Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

*Кафедра програмного забезпечення автоматизованих систем*

**Лабораторна робота №3**

Тема:

**«Cтатистична обробка неперервних даних»**

***Виконали*:**

ст. групи ПіТ-15-3

Свирид О.Б., Турчин М.І.

***Перевірив:***

Незамай Б.С.

Івано-Франківськ

2015

**Завдання:** задатись вибіркою із 100N елементів (вивід стандартного генератора випадкових чисел без округлень) в діапазоні від 0 до N включно. Побудувати полігон та гістограму. Для побудови графіки рекомендується використовувати стандартні бібліотеки..

**Хід роботи**

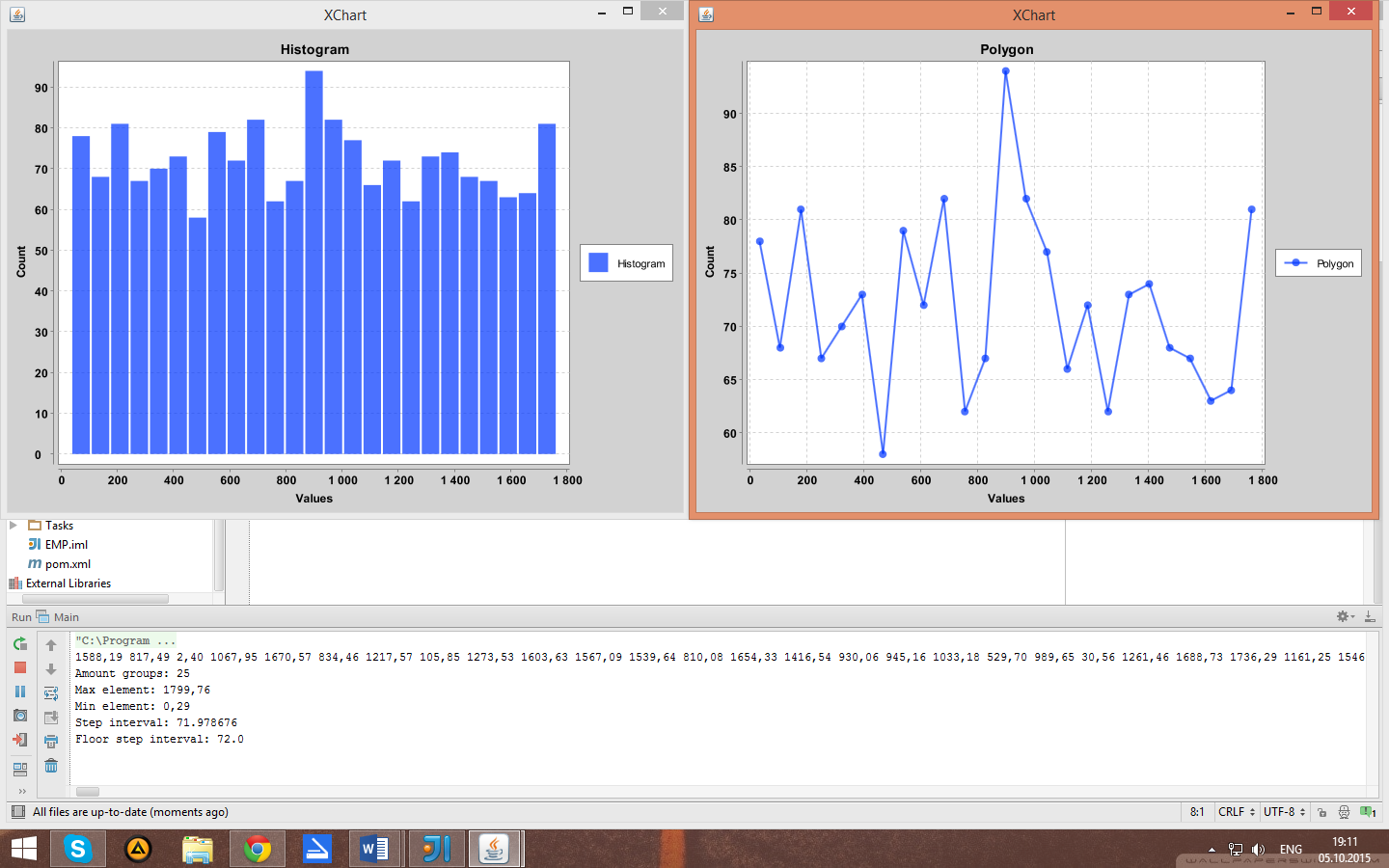
Код програми на мові Java:

**public class** Calculate {  
  
 **protected static** Random *random* = **new** Random();  
  
 */\*\*  
 \* Generate random float numbers in range  
 \** ***@param min*** *Min range  
 \** ***@param max*** *Max range  
 \** ***@return*** *Random value in range above  
 \*/* **public static float** randomInRange(**float** min, **float** max) {  
 **float** range = max - min;  
 **float** scaled = (**float**)*random*.nextDouble() \* range;  
 **float** shifted = scaled + min;  
 **return** shifted; *// == (rand.nextDouble() \* (max-min)) + min;* }  
  
 */\*\*  
 \* Initialize array random numbers  
 \*/* **protected void** initializeArray(**float**[] arr, **int** amountItems){  
 **for**(**int** i = 0; i < arr.**length**; i++) {  
 arr[i] = *randomInRange*(0, amountItems);  
 System.***out***.format(**"%2.2f "**, arr[i]);  
 }  
 }  
  
 */\*\*  
 \* Return amount of groups  
 \** ***@return*** *\*/* **protected static int** getH(**int** amountItems) {  
 **return** (**int**) Math.*floor*(*sterdjessFormule*(amountItems));  
 }  
  
 */\*\*  
 \* Return step of interval  
 \** ***@return*** *\*/* **protected float** getStepInterval(**float**[] arr, **int** amountItems) {  
 **float** stepInterval = (*getMaxElement*(arr) - *getMinElement*(arr)) / *getH*(amountItems);  
 System.***out***.println(**"Step interval: "** + stepInterval);  
 **return** Math.*round*(stepInterval);  
 }  
  
 */\*\*  
 \* Get maximum element from array  
 \** ***@param arr*** *Array of numbers  
 \** ***@return*** *Maximum element from array  
 \*/* **protected static float** getMaxElement(**float**[] arr) {  
  
 *//set max element* **float** max = 0;  
  
 *//flag if value is first in collection* **boolean** isFirstEnter = **false**;  
  
 *//iterating collection of frequencies* **for**(**int** i = 0; i < arr.**length**; i++) {  
  
 *//get concrete value* **float** value = arr[i];  
  
 *//check if iteration is first* **if**(!isFirstEnter) {  
  
 *//set max value first value from array* max = value;  
  
 *//flag that was first iteration* isFirstEnter = **true**;  
 }  
  
 *//check if max value not largest that previous value* **if**(max < value) {  
  
 *//set max value to actual value* max = value;  
 }  
 }  
  
 *//return max value or values* **return** max;  
 }  
  
 */\*\*  
 \* Get maximum element from array  
 \** ***@param arr*** *Array of numbers  
 \** ***@return*** *Maximum element from array  
 \*/* **protected static float** getMinElement(**float**[] arr) {  
  
 *//set min element* **float** min = 0;  
  
 *//flag if value is first in collection* **boolean** isFirstEnter = **false**;  
  
 *//iterating collection of frequencies* **for**(**int** i = 0; i < arr.**length**; i++) {  
  
 *//get concrete value* **float** value = arr[i];  
  
 *//check if iteration is first* **if**(!isFirstEnter) {  
  
 *//set min value first value from array* min = value;  
  
 *//flag that was first iteration* isFirstEnter = **true**;  
 }  
  
 *//check if min value largest that previous value* **if**(min > value) {  
  
 *//set min value to actual value* min = value;  
 }  
 }  
  
 *//return min value or values* **return** min;  
 }  
  
 */\*\*  
 \* Sterjess Formule  
 \** ***@param amountItems*** *Amount items in array  
 \** ***@return*** *Result  
 \*/* **private static final double** sterdjessFormule(**int** amountItems) {  
 **return** (1 + 3.322 \* Math.*log*(amountItems));  
 }  
  
 */\*\*  
 \* Convert array to List(need for draw graphics)  
 \** ***@param amountItems*** *Amount items in array  
 \** ***@param arr*** *Array of items  
 \** ***@return*** *List of items  
 \*/* **protected** List<Double> getGaussianData(**int** amountItems, **float**[] arr) {  
  
 List<Double> data = **new** ArrayList<Double>(amountItems);  
 **for** (**int** i = 0; i < amountItems; i++) {  
 data.add((**double**)arr[i]);  
 }  
 **return** data;  
 }  
}

**public class** Drawing {  
 **protected** Chart buildHistogram(List<Double> data, **int** h, **float** max, **float** min) {  
 Chart chart = **new** ChartBuilder()  
 .chartType(StyleManager.ChartType.***Bar***)  
 .width(1000).height(600).title(**"Histogram"**)  
 .xAxisTitle(**"Values"**).yAxisTitle(**"Count"**).build();  
  
 Histogram histogram = **new** Histogram(data, h, (**double**)min, (**double**)max);  
 chart.addSeries(**"Histogram"**, histogram.getxAxisData(), histogram.getyAxisData());  
 **return** chart;  
 }  
  
 **protected** Chart buildPolygon(List<Double> data, **int** h, **float** max, **float** min) {  
 Chart chart = **new** ChartBuilder()  
 .chartType(StyleManager.ChartType.***Line***)  
 .width(1000).height(600).title(**"Polygon"**)  
 .xAxisTitle(**"Values"**).yAxisTitle(**"Count"**).build();  
  
 Histogram histogram = **new** Histogram(data, h, (**double**)min, (**double**)max);  
 chart.addSeries(**"Polygon"**, histogram.getxAxisData(), histogram.getyAxisData());  
  
 **return** chart;  
 }  
}

**public class** Main {  
 **private static final int *N*** = 18;  
 **private static final int *AMOUNT\_ITEMS*** = 100 \* ***N***;  
 **private static float**[] *array* = **new float**[***AMOUNT\_ITEMS***];  
  
 **public static void** main(String[] args) {  
 Calculate calculate = **new** Calculate();  
 Drawing drawing = **new** Drawing();  
  
 calculate.initializeArray(*array*, ***AMOUNT\_ITEMS***);  
 System.***out***.println(**"\nAmount groups: "** + calculate.*getH*(***AMOUNT\_ITEMS***));  
  
 System.***out***.format(**"Max element: %2.2f\n"**, Calculate.*getMaxElement*(*array*));  
 System.***out***.format(**"Min element: %2.2f\n"**, Calculate.*getMinElement*(*array*));  
 System.***out***.println(**"Floor step interval: "** + calculate.getStepInterval(*array*, ***AMOUNT\_ITEMS***));  
  
 **new** SwingWrapper(drawing.buildHistogram(calculate.getGaussianData(***AMOUNT\_ITEMS***, *array*), calculate.*getH*(***AMOUNT\_ITEMS***), Calculate.*getMaxElement*(*array*), Calculate.*getMinElement*(*array*))).displayChart();  
 **new** SwingWrapper(drawing.buildPolygon(calculate.getGaussianData(***AMOUNT\_ITEMS***, *array*), calculate.*getH*(***AMOUNT\_ITEMS***), Calculate.*getMaxElement*(*array*), Calculate.*getMinElement*(*array*))).displayChart();  
 }  
}

Результат виконання програми:



**Висновок:** в ході лабораторної роботи ми реалізували програму на мові Java яка будує полігон та гістограму.