Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления
Кафедра Интеллектуальных информационных технологий

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине «Проектирование программ в интеллектуальных системах»
на тему:

“РАЗРАБОТКА СОБСТВЕННОГО КОМПОНЕНТА ДЛЯ ПОСТРОЕНИЯ ГРАФИКОВ”

Выполнил: Студент группы 821702
Макаревич Д.А.

Проверил: Садовский М.Е.

Минск, 2020

Задание:

Разработать оконное приложение с одним главным окном.

Вариант 18

Приложение должно быть построено при помощи шаблона проектирования Model-View-Controller.

Компонент для рисования графиков должен обладать следующими возможностями:

1) Компонент должен поддерживать рисование нескольких графиков одновременно. Обновление графика происходит после вычисления каждой последующей точки. Например, вычислилось 1-е значение функции - появилось на графике, вычислилось второе значение функции - появилось на графике и т.д.

2) Компонент должен рисовать оси координат с подписями на стрелках

3) Подписями должны быть отмечены начало координат и деления на осях.

4) Если график не помещается на компонент, то должны появляться вертикальные и горизонтальные полосы прокрутки.

5) Компонент должен иметь две кнопки позволяющие увеличивать или уменьшать масштаб графика. Также масштаб можно изменить при помощи зажатой клавиши Ctrl и крутить колесо мыши на графике.

6) При зажатой на графике только левой клавише мыши и ее перетягивании должен происходить сдвиг отображаемой области графика.

7) Под графиком располагаться группа элементов, которая показывает текущий режим отображения графика. Единичный отрезок, текущий масштаб. Цветовая полоска с некоторым описанием функции.

Функции:

а) $f(x)=3x+1$

б) f(x) Таблица функции задается по следующему правилу: программой генерируются одномерные числовые массивы размером от 2 до n элементов. К каждому массиву применяется алгоритм сортировки с помощью прямого включения. В таблицу заносятся количество элементов и среднее время сортировки массива. Для получение среднего времени необходимо отсортировать k массивов одного размера. Шаг увеличения размера массива 1. Под шагом понимается, что сначала программа отсортирует k массивов размером 2, затем k массивов размером 3, и т.д.

Графическое отображение

Главное окно приложения:

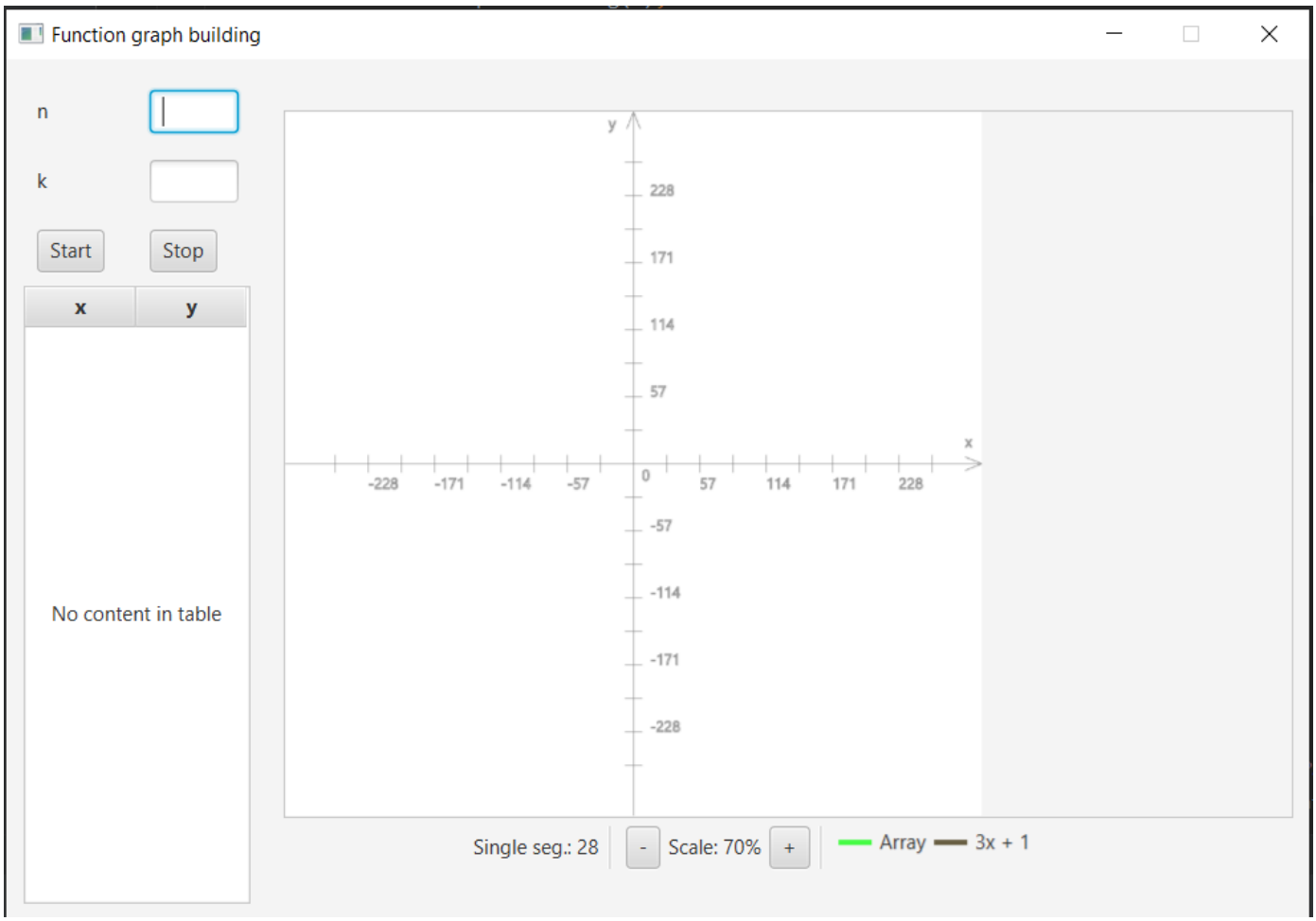

Рисунок 1 Окно приложения

Построенные графики:

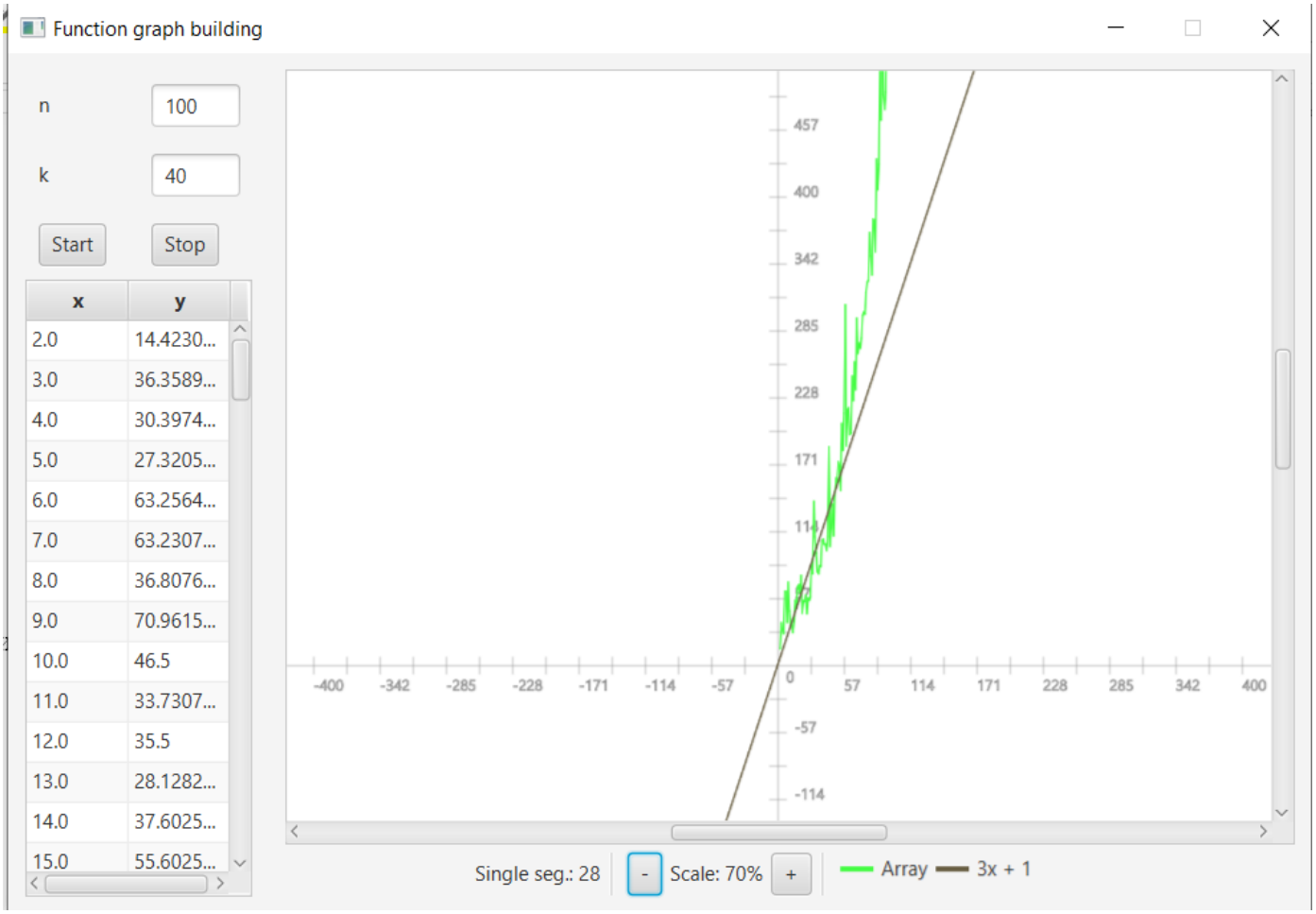

Рисунок 2 Графики

Описание классов

Данный класс содержит в себе список всех студентов надо которыми совершаются дальнейшие действия.

Класс Model:

```
public class Model {

    private List<Student> studentList;

    public Model(int studentsNumber) {
        if (studentsNumber > 0) {
            studentList = generateStudents(studentsNumber);
        } else {
            studentList = new ArrayList<>();
        }
    }

    private List<Student> generateStudents(int studentsNumber) {
        List<Student> students = new ArrayList<>();

        for (int i = 0; i < studentsNumber; i++) {
            students.add(new Student(new SNP(RandomizationData.reqSurname(), RandomizationData.reqName(), RandomizationData.reqPatronymic()),
                    RandomizationData.reqGroupNumber(),
                    RandomizationData.reqCourseNumber(),
                    RandomizationData.reqTotalWorksNumber(),
                    RandomizationData.reqFinishedWorksCounter(),
                    RandomizationData.reqComputerLanguage()
                   ));
        }
        return students;
    }

    public List<Student> getStudentList() {
        return studentList;
    }

    public void setStudentList(List<Student> studentList) {
        this.studentList = studentList;
    }

    public void addStudent(Student student) {
        studentList.add(student);
    }
}
```

Рисунок 3 Класс модели

Класс функции, хранящий имя функции, ее верхний и нижний пределы по оси X и список точек функции.

```java
package model;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;


public class Function {
    public static final double MIN_X_DOWN_LIMIT = -200;
    public static final double MAX_X_UP_LIMIT = 400;

    private String definition;

    private double xDownLimit;
    private double xUpLimit;

    private ObservableList<Point> points;


    public Function(String definition, double xDownLimit, double xUpLimit) {
        this.definition = definition;

        this.xDownLimit = xDownLimit;
        this.xUpLimit = xUpLimit;

        points = FXCollections.observableArrayList();
    }

    public Function() { this( definition: "", MIN_X_DOWN_LIMIT, MAX_X_UP_LIMIT); }

    public Function(String definition) { this(definition, MIN_X_DOWN_LIMIT, MAX_X_UP_LIMIT); }

    public String getDefinition() { return definition; }

    public double getXUpLimit() { return xUpLimit; }

    public void setXUpLimit(double xUpLimit) { this.xUpLimit = xUpLimit; }
```

Рисунок 4 Класс Funnction

Класс контроллера, который запускает три потока: 2 для рисования функций и один для вывода информации на экран:

```java
package controller;

import layout.GraphicCanvas;
import model.Function;

import java.util.HashMap;
import java.util.Map;


public class Controller {private Function arrayFunction;
    private Function linearFunction;
    private GraphicCanvas graphicCanvas;
    private Thread arrayFunCalcThread;
    private Thread linFunCalcThread;

    private Map<Function, Integer> functionOrders;
    private Integer functionOrder;

    private boolean isTCalcThreadsAlive;


    public Controller(Function arrayFunction, Function linearFunction, GraphicCanvas graphic) {
        this.arrayFunction = arrayFunction;
        this.linearFunction = linearFunction;
        this.graphicCanvas = graphic;
        arrayFunCalcThread = new Thread( name: "array-calc");
        linFunCalcThread = new Thread( name: "linear-calc");

        Thread drawThread = new Thread(new DrawingTask(graphic));
        drawThread.setName("draw");
        drawThread.setDaemon(true);
        drawThread.start();

        functionOrders = new HashMap<>();
        functionOrder = 0;
        functionOrders.put(arrayFunction, functionOrder++);
        functionOrders.put(linearFunction, functionOrder++);
```

Рисунок 5 Класс контроллера

Класс реализующий интерфейс Runnable, строит линейную функцию:

```java
package controller;

import layout.GraphicCanvas;
import model.Function;
import model.Point;


public class LinearFunctionCalcTask implements Runnable {
    private final Function linearFunction;
    private final int calcTaskNumber;
    private final GraphicCanvas graphicCanvas;


    public LinearFunctionCalcTask(Function linearFunction, int calcTaskNumber, GraphicCanvas graphicCanvas) {
        this.linearFunction = linearFunction;
        this.calcTaskNumber = calcTaskNumber;
        this.graphicCanvas = graphicCanvas;
    }

    @Override
    public void run() {
        double a = 3;
        double b = 1;
        double step = 1;
        int sleepTime = 30;

        for (double x = linearFunction.getXDownLimit(); x <= linearFunction.getXUpLimit(); x += step) {
            linearFunction.getPoints().add(new Point(x,  y: a*x + b));
            graphicCanvas.updateFunctionIterator(linearFunction);

            try {
                Thread.sleep(sleepTime);
```

Рисунок 6 Класс LinearFunctionCalcTask

Класс отображавший графические элементы приложения:

```
package layout;

import controller.Controller;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.geometry.Orientation;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import model.Function;
import model.Point;
import sample.Main;


public class GraphicBuildingComponent {
    private static final String SCALE_TEXT = "Scale: ";
    private static final String SINGLE_SEGMENT_TEXT = "Single seg.: ";
    private static final int FULL_PERCENTS = 100;

    private GridPane gridPane;

    private Label singleScaleSegment;
    private TextField nTextField;
    private TextField kTextField;
    private Button startBuildButton;
    private Button stopBuildButton;
    private TableView<Point> functionTable;
```

Рисунок 7 Класс GraphicBuildingComponent

Ход работы

Приложение параллельно высчитывает координаты для двух функций и отрисовывает их на графике. При нажатии кнопки “Stop” рисование заканчивается и потоки останавливаются.

Вывод

В данной лабораторной работе было реализовано оконное приложение с помощью библиотеки JavaFx без использование редактора форм. Приложение имеет главное окно в котором содержится таблица значений функции и построенный график. Приложение построено при помощи модели проектирования Model-View-Controller и полностью соответствует всем заявленным требованиям.