Основи програмування

Лабораторна робота №3

**Тема**: Управління потоком виконання

**Хід роботи**:

1. Повторити теоретичні відомості

2. Виконати три завдання з таблиці 2 відповідно до свого варіанту у таблиці 1.

- Кожне завдання має бути реалізовано як окремий клас.

- Кожен клас має складатись щонайменше з двох методів:

- public static void main(String[] agrs) - точка входу. Містить код, що кілька разів знаходить результат завдання при різних значеннях аргументів та параметрів. Для перевірки мають бути присутні як дозволені так і заборонені комбінації аргументів та параметрів.

- Метод, що реалізує задане завдання. Метод має перевіряти аргументи і параметри функції та у разі їх помилковості аварійно закінчувати свою роботу шляхом викидання стандартного виключення IllegalArgumentException (дивись приклад). В жодному разі цей метод не повинен напряму взаємодіяти з користувачем через консоль або інший UI (ніколи не змішуйте бізнес-логіку та користувацький інтерфейс).

- Клас може містити інші допоміжні методи.

3. Відповісти на контрольні питання

**Приклад виконання:**

Обчислити значення функції 

Програма:

package lab03;

/\*\*

\* Lab 03, Task 1 example

\*

\* @author Bukasov

\*/

public class Task1 {

/\*\*

\* Entry point. Tests the method {@code axx(a,x)} by calling the helper

\* method {@code printResults(a,x)} few times with different arguments.

\*

\* @param a the parameter of the function {@code axx(a,x)}.

\* @param x the argument of the function {@code axx(a,x)}.

\*/

public static void main(String[] args) {

printResults(0, 0);

printResults(0, 1);

printResults(1, 0);

printResults(1, 1);

printResults(2, 1);

printResults(1, 2);

printResults(2, -10);

printResults(20, 20);

printResults(21, 21);

printResults(1, 10);

printResults(-1, 10);

printResults(0, Double.MAX\_VALUE);

printResults(1, Double.MAX\_VALUE);

printResults(1, Double.NaN);

}

/\*\*

\* Returns the value of the first argument multiplied by the square of the

\* second argument. Special cases:

\* <ul><li>If the first argument is less than 0 or is greater than 20, then

\* exception IllegarArgumentException is thrown.</ul>

\*

\* @param a the parameter of the function.

\* @param x the argument of the function.

\* @return a \* x \* x.

\* @exception IllegalArgumentException if a&lt;0 or a&gt;20

\*/

public static double axx(double a, double x) {

if (a < 0 || a > 20) {

throw new IllegalArgumentException("param a = " + a);

}

return a \* x \* x;

}

/\*\*

\* Helper method for printing result of {@code axx(a,x)}.

\*

\* @param a the parameter of the function.

\* @param x the argument of the function.

\*/

static void printResults(double a, double x) {

System.out.print("a:" + a + " x:" + x + " result:");

try {

System.out.println(axx(a, x));

} catch (IllegalArgumentException e) {

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

}

}

}

Результати роботи програми:

a:0.0 x:0.0 result:0.0

a:0.0 x:1.0 result:0.0

a:1.0 x:0.0 result:0.0

a:1.0 x:1.0 result:1.0

a:2.0 x:1.0 result:2.0

a:1.0 x:2.0 result:4.0

a:2.0 x:-10.0 result:200.0

a:20.0 x:20.0 result:8000.0

a:21.0 x:21.0 result:EXCEPTION! param a = 21.0

a:1.0 x:10.0 result:100.0

a:-1.0 x:10.0 result:EXCEPTION! param a = -1.0

a:0.0 x:1.7976931348623157E308 result:0.0

a:1.0 x:1.7976931348623157E308 result:Infinity

a:1.0 x:NaN result:NaN

Таблиця 1. Таблиця варіантів

|  |  |  |  |
| --- | --- | --- | --- |
| Варіант | Завдання № 1 | Завдання № 2 | Завдання № 3 |
| 1 | 1 | 6 | 11 |
| 2 | 2 | 7 | 12 |
| 3 | 3 | 8 | 13 |
| 4 | 4 | 9 | 14 |
| 5 | 5 | 10 | 15 |
| 6 | 1 | 7 | 13 |
| 7 | 2 | 8 | 14 |
| 8 | 3 | 9 | 15 |
| 9 | 4 | 10 | 11 |
| 10 | 5 | 6 | 12 |
| 11 | 1 | 8 | 15 |
| 12 | 2 | 9 | 11 |
| 13 | 3 | 10 | 12 |
| 14 | 4 | 6 | 13 |
| 15 | 5 | 7 | 14 |
| 16 | 1 | 9 | 12 |
| 17 | 2 | 10 | 13 |
| 18 | 3 | 6 | 14 |
| 19 | 4 | 7 | 15 |
| 20 | 5 | 8 | 11 |
| 21 | 1 | 10 | 14 |
| 22 | 2 | 6 | 15 |
| 23 | 3 | 7 | 11 |
| 24 | 4 | 8 | 12 |
| 25 | 5 | 9 | 13 |
| 26 | 1 | 10 | 12 |
| 27 | 2 | 9 | 14 |
| 28 | 3 | 8 | 11 |
| 29 | 4 | 7 | 13 |
| 30 | 5 | 6 | 15 |

Таблиця 2. Таблиця завдання

|  |  |
| --- | --- |
| **Завдання** | **Вирази** |
|
| 1 | Обчислити суму ряду: |
| 2 | Обчислити суму ряду: |
| 3 | Обчислити суму ряду: |
| 4 | Обчислити суму ряду: |
| 5 | Обчислити суму ряду: |
| 6 | Обчислити значення функції: |
| 7 | Обчислити значення функції: |
| 8 | Обчислити значення функції: |
| 9 | Обчислити значення функції: |
| 10 | Обчислити значення функції: |
| 11 | Обчислити нескінченну суму  із заданою точністю ε (ε>0). Вважати, що необхідна точність досягнута, якщо черговий доданок виявився по модулю меншим, ніж ε. Цей і всі наступні доданки можна не враховувати. |
| 12 | Обчислити нескінченну суму  із заданою точністю ε (ε>0). Вважати, що необхідна точність досягнута якщо черговий доданок виявився по модулю меншим, ніж ε. Цей і всі наступні доданки можна не враховувати. |
| 13 | Обчислити нескінченну суму  із заданою точністю ε (ε>0). Вважати, що необхідна точність досягнута якщо черговий доданок виявився по модулю меншим, ніж ε. Цей і усі наступні доданки можна не враховувати. |
| 14 | Обчислити нескінченну суму  із заданою точністю ε (ε>0). Вважати, що необхідна точність досягнута якщо черговий доданок виявився по модулю меншим, ніж ε. Цей і усі наступні доданки можна не враховувати. |
| 15 | Обчислити нескінченну суму  із заданою точністю ε (ε>0). Вважати, що необхідна точність досягнута якщо черговий доданок виявився по модулю меншим, ніж ε. Цей і усі наступні доданки можна не враховувати. |

**Контрольні питання**:

1. Що таке структурне програмування?

2. Назвіть основні конструкції структурного програмування.

3. Назвіть основні принципи структурного програмування.

4. Поясніть, чому в більшості мов програмування не радять використовувати оператор GOTO, а в мові Java його взагалі немає?

5. Що таке спагеті-код?

6. Що таке проектування зверху-вниз? В чому його переваги?

7. Що таке область видимості ідентифікатора?

8. Що таке блок? Як блоки впливають на область видимості?

9. Наведіть приклади, у яких випадках краще використовувати наступні оператори та конструкції:

- if

- if-else

- ланцюги if-else if -else if...

- switch

- тернарна умовна операція ?:

10. Наведіть приклади, у яких випадках краще використовувати наступні оператори та конструкції:

- while

- do-while

- for

11. Чим оператор break відрізняється від оператора continue?

12. Навіщо у мові Java є мітки, якщо немає goto?

13. До якого результату призведе виконання наступного фрагменту коду? Перевірте. Поясніть.

boolean a = false;

boolean b = false;

if (a=false) {

System.out.println("a is false");

}

if (b=true) {

System.out.println("b is true");

}

if (a=b) {

System.out.println("a = b");

}

14. До якого результату призведе виконання наступного фрагменту коду? Перевірте. Поясніть.

int a = 1;

a++;

++a;

switch(a) {

case 1: System.out.println("1");

case 2: System.out.println("2");

case 3: System.out.println("3");

case 4: System.out.println("4");

}

15. До якого результату призведе виконання наступного фрагменту коду (true/false)? Перевірте. Поясніть.

double sum = 0;

// Let's find: 1 + 1/2 + 1/3 + 1/4 + 1/5 + ...

for(int i=1; i<10; i++) {

sum = sum + 1/i;

}

System.out.println(sum > 1);