА»

Институт математики и информатики

Лабораторная работа №4   
по курсу «Алгоритмизация и программирование»

НАСЛЕДОВАНИЕ. ОБРАБОТКА ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ

**Выполнил**студент 1-го курса   
гр. ПИ-19-2   
Бережнев И.Е.

**Принял**ст. преподаватель Петров Н.В.

Якутск 2020

**Пример 4.1**

Выводится “0”

Потом вызывает ошибку RuntimeException (“Непроверяемая ошибка”)

Эту ошибку ловит catch и выводит “1” и “java.lang.RuntimeException: Непроверяемая ошибка”

И в конце выводит “2”.

**Пример 4.2**

Выводится “0”

Затем вызывается ошибка RuntimeException (“Непроверяемая ошибка”)

Затем пытается вывести “1”, затем ломается программа, выводя ошибку java: unreachable statement.

**Пример 4.3**

Выводится “0”

Затем вызывается ошибка RuntimeException (“ошибка”)

После этого она перехватывается перехватчиком предка catch (RuntimeException e), выводя “2” и игнорируя NullPointerException и пропуская Error

И в конце выводит “4”.

**Пример 4.4**

Выводится “0”

Затем вызывается ошибка RuntimeException (“ошибка”)

После этого она перехватывается перехватчиком предка catch (Exception e), выводя “2” и игнорируя NullPointerException и пропуская Error

В конце выводит “4”.

**Пример 4.5**

Выводится “0”

Затем вызывается ошибка RuntimeException (“ошибка”)

Ошибка не перехватывается (потому что NullPointerException не подходит) и вызывает ошибку RuntimeException: ошибка.

**Пример 4.6**

Выводится “0”

Затем вызывается ошибка RuntimeException (“ошибка”)

Она не перехватывается, так как последовательность перехвата не соответствует иерархии исключений

Программа ломается.

**Пример 4.7**

Выводится “0”

Вызывается ошибка NullPointerException (“ошибка”)

Ошибка перехватывается, выводит “1” и вызывается ошибка ArithmeticException

Программа выдает ошибку, так как нельзя вызывать ошибки в catch.

**Пример 4.8**

Выводится “0”

Вызывается ошибка RuntimeException

В любом случае выводится “1” так как он находится в finally

Программа выдает ошибку, так как RuntimeException не перехватывается.

**Пример 4.9**

Выводится “0”

Возвращается 55, то есть выходит из метода выводя 55, но так как в программе имеется finally с sout(“1”) сначала выведется 1, а затем 55.

**Пример 4.10**

Выводится “0”

Возвращается 15

Так как тут имеется finally, то выводится “1” и возвращается 20

В итоге выводится:

**0**

**1**

**20**

Потому что возврат 15, замещается возвратом 20 из finally.

**Пример 4.11**

Выводится “0”

Вызывается ошибка NullPointerException (“ошибка”)

Ошибка перехватывается и выводит “1”

Затем выводится “2” из finally

После всего этого выводится “3”.

**Пример 4.12**

Из main в m передается 2 значения – «str = null» и «сhislo = 0.000001»

Если str == null, то вызывается ошибка IllegalArgumentException (“Строка введена неверно”)

Выдается ошибка, потому что IllegalArgumentException не перехватили, и программа останавливается.

**Пример 4.13**

Задается переменная l = длине массива args

Выводится “Размер массива= ” длина массива args = 0

h/l – получается ошибка ArithmeticException (деление на ноль)

Присваивается String значение “10” элементу массива args с индексом l +1 (0 + 1)

Перехватывается ArithmeticException и выводится “Деление на ноль”

Выходит из try.

**Пример 4.14**

Задается переменная l = длине массива args

Выводится “Размер массива= ” длина массива args = 0

Вызывается метод m (x = l)

Там задается h = 10/x

Ошибка перехватывается и выводится “Ошибка: Деление на ноль”.

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Что такое наследование?

Наследование - это механизм, позволяющий создавать классы на основе другого класса.

2. Иерархия классов исключений, базовые классы и их статус.

Исключение – это ошибка, возникающая во время исполнения программы (но не в процессе компиляции). В JVM предусмотрена реакция на любую ошибку – автоматическое срабатывание обработчика исключений по умолчанию. В результате этого программа завершает свою работу, а пользователь видит на экране сформированную трассу стека (Stack Trace), где указывается класс, соответствующий перехваченному исключению, место расположения ошибки и последовательность вызываемых методов, через которые эта ошибка передается («летит»).

На вершине иерархии исключений стоит класс Throwable, являющийся наследником класса Object. Каждый из типов исключений является подклассом класса Throwable. Два непосредственных наследника класса Throwable – Error и Exception делят иерархию подклассов исключений на две различные ветви.

3. Типовая структура обработчика исключительной ситуации.

К механизму обработки исключений в Java имеют отношение 5 клю­чевых слов: — **try**, **catch**, **throw**, **throws** и **finally**. Схема работы этого механизма следующая. Вы пытаетесь (try) выполнить блок кода, и если при этом возникает ошибка, система возбуждает (throw) исключение, ко­торое в зависимости от его типа вы можете перехватить (catch) или пере­дать умалчиваемому (finally) обработчику.

**try** – попытаться выполнить;

**catch** – перехватить и обработать ошибку;

**finally** – окончательно (финальный блок, выполняемый всегда);

**throw** – генерация («бросание») исключения;

**throws** – пометка метода, «бросающего» исключение.

4. Последовательность перехвата ошибок родительскими и дочерними классами.

Последовательность перехвата должна соответствовать иерархии классов исключений. Предок не должен перехватывать исключения раньше потомков.

5. Как помечается метод, генерирующий исключительную ситуацию, но не обрабатывающий ее?

Если метод способен к порождению исключений, которые он не обрабатывает, он должен быть определен так, чтобы вызывающие методы могли сами предохранять от данного исключения. Для этого используется ключевое слово throws в сигнатуре метода. Это необходимо для всех исключений, кроме исключений типа Error и RuntimeException, и, соответственно, для любых их подклассов.