Задание 4.1

Пример 4.1

Сперва выводит “0”;

Затем вызывает ошибку RuntimeException(“Непроверяемая ошибка”);

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

В конце выводит “2”;

Пример 4.2

Выводится “0”;

Потом вызывается ошибка RuntimeException(“Непроверяемая ошибка”);

Затем пытается вывести “1”, после ломается программа, выводя ошибку java: unreachablestatement;

Пример 4.3

Выводится “0”;

Затем вызывается ошибка RuntimeException(“ошибка”);

После этого она перехватывается перехватчиком предка catch (RuntimeExceptione), выводит “2”, игнорирует NullPointerException и пропускает Error;

В конце выводится “4”

Пример 4.4

Выводится “0”;

Затем вызывается ошибка RuntimeException(“ошибка”);

После этого она перехватывается перехватчиком предка catch (Exceptione), выводит “2”, игнорирует NullPointerException и пропускает Error;

В конце концов выводится “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;

Меняемместамиcatch(Exceptione) иcatch(RuntimeExceptione)

Теперь программа выводит:

0

3

4

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 – это 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= ” длинна массива args = 0;

h/l – получается ошибка ArithmeticException(потому что делится 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= ” длинна массива args = 0;

Вызывается метод m(x = l);

Там задается h = 10/x;

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

Задание 4.2

Вариант 2

В программе, вычисляющей минимум среди отрицательных элементов одномерного массива (тип элементов float) и номер этого минимума, могут возникать ошибки в следующих случаях: – ввод строки вместо числа; – отрицательные элементы отсутствуют.

1.1

Cоздаем массив mas[k];

try{ создаем цикл который будет заполнять массив

создаем цикл который будет находить минимальное отрицательное число в массиве

если минимальное число = 0 то тогда вызывается ошибка RuntimeException

}

catch(ошибка InputMismatchExceptione){выводится строка “Ввод строки вместо числа”};

catch(ошибка RuntimeExceptione){выводится “Отрицательные элементы отсутствуют”};

import java.util.InputMismatchException;  
import java.util.Scanner;  
  
public class variant5\_1 {  
public static void main(String[] args) {  
try {  
 Scanner sc = new Scanner(System.*in*);  
System.*out*.println("Введите длинну массива: ");  
 int k = sc.nextInt();  
 int[] mas = new int[k];  
 for(int i = 1; i <= k; i++){  
 System.*out*.println("Введите значение " + i + " элемента");  
 int x = sc.nextInt();  
mas[i-1] = x;  
}  
int min = 0;  
 for(int i = 0; i < mas.length; i++){  
if(mas[i] < min){  
 min = mas[i];  
}  
 }  
if(min == 0){  
throw new RuntimeException();  
}else{  
 System.*out*.println("Минимальное отрицательное число = " + min);  
}  
 } catch (InputMismatchException e) {  
 System.*out*.println("Ввод строки вместо числа");  
} catch (RuntimeException e) {  
 System.*out*.println("Отрицательные элементы отсутствуют");  
}  
 }  
}

1.2

Метод mainвызывает метод massiv() перехватывая ошибки InputMismatchExceptionи RuntimeException, в первом случае выводя “Ввод строки вместо числа” и во втором случае “ Отрицательные элементы отсутствуют ”

Метод massiv() создает и массив mas[k] и вызывает методminimum(mas)

Методminimum(int[] mas) находит и выводит минимальное отрицательное число и если его нет вызывает ошибку RuntimeException

importjava.util.InputMismatchException;  
importjava.util.Scanner;  
  
publicclassvariant5\_2 {  
publicstaticvoidmassiv(){  
Scannersc = newScanner(System.*in*);  
System.*out*.println("Введите длинну массива: ");  
intk = sc.nextInt();  
int[] mas = newint[k];  
for(inti = 1; i<= k; i++){  
System.*out*.println("Введите значение " + i + " элемента");  
intx = sc.nextInt();  
mas[i-1] = x;  
}  
*minimum*(mas);  
}  
  
publicstaticvoidminimum(int[] mas){  
intmin = 0;  
for(inti = 0; i<mas.length; i++){  
if(mas[i] <min){  
min = mas[i];  
}  
 }  
if(min == 0){  
thrownewRuntimeException();  
}else{  
System.*out*.println("Минимальное отрицательное число = " + min);  
}  
 }  
  
publicstaticvoidmain(String[] args) {  
try{  
*massiv*();  
}catch(InputMismatchExceptione){  
System.*out*.println("Ввод строки вместо числа");  
}catch(RuntimeExceptione){  
System.*out*.println("Отрицательные элементы отсутствуют");  
}  
 }  
}

Контрольные вопросы

1. Наследование позволяют дополнить один класс другим, который унаследовал его.
2. …RuntimeException(unchecked) > Exception(checked) > Throwable(checked) >java.lang.Object

…Error(unchecked) > Throwable(checked) > java.lang.Object

1. try{часть кода, которая может вызвать ошибку}catch(ошибка e){при перехвате ошибки, выполняется эта часть кода}
2. По иерархии снизу вверх(вопрос 2)
3. throw