

ОДЕССКИЙ НАЦИОНАЛЬНЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИНСТИТУТ КОМПЬЮТЕРНЫХ СИСТЕМ
Кафедра «Информационные технологии»

Лабораторная работа №1

по дисциплине: «Объектно-ориентированное программирование»

на тему: «Разработка консольных приложений»

Вариант 13

Выполнил:

ст. гр. НАД-191

Краковский В.А.

Проверили:

д. Рудниченко М.Д.

ст. пр. Павлов О.А.

Одесса 2020

Оглавление

ВВЕДЕНИЕ.....	3
ТЕОРЕТИЧЕСКАЯ ЧАСТЬ.....	4
ПРАКТИЧЕСКАЯ ЧАСТЬ.....	5
ВЫВОД.....	8
ЛИТЕРАТУРА.....	9

ВВЕДЕНИЕ

Целью данной лабораторной работы является изучение процесса разработки консольных приложений на языке программирования Java с помощью IDE для реализации простейшего приложения.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Консольные приложения JAVA представляют собой созданный и откомпилированный программистом класс, содержащий точку входа.

Класс System

Класс System содержит набор полезных статических методов и полей системного уровня. Экземпляр этого класса не может быть создан или получен.

Наиболее широко используемой возможностью, предоставляемой System, является стандартный вывод, доступный через переменную System.out.

Класс String

Класс String содержит основные методы для работы со строками:

concat(String s) или + – слияние строк;

equals(Object ob), equalsIgnoreCase(String s) – сравнение строк с учетом и без учета регистра;

compareTo(String s), compareToIgnoreCase (String s) – лексикографическое сравнение строк с учетом и без учета регистра;

contentEquals(StringBuffer ob) – сравнение строки и содержимого объекта типа StringBuffer;

charAt(int n)– извлечение из строки символа с указанным номером (нумерация с нуля);

substring(int n, int m)- извлечение из строки подстроки длины m-n, начиная с позиции n;

length() – определение длины строки;

valueOf(объект) – преобразование примитивного объекта к строке;

toUpperCase() / toLowerCase() – преобразование всех символов вызывающей строки в верхний/нижний регистр;

replace(char c1, char c2) – замена в строке всех вхождений первого символа вторым символом;

ПРАКТИЧЕСКАЯ ЧАСТЬ

1. Условие

13. Выпуклый многоугольник задан на плоскости перечислением координат вершин в порядке обхода его границы. Определить площадь многоугольника.

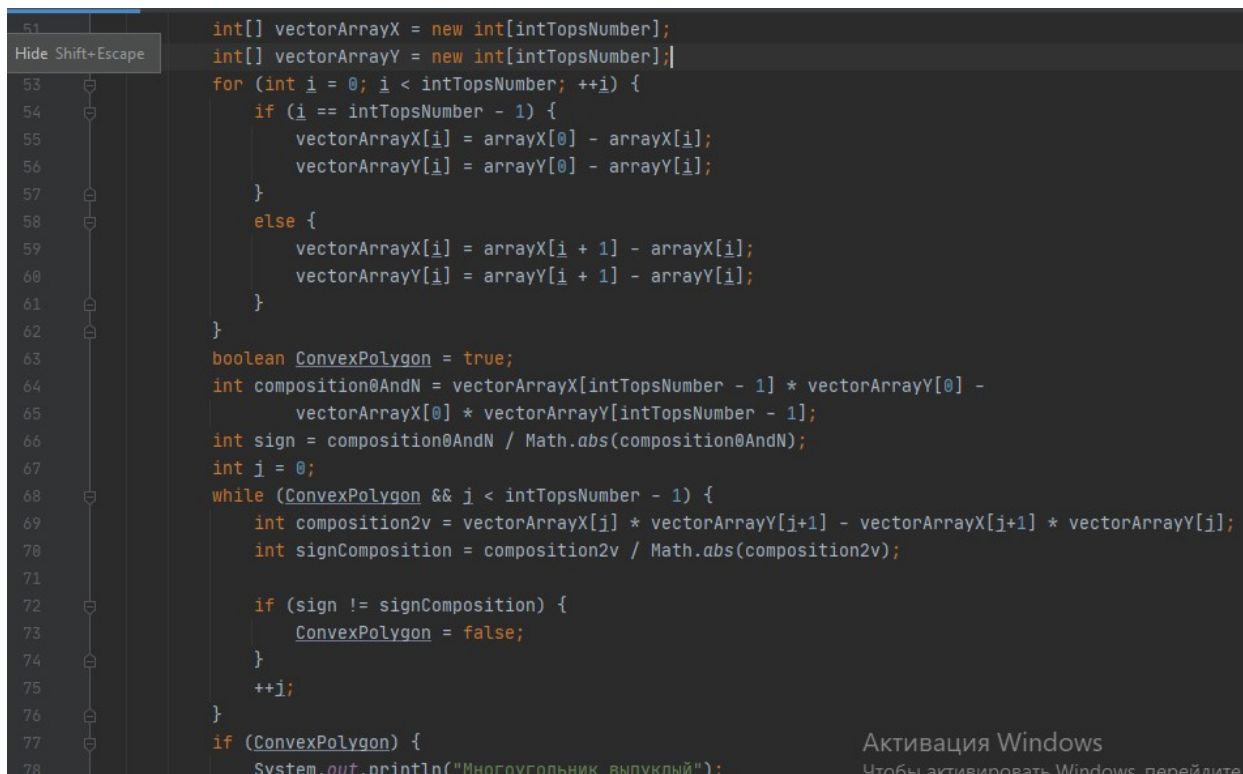
Рисунок 1: Условие задания

2. Вводим координаты с клавиатуры:

```
13 //-----Перечисляем коор-ты вершин-----//
14 System.out.println("Введите количество вершин многоугольника:");
15 Scanner topsNumber = new Scanner(System.in);
16
17 //Проверка вводимого числа
18 if (topsNumber.hasNextInt()) {
19     int intTopsNumber = topsNumber.nextInt();
20
21     if (intTopsNumber > 2) {
22         int[] arrayX = new int[intTopsNumber];
23         int[] arrayY = new int[intTopsNumber];
24
25         for (int i = 0; i < intTopsNumber; ++i) {
26             boolean checkInput = false;
27             while (!checkInput) {
28                 System.out.println("Введите координаты (X, Y) вершины #" + i);
29                 Scanner X = new Scanner(System.in);
30                 Scanner Y = new Scanner(System.in);
31
32                 if (X.hasNextInt() && Y.hasNextInt()) {
33                     arrayX[i] = X.nextInt();
34                     arrayY[i] = Y.nextInt();
35                     checkInput = true;
36                 }
37                 else {
38                     System.out.println("Вы ввели не целое число!");
39                 }
40             }
41         }
42     }
43 }
```

Рисунок 2: Перечисление координат вершин многоугольника

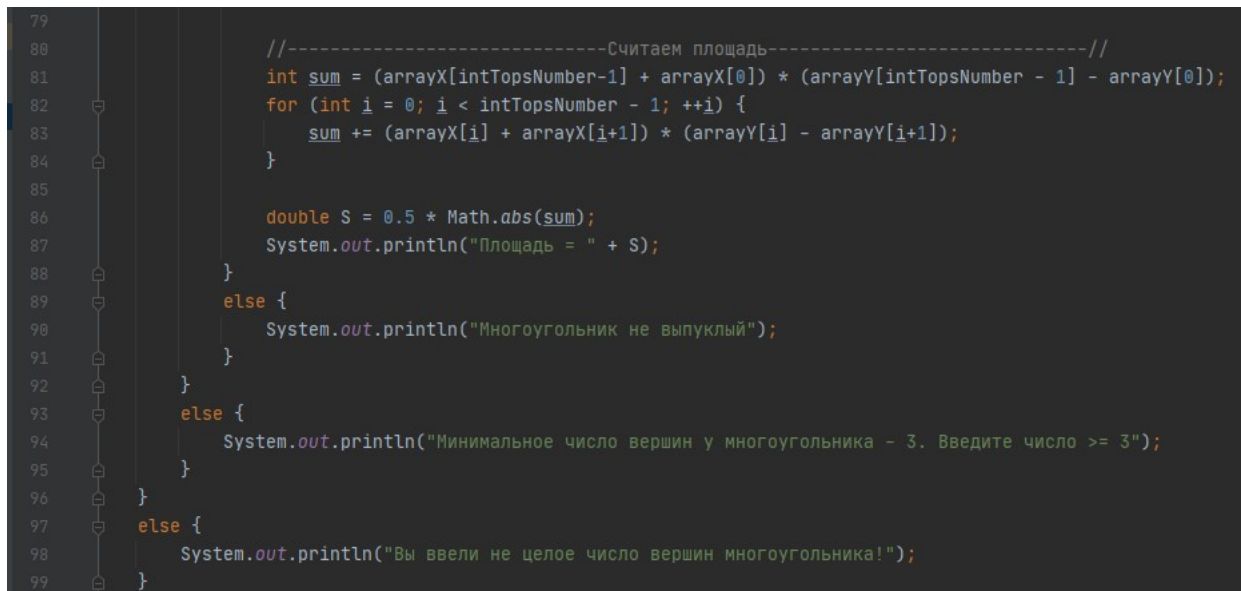
3. Проверяем многоугольник на выпуклость:



```
51      int[] vectorArrayX = new int[intTopsNumber];
52      int[] vectorArrayY = new int[intTopsNumber];
53      for (int i = 0; i < intTopsNumber; ++i) {
54          if (i == intTopsNumber - 1) {
55              vectorArrayX[i] = arrayX[0] - arrayX[i];
56              vectorArrayY[i] = arrayY[0] - arrayY[i];
57          }
58          else {
59              vectorArrayX[i] = arrayX[i + 1] - arrayX[i];
60              vectorArrayY[i] = arrayY[i + 1] - arrayY[i];
61          }
62      }
63      boolean ConvexPolygon = true;
64      int composition0AndN = vectorArrayX[intTopsNumber - 1] * vectorArrayY[0] -
65          vectorArrayX[0] * vectorArrayY[intTopsNumber - 1];
66      int sign = composition0AndN / Math.abs(composition0AndN);
67      int j = 0;
68      while (ConvexPolygon && j < intTopsNumber - 1) {
69          int composition2v = vectorArrayX[j] * vectorArrayY[j+1] - vectorArrayX[j+1] * vectorArrayY[j];
70          int signComposition = composition2v / Math.abs(composition2v);
71
72          if (sign != signComposition) {
73              ConvexPolygon = false;
74          }
75          ++j;
76      }
77      if (ConvexPolygon) {
78          System.out.println("Многоугольник выпуклый");
```

Рисунок 3: Проверка многоугольника на выпуклость

4. Считаем искомую площадь:



```
79
80      //-----Считаем площадь-----//
81      int sum = (arrayX[intTopsNumber-1] + arrayX[0]) * (arrayY[intTopsNumber - 1] - arrayY[0]);
82      for (int i = 0; i < intTopsNumber - 1; ++i) {
83          sum += (arrayX[i] + arrayX[i+1]) * (arrayY[i] - arrayY[i+1]);
84      }
85
86      double S = 0.5 * Math.abs(sum);
87      System.out.println("Площадь = " + S);
88  }
89  else {
90      System.out.println("Многоугольник не выпуклый");
91  }
92  }
93  else {
94      System.out.println("Минимальное число вершин у многоугольника - 3. Введите число >= 3");
95  }
96  }
97  else {
98      System.out.println("Вы ввели не целое число вершин многоугольника!");
99  }
```

Рисунок 4: Подсчёт площади выпуклого многоугольника

5. Результат:

```
Введите количество вершин многоугольника:
5
Введите координаты (X, Y) вершины #0
-2
2
Введите координаты (X, Y) вершины #1
1
4
Введите координаты (X, Y) вершины #2
3
0
Введите координаты (X, Y) вершины #3
1
-1
Введите координаты (X, Y) вершины #4
-1
-1
Координаты вершины №0: (-2, 2)
Координаты вершины №1: (1, 4)
Координаты вершины №2: (3, 0)
Координаты вершины №3: (1, -1)
Координаты вершины №4: (-1, -1)
Многоугольник выпуклый
Площадь = 15.5
```

Рисунок 5: Тестовые данные

ВЫВОД

На этой лабораторной работе я узнал каким образом создавать простейшее приложение с помощью консоли и простейших операций на языке программирования Java. Как результат, была реализована программа для подсчёта площади выпуклого многоугольника, которую возможно использовать в соответствующих математических подсчётах.

ЛИТЕРАТУРА

1. Выпуклый многоугольник [Электронный ресурс] - https://ru.wikipedia.org/wiki/Выпуклый_многоугольник
2. Выпуклость многоугольника [Электронный ресурс] - <https://tux.org.ua/vy-puklost-mnogougol-nika/>