МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по учебной практике

Тема: «Визуализация пузырьковой сортировки»

Студент гр.8304 _____ Сани Заяд

Студент гр.8381 _____ Нгуен Ш. Х.

Преподаватель _____ Фирсов М. А.

Санкт-Петербург

2020

ЗАДАНИЕ
НА УЧЕБНУЮ ПРАКТИКУ

Студент Сани Заяд группы 8304

Студент Нгуен Ш. Х. группы 8381

Тема практики: «Визуализация пузырьковой сортировки»

Задание на практику:

Требуется визуализировать алгоритм пузырьковой сортировки с анимацией, представленной графиками. А также выполняемые команды отображаются и отображается сообщение, которое отображается при запуске алгоритма.

Сроки прохождения практики: 29.06.2020 – 13.07.2020

Дата сдачи отчета: 13.07.2020

Дата защиты отчета: 13.07.2020

Студент	_____	Сани Заяд
Студент	_____	Нгуен Ш. Х.
Руководитель	_____	Фирсов М.А.

АННОТАЦИЯ

Целью выполнения данной работы является практическое применения алгоритмы пузырьковой сортировки. Созданная программа выполняет визуализацию графика массива, также сообщение, которое отображается при запуске алгоритма, в то же время выполняемый код выделяется. Анимация алгоритма сортировки пузырьков выполняется в соответствии с заданным уведомлением. В результате мы получаем отсортированный график.

SUMMARY

The aim of this work is the practical application of bubble sorting algorithms. The created program performs visualization of the graph of the array, as well as the message that is displayed when the algorithm starts, while the executed code is highlighted. The animation of the bubble sorting algorithm is performed in accordance with the specified notification. As a result, we get a sorted chart.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

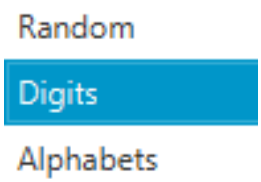
На вход программе подаётся график массива, сообщения и также поля управления алгоритма пузырьковой сортировки. Для запуска программы сначала пользователь должен выбрать, как загрузить входные данные. Затем выберите скорость программы, которую вы хотите и также количество элементов массива. Скорость работы алгоритма количество элементов массива и оцениваются по шкале значения от 1 до 10. Пользователь может запустить алгоритм с анимацией или запустить алгоритм шаг за шагом, тогда график будет обновляться при нажатии кнопки «Step Sort». Пользователи могут перемешать данные с помощью кнопки «Shuffle», данные не изменят значения, а только изменят положение между ними в массиве. Данные будут удалены, а алгоритм будет сброшен с помощью кнопки «Reset». Окончательный результат будет сохранен как изображение .png.

1. СПЕЦИФИКАЯ ПРОГРАММЫ

1.1 Исходные требования к программе

1.1.1 Требования к входным данным

Входные данные могут быть одного из следующих двух типов : «цифры» и «буквы». Если выбран «цифры», входные данные представляют собой строку чисел. Если выбран «буквы », входные данные представляют собой строку символов. Наоборот если «случайный» выбран, то тип данных и данные генерируются автоматически.



Перед запуском программы выбираются количество элементов входных данных и скорость алгоритма.

1.1.2 Требования к выходным данным

В результате график сохраняется в виде файла .png путем захвата экрана.

1.1.3 Требования к визуализации

Интерфейс состоит из 5 частей :

- Поле управление(нижний левый угол интерфейса), которое управляет способом ввода, количеством элементов и скоростью работы. М

- Поле кнопки (нижний справый угол интерфейса) : “Sort”, “Shuffle”, “Step Sort”, “Step”
- График (верхний левый угол интерфейса) представляет состояние данных массива в настоящее время
- Поле сообщения (верхний справый угол интерфейса): отображается сообщение, которое отображается при запуске алгоритма, в то же время выполняемый код выделяется.
- Строка меню “MenuBar” состоит из 2 частей: ‘File’ и ‘Help’:

  + Пукт ‘File’ включает в себя ‘Save File’ - Сохраните результат алгоритма и график результатов будет сохранен в виде файла .png, ‘Quit’ – выход программа.

  + Пукт ‘Help’ включает в себя ‘About Program’ - руководство по программе и ‘Authors’ - Информация об авторах

1.2 Уточнение требований после консультации с преподавателем

По итогу консультации с преподавателем были отмечены следующие замечания, учтённые в последующих версиях программы:

- Анимация добавлена с функцией запуска шаг за шагом
- В начальной версии формата выходых данных содержалось ресунок всей программы. Это было исправлено, чтобы сохранить только график результаты программы
- Изменить о деталях, чтобы объяснить более четко для пользователей

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

К 01.07.20 выбрать тему проекта, обсудить задание, распределить роли между членами команды и создать примерный план разработки.

К 05.07.20 выполнить первую итерацию, создав прототип интерфейса и отчет о проделанной работе.

К 07.07.20 Алгоритм пузырьковой сортировки установлен

К 09.07.20 Выполнить MenuBar содержат : Save – сохранить результат, About – Информация и инструкции по использованию программы, Author– Информация об авторах

К 11.07.20 произвести разбор и устранение недочетов, выявленных после сдачи второй итерации

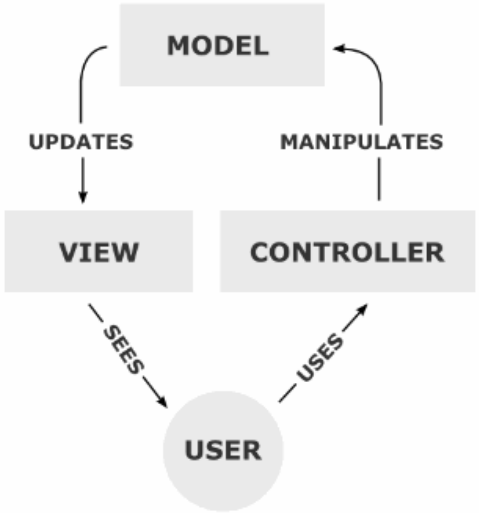
К 12.07.20 должно быть завершено анимация графика и сообщения и написать финальную версию отчета

2.2. Распределение ролей в бригаде

- Сани Заяд : лидер, алгоритмист, тестировщик, фронтенд
- Нгуен Хай : алгоритмист, тестировщик, фронтенд, документация

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

Концепция MVC позволяет разделить модель, представление и контроллер на три отдельных компонента:



- **Модель (Model)** предоставляет данные и реагирует на команды контроллера, изменяя своё состояние
- **Представление (View)** отвечает за отображение данных модели пользователю, реагируя на изменения модели
- **Контроллер (Controller)** интерпретирует действия пользователя, оповещая модель о необходимости изменений

3.1. Структуры Модель (Model)

Класс BarItem - базовый класс для создания графиков. Каждый объект BarItem соответствует столбцу на графике, который включает значения и состояния (ACTIVE, NOACTIVE, SORTED)

Класс CodeInfoModelItem - Представляет соответствующее значение (value) каждого BarItem

Модель даже содержит много классов enum, которые представляют состояние объектов в программе:

Таблица 1. Классы enum в модели

Enum class State{ACTIVE, NOACTIVE, SORTED}	Цветное состояние BarItem : ACTIVE – ORANGE, NOACTIVE - DARKGRAY, SORTED- DARKGREEN
Enum class Type {DIGITS, ALPHABETS, RANDOM}	Метод ввода : DIGITS - массив чисел, ALPHABETS - массив символов, RANDOM - случайный массив
Enum class Codebg{c, NOACTIVE}	Цветное состояние фона сообщений : ACTIVE – BLACK, NOACTIVE - WHITE
Num class CodeFill{ACTIVE, NOACTIVE}	Цветное состояние сообщений : ACTIVE – WHITE, NOACTIVE - BLACK

3.2 Структуры Контроллер (Controller)

Класс InputController - Выбранный пользователем тип данных даже для количества элементов и скорости работы алгоритма.

Методы:

- private fun digitsDropdown() - Получить входные данные(массива чисел)
- private fun alphabetDropDown() - Получить входные данные(массива символов)
- private fun randomDropDown() - Автоматически сгенерированные данные (могут быть числовыми массивами, могут быть символьными

массивами) на основе количества элементов, введенных пользователем

- fun changeType(typeName:String): Type? – изменить состояние ввода данных
- fun getInput() - Получить входные данные на основе состояние ввода данных

Класс MenuBarController содержит функции fun save() - Сохранить результаты (графики) в виде .png программой захвата экрана

Класс Processor - основные методы в программме

Методы:

- fun reset() - сброс алгоритма, а также графики
- fun suffle() – перемешать входные данные
- private fun codeStage() - Установить состояние фона и цвета сообщений
- private fun changeBarState(type:String, pos:Int = 0) – Изменить состояние BarItem
- fun stepSort() - запустить алгоритм шаг за шагом
- fun sort() - алгоритм пузырьковой сортировки
- fun reassignList(newList: SortedFilteredList<BarItem>) – Переприсвоить состояние BarItem
- private fun checkSorted():Boolean - Проверьте, отсортирован ли массив еще

3.3 Структуры представление (View)

3.3.1 Интерфейн

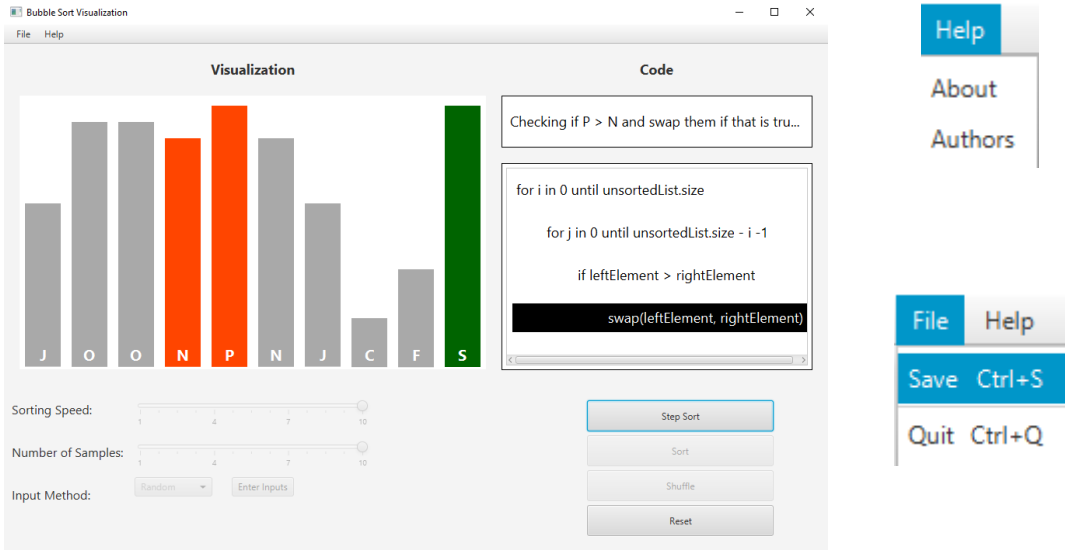


Рисунок 1. интерфейс программы

Интерфейс программы состоит из:

- Поле управление(нижний левый угол интерфейса), которое управляет способом ввода, количеством элементов и скоростью работы. Входные данные могут быть числами, буквами или случайными выбраны в пункте “Input Method”. Максимальное количество элементов входных данных составляет 10, выбранных в пункте “Numbers of Samples”. Скорость работы алгоритма оценивается по шкале от 1 до 10, который выбран в пункте “Sorting Speed”. Скорость работы алгоритма также пропорциональна скорости анимации и заданному сообщению.

- Поле кнопки (нижний справый угол интерфейса)

+ Кнопка «Sort» для запуска алгоритма пузырьковой сортировки
+ Кнопка «Shuffle» для перемешивания входных данных
+ Кнопка «Reset» для сброса алгоритма
+ Кнопка «Step Sort» для запуска алгоритма по шагами

- График (верхний левый угол интерфейса) представляет состояние данных массива в настоящее время. Каждый элемент массива представлен столбцом значения на графике. График столбца значения состоит из 3 разных цветов, соответствующих несортированному элементу, сортируемому элементу и отсортированному элементу.

- Поле сообщения (верхний справый угол интерфейса) : отображается сообщение, которое отображается при запуске алгоритма, в то же время выполняемый код выделяется.

- Строка меню “MenuBar” состоит из 2 частей: ‘File’ и ‘Help’: + Пукт ‘File’ включает в себя ‘Save File’ - Сохраните результат алгоритма и график результатов будет сохранен в виде файла .png, ‘Quit’ – выход программа.

  + Пукт ‘Help’ включает в себя ‘About Program’ - руководство по программе и ‘Authors’ - Информация об авторах

3.3.2 Классы для GUI

Интерфейс представляет собой набор классов, каждый из которых отображает соответствующую область интерфейса,

расположенную на разных макетах. Основным интерфейсом класса является MainView. Другие интерфейсные классы содержатся в этом классе.

Таблица 2. Классы интерфейн

Class MainView : View()	Основной класс содержит остальные классы, расположенные в макете borderpane
Class MenuBar : View()	MenuBar расположен в верхней части. Сохранить результаты, помощь и информацию об авторе
Class InputDialog() : View()	Диалоговое окно появляется, когда пользователь вводит входные данные
Class InfoView() : View()	График и сообщение
Class AuthorView() : View()	В окне отображается информация об авторе
Class AboutView() : View()	В окне отображается информация о программе

4. ТЕСТИРОВАНИЕ

ЗАКЛЮЧЕНИЕ

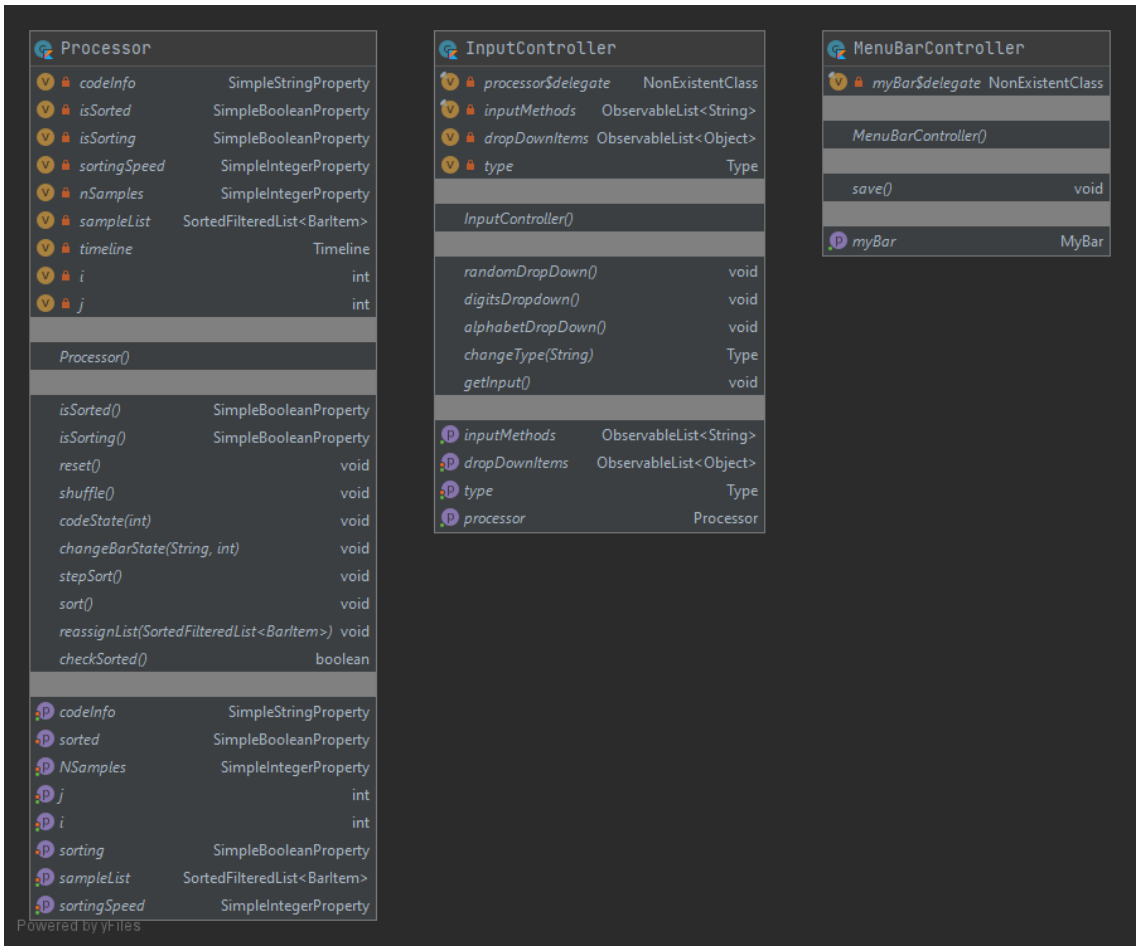
Проект был завершен успешно. Были выполнены поставленные цели, содержащиеся в техническом задании. Был реализован удобный пользовательский интерфейс, упрощающий взаимодействие с программой, а также наглядная визуализация процесса сортировки значений в массиве и анимация представления графа. Окончательный результат будет сохранен как изображение .png.

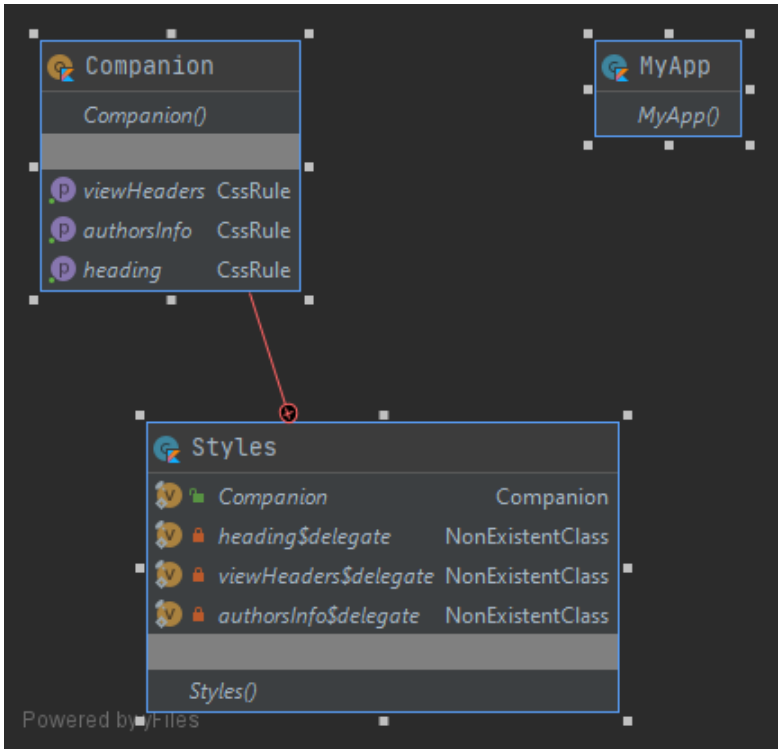
Работа программы основывается на алгоритме пузырьковой сортировки, сортирующий элементы в массиве по инкрементным значениям
Работоспособность алгоритма была проверена тестированием.

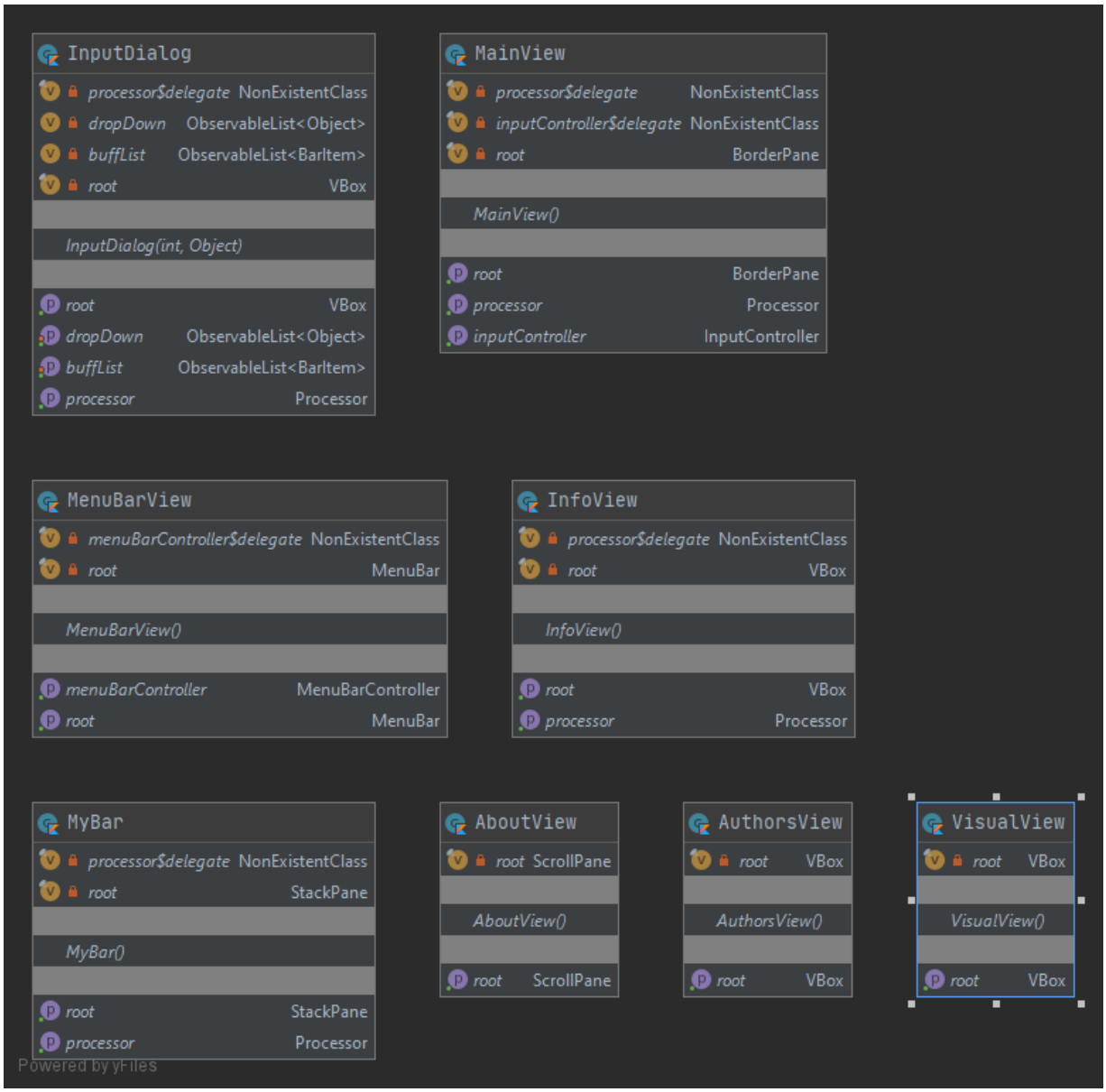
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

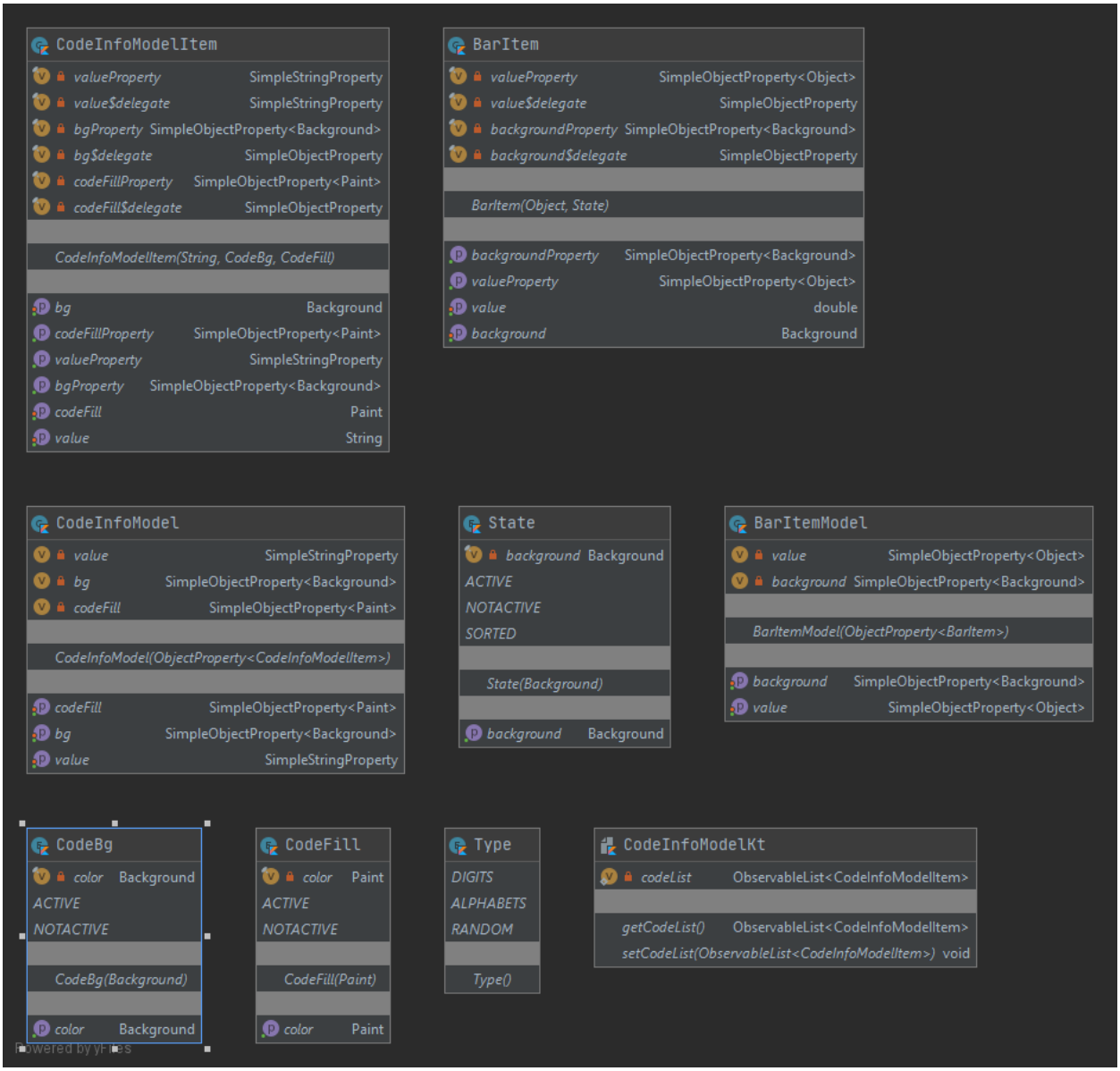
1. Официальная документация к Kotlin: https://kotlinlang.org/docs/tutorials/
2. Официальная документация к TornadorFx: https://edvin.gitbooks.io/tornadofx-guide/content/
3. https://ru.stackoverflow.com/
4. Википедия: https://ru.wikipedia.org
5. Github.com
6. Учебный курс по основам Kotlin на Stepik: https://stepik.org/course/5448/syllabus?auth=registration

ПРИЛОЖЕНИЕ А. UML-ДИАГРАММА






InputDialog
processor$delegate NonExistentClass
dropDown ObservableList<Object>
buffList ObservableList<BarItem>
root VBox
InputDialog(int, Object)
root VBox
dropDown ObservableList<Object>
buffList ObservableList<BarItem>
processor Processor
MainView
processor$delegate NonExistentClass
inputController$delegate NonExistentClass
root BorderPane
MainView()
root BorderPane
processor Processor
inputController InputController
MenuBarView
menuBarController$delegate NonExistentClass
root MenuBar
MenuBarView()
menuBarController MenuBarController
root MenuBar
InfoView
processor$delegate NonExistentClass
root VBox
InfoView()
root VBox
processor Processor
MyBar
processor$delegate NonExistentClass
root StackPane
MyBar()
root StackPane
processor Processor
AboutView
root ScrollPane
AboutView()
root ScrollPane
AuthorsView
root VBox
AuthorsView()
root VBox
VisualView
root VBox
VisualView()
root VBox
Powered by yFiles


CodeInfoModelItem
valueProperty SimpleStringProperty
value$delegate SimpleStringProperty
bgProperty SimpleObjectProperty<Background>
bg$delegate SimpleObjectProperty
codeFillProperty SimpleObjectProperty<Paint>
codeFill$delegate SimpleObjectProperty
CodeInfoModelItem(String, CodeBg, CodeFill)
bg Background
codeFillProperty SimpleObjectProperty<Paint>
valueProperty SimpleStringProperty
bgProperty SimpleObjectProperty<Background>
codeFill Paint
value String
BarItem
valueProperty SimpleObjectProperty<Object>
value$delegate SimpleObjectProperty
backgroundProperty SimpleObjectProperty<Background>
background$delegate SimpleObjectProperty
BarItem(Object, State)
backgroundProperty SimpleObjectProperty<Background>
valueProperty SimpleObjectProperty<Object>
value double
background Background
CodeInfoModel
value SimpleStringProperty
bg SimpleObjectProperty<Background>
codeFill SimpleObjectProperty<Paint>
CodeInfoModel(ObjectProperty<CodeInfoModelItem>)
codeFill SimpleObjectProperty<Paint>
bg SimpleObjectProperty<Background>
value SimpleStringProperty
State
background Background
ACTIVE
NOTACTIVE
SORTED
State(Background)
background Background
BarItemModel
value SimpleObjectProperty<Object>
background SimpleObjectProperty<Background>
BarItemModel(ObjectProperty<BarItem>)
background SimpleObjectProperty<Background>
value SimpleObjectProperty<Object>
CodeBg
color Background
ACTIVE
NOTACTIVE
CodeBg(Background)
color Background
CodeFill
color Paint
ACTIVE
NOTACTIVE
CodeFill(Paint)
color Paint
Type
DIGITS
ALPHABETS
RANDOM
Type()
CodeInfoModelKt
codeList ObservableList<CodeInfoModelItem>
getCodeList() ObservableList<CodeInfoModelItem>
setCodeList(ObservableList<CodeInfoModelItem>) void

ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл MyApp.kt

```
package com.example.demo.app
import com.example.demo.view.MainView
import tornadofx.App
class MyApp: App(MainView::class, Styles::class)
```

Файл Styles.kt

```
package com.example.demo.app
import javafx.geometry.Pos
import javafx.scene.paint.Color
import javafx.scene.text.FontWeight
import tornadofx.*

class Styles : Stylesheet() {
    companion object {
        val heading by cssclass()
        val viewHeaders by cssclass()
        val authorsInfo by cssclass()
    }

    init {
        label and heading {
            padding = box(10.px)
            fontSize = 20.px
            fontWeight = FontWeight.EXTRA_BOLD
            textFill = Color.RED
```

```
        }
        authorsInfo {
            padding = box(10.px)
            fontSize = 18.px

        }
        viewHeaders{
            fontSize = 18.px
            fontWeight = FontWeight.BOLD
            maxWidth = infinity
            alignment = Pos.CENTER
        }

    }
}
```

Файл InputController.kt

```
package com.example.demo.controller

import com.example.demo.model.BarItem
import com.example.demo.model.Type
import com.example.demo.view.InputDialog
import javafx.collections.FXCollections
import tornadofx.*
import java.util.*

class InputController :Controller(){
    val processor: Processor by inject()
```

```
    val inputMethods = FXCollections.observableArrayList("Random", "Digits",
"Alphabets")
    var dropDownItems  = FXCollections.observableArrayList<Any>();

    var type:Type? = Type.RANDOM
    private fun randomDropDown(){
        if (Random().nextInt(2) == 0){
            val buffList = FXCollections.observableArrayList<BarItem>()
            for (i in 0 until processor.nSamples.value){
                buffList.add(BarItem(Random().nextInt(20)))
            }
            processor.reassignList(SortedFilteredList<BarItem>(buffList))
        }
        else{
            val upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
            val buffList = FXCollections.observableArrayList<BarItem>()
            for (i in 0 until processor.nSamples.value){
                buffList.add(BarItem(upper[Random().nextInt(26)]))
            }
            processor.reassignList(SortedFilteredList<BarItem>(buffList))
        }
    }
    private fun digitsDropdown(){
        dropDownItems.clear()
        for (i in (1..20)){
            dropDownItems.add(i)
        }
    }
    private fun alphabetDropDown(){
```

```
        dropDownItems.clear()
        for (i in ('A'..'Z')){
            dropDownItems.add(i)
        }
    }
    fun changeType(typeName:String): Type?{
        when(typeName){
            "Random" -> return Type.RANDOM
            "Digits" -> return Type.DIGITS
            "Alphabets" -> return Type.ALPHABETS
            else -> print("errror type")
        }
        return null
    }
    fun getInput(){
        if(type == Type.RANDOM){
            randomDropDown()
        }else{
            when(type){
                Type.DIGITS -> digitsDropdown()
                Type.ALPHABETS -> alphabetDropDown()
                else -> print("errror type")
            }
            InputDialog(processor.nSamples.value, dropDownItems).openModal()
        }
    }
}
```

Файл MenuBarController.kt

```
package com.example.demo.controller

import com.example.demo.view.MainView
import com.example.demo.view.MyBar
import javafx.embed.swing.SwingFXUtils
import javafx.scene.image.Image
import javafx.stage.FileChooser
import tornadofx.*
import java.awt.Rectangle
import java.awt.Robot
import java.io.IOException
import javax.imageio.ImageIO

class MenuBarController :Controller(){
    val myBar: MyBar by inject()



    fun save(){
        val fileChooser = FileChooser()
        val extFilter = FileChooser.ExtensionFilter("(PNG)", "*.png")
        fileChooser.extensionFilters.add(extFilter)
        val file = fileChooser.showSaveDialog(primaryStage)

        val x = myBar.root.localToScreen(0.0, 0.0).x
        val y = myBar.root.localToScreen(0.0, 0.0).y
        val width  = myBar.root.boundsInParent.width
        val height  = myBar.root.boundsInParent.height
```

```
        val image =
Robot().createScreenCapture(Rectangle(x.toInt(),y.toInt(),width.toInt(),height.toInt
()))
        val convImage = SwingFXUtils.toFXImage(image,null)

        file?.let{
            try {
                ImageIO.write(SwingFXUtils.fromFXImage(convImage as Image?,
null), "png", file)
            } catch (e: IOException) {
            }
        }
    }
}
```

Файл Processor.kt

```
package com.example.demo.controller

import com.example.demo.model.*
import javafx.animation.KeyFrame
import javafx.animation.Timeline
import javafx.beans.property.SimpleBooleanProperty
import javafx.beans.property.SimpleIntegerProperty
import javafx.beans.property.SimpleStringProperty
import javafx.event.EventHandler
import javafx.util.Duration.seconds
import tornadofx.*
import java.lang.Exception
import java.util.*
```

```
class Processor : Controller(){
    var codeInfo = SimpleStringProperty("List is empty")
    var isSorted = SimpleBooleanProperty(true)
    var isSorting = SimpleBooleanProperty(false)
    var sortingSpeed = SimpleIntegerProperty(10)
    var nSamples = SimpleIntegerProperty(5)
    var sampleList = SortedFilteredList<BarItem>()
    private var timeline:Timeline = Timeline()


    fun reset(){
        timeline.stop()
        i = 0
        j = 0
        isSorting.value = false
        isSorted.value = false
        sortingSpeed.value = 10
        nSamples.value = 5
        codeState(-1)
        codeInfo.value = "list is empty"
        sampleList.clear()
    }
    fun shuffle(){
        val rgen = Random()
        for (i in sampleList.indices) {
            val randomPosition = rgen.nextInt(sampleList.size)
            sampleList.swap(i, randomPosition)
```

```
    }
    timeline.stop()
    i = 0
    j = 0
    if(!checkSorted()){
        isSorted.value = false
        codeInfo.value = "list is unsorted"
        sampleList.forEach {
            it.background = State.NOTACTIVE.background
        }
    }
}

private  fun codeState(index:Int){
    codeList.forEach {
        it.bg = CodeBg.NOTACTIVE.color
        it.codeFill = CodeFill.NOTACTIVE.color
    }
    try {
        codeList[index].bg = CodeBg.ACTIVE.color
        codeList[index].codeFill = CodeFill.ACTIVE.color
    }catch (e: Exception) { }
}
private fun changeBarState(type:String, pos:Int = 0){
    if(type == "active"){
        sampleList[j].background = State.ACTIVE.background
        sampleList[j + 1].background = State.ACTIVE.background
    }else if(type == "activeComplete"){
        try {
```

```
                sampleList[pos -1].background = State.NOTACTIVE.background
                sampleList[pos - 2].background = State.NOTACTIVE.background
            }
            catch (e: Exception) { }
        }else{
            try {
                sampleList[sampleList.size - i - 1].background =
State.SORTED.background
                sampleList[sampleList.size - i - 2].background =
State.NOTACTIVE.background
            }
            catch (e: Exception) { }finally {
                j = 0
                i++
            }
        }

    }
    var i = 0
    var j = 0

    fun stepSort() {
        val speed = (sortingSpeed.value.toDouble()+3)/100
        isSorting.value = true
        if (i < sampleList.size) {
            codeState(0)
            if(j < sampleList.size - i - 1){
                codeState(1)
                val jTemp = j
```

```
            val timeline = Timeline(KeyFrame(seconds(speed), EventHandler {
                codeInfo.value = "Checking if ${sampleList[j].value} >
${sampleList[j + 1].value} and swap them if that is true.."
                changeBarState("active")
                codeState(2)
                val a = sampleList[j].getValue()
                val b = sampleList[j+1].getValue()
                if (a > b) {
                    sampleList.swap(j, j + 1)
                    codeState(3)
                }
                j++
            }))
            timeline.play()
            timeline.setOnFinished {
                changeBarState("activeComplete", jTemp)
            }
        }else{
            changeBarState("sorted")
        }
    } else {
        i = 0
        j = 0
        isSorting.value = false
        isSorted.value = true
        codeInfo.value = "List is sorted"
        codeState(-1)
        println("finished")
    }
```

```
}
fun sort(){
    val speed = ((sortingSpeed.value.toDouble()+3)/100) * sampleList.size
    if (!isSorting.value){
        timeline = Timeline(
                KeyFrame(seconds(speed), EventHandler {
                    stepSort()
                }))
        val comparisons = ((sampleList.size + 1) * sampleList.size)/2
        timeline.cycleCount = comparisons + 1
        timeline.play()
    }

}

fun reassignList(newList: SortedFilteredList<BarItem>){
    sampleList.clear()
    sampleList.addAll(newList)
    if (checkSorted()){
        isSorted.value = true
        codeInfo.value = "list is sorted"
        sampleList.forEach {
            it.background =  State.SORTED.background
        }
    }else{
        isSorted.value = false
        codeInfo.value = "list is unsorted"
    }
```

```
    }
    private fun checkSorted():Boolean{
        for(i in  0 until sampleList.size-1){
            if (sampleList[i].getValue() > sampleList[i+1].getValue())
                return false
        }
        isSorted.value = true
        return true
    }
}
```

Файл BarItemModel.kt

```
package com.example.demo.model

import javafx.beans.property.ObjectProperty
import javafx.beans.property.SimpleObjectProperty
import javafx.collections.FXCollections
import javafx.geometry.Insets
import javafx.scene.layout.Background
import javafx.scene.layout.BackgroundFill
import javafx.scene.layout.CornerRadii
import javafx.scene.paint.Color
import tornadofx.*
import java.util.*
import tornadofx.getValue
import tornadofx.setValue

enum class State(val background: Background) {
```

```
    ACTIVE(Background(BackgroundFill(Color.ORANGERED,
CornerRadii.EMPTY, Insets.EMPTY))),
    NOTACTIVE(Background(BackgroundFill(Color.DARKGRAY,
CornerRadii.EMPTY, Insets.EMPTY))),
    SORTED(Background(BackgroundFill(Color.DARKGREEN,
CornerRadii.EMPTY, Insets.EMPTY)))
}

enum class Type{
    DIGITS,
    ALPHABETS,
    RANDOM
}

class BarItem(value: Any,state: State = State.NOTACTIVE) {
    val valueProperty = SimpleObjectProperty<Any>(value)
    var value by valueProperty

    val backgroundProperty =
SimpleObjectProperty<Background>(state.background)
    var background by backgroundProperty



    fun getValue():Double{
        if(value is Char)
            return (value as Char).toInt().toDouble() - 64.0
        else if(value is Int)
            return  (value as Int).toDouble()
        return 0.0
```

```
    }
}

class BarItemModel(property: ObjectProperty<BarItem>) :
ItemViewModel<BarItem>(itemProperty = property) {
    var value = bind(autocommit = true) { item?.valueProperty }
    var background = bind(autocommit = true) { item?.backgroundProperty }
}
```

Файл CodeInfoModel.kt

```
package com.example.demo.model

import javafx.beans.property.ObjectProperty
import javafx.beans.property.SimpleObjectProperty
import javafx.beans.property.SimpleStringProperty
import javafx.collections.FXCollections
import javafx.geometry.Insets
import javafx.scene.layout.Background
import javafx.scene.layout.BackgroundFill
import javafx.scene.layout.CornerRadii
import javafx.scene.paint.Color
import javafx.scene.paint.Paint
import tornadofx.*
import tornadofx.getValue
import tornadofx.setValue

enum class CodeBg(val color: Background) {
    ACTIVE(Background(BackgroundFill(Color.BLACK, CornerRadii.EMPTY,
Insets.EMPTY))),
```

```
    NOTACTIVE(Background(BackgroundFill(Color.WHITE,
CornerRadii.EMPTY, Insets.EMPTY))),
}

enum class CodeFill(val color: Paint) {
    ACTIVE(Color.WHITE),
    NOTACTIVE(Color.BLACK),
}

class CodeInfoModelItem(value: String,bg: CodeBg =
CodeBg.NOTACTIVE,codeFill: CodeFill = CodeFill.NOTACTIVE) {
    val valueProperty = SimpleStringProperty(value)
    var value by valueProperty

    val bgProperty = SimpleObjectProperty<Background>(bg.color)
    var bg by bgProperty

    val codeFillProperty = SimpleObjectProperty<Paint>(codeFill.color)
    var codeFill by codeFillProperty
}

class CodeInfoModel(property: ObjectProperty<CodeInfoModelItem>) :
ItemViewModel<CodeInfoModelItem>(itemProperty = property) {
    var value = bind(autocommit = true) { item?.valueProperty }
    var bg = bind(autocommit = true) { item?.bgProperty }
    var codeFill = bind(autocommit = true) { item?.codeFillProperty }
}

var codeList = FXCollections.observableArrayList<CodeInfoModelItem>(
```

```
        CodeInfoModelItem("for i in 0 until unsortedList.size"),
        CodeInfoModelItem("\tfor j in 0 until unsortedList.size - i -1"),
        CodeInfoModelItem("\t\tif leftElement > rightElement"),
        CodeInfoModelItem("\t\t\tswap(leftElement, rightElement)"))
```

Файл AboutView.kt

```
package com.example.demo.view

import javafx.scene.text.FontWeight
import tornadofx.*
import javax.swing.text.html.ImageView
import javafx.scene.image.Image
import javafx.scene.text.Font

class AboutView : View("About") {
    override val root = scrollpane {
        style{
            setMinSize(920.0,700.0)
            setMaxSize(1120.0,700.0)
        }

        borderpane {
            top = label("Visualization Bubble Sort"){
                style{
                    paddingLeft = 350
                    paddingTop = 20
                    fontSize = 20.px
                    fontWeight = FontWeight.EXTRA_BOLD
                }
```

```
        }

        center = vbox {
            style{
                padding = box(20.px)
                fontSize = 14.px
            }

            borderpane{
                top = label("1.Bubble Sort") {
                    style{
                        fontWeight = FontWeight.EXTRA_BOLD
                        paddingBottom = 10
                    }
                }
                center = hbox {
                    vbox{
                        label("\t•  Сортировка простыми обменами, сортиро́вка
пузырько́м (англ. bubble sort)\n " +
                                "-простой алгоритм сортировки. Для  понимания и
реализации этот алгоритм\n" +
                                "-простейший, но  эффективен он  лишь  для  небольших
массивов. Сложность \n" +
                                "алгоритма: O(n^2)\n\n") {
                            style{
                                paddingRight = 20
                            }
                        }
```

```
            label("\t•  Алгоритм состоит из повторяющихся проходов по
сортируемому массиву.\n" +
                  "За каждый проход элементы последовательно
сравниваются попарно и, если \n" +
                  "порядок в паре неверный, выполняется обмен
элементов. Проходы по массиву\n" +
                  "повторяются N-1 раз или до тех пор, пока на очередном
проходе не окажется,\n" +
                  "что обмены больше не нужны, что означает —
массив отсортирован. При\n" +
                  "каждом проходе алгоритма по внутреннему циклу,
очередной наибольший \n" +
                  "элемент массива ставится на своё место в конце
массива рядом с предыдущим\n" +
                  "« наибольшим элементом », а наименьший элемент
перемещается на одну \n" +
                  "позицию к началумассива ( «всплывает» до нужной
позиции, как пузырёк в \n" +
                  "воде — отсюда и название алгоритма)\n\n") {

        }
    }
    vbox {
        style{
            paddingRight = 10
        }
        label("\nПсевдокод"){
            style{
                fontWeight = FontWeight.LIGHT
```

```
            }
        }
        imageview(Image("bubbleSort.png"))
    }

}
}

borderpane{
    style{
        paddingTop = 10
    }

    top = label("2.Интерфейс"){
        style{
            fontWeight = FontWeight.EXTRA_BOLD
            paddingBottom = 10
        }
    }
    center = hbox{

        vbox{
            style{
                setMaxSize (535.0,1500.0)
            }

            label("\tИнтерфейс программы состоит из:\n"+
```

```
                "\n\t\t•  Поле управление(нижний левый угол
интерфейса), которое управляет\n" +
                "\t\tспособом ввода, количеством элементов и
скоростью работы. Входные\n" +
                "\t\tданные  могут  быть  числами, буквами  или
случайными  выбраны  в\n" +
                "\t\tпункте “Input Method”. Максимальное количество
элементов входных\n" +
                "\t\tданных составляет 10, выбранных в пункте
“Numbers of  Samples”.\n" +
                "\t\tСкорость работы алгоритма оценивается по шкале
от 1 до 10, который\n" +
                "\t\tвыбран в пункте “Sorting Speed”. Скорость работы
алгоритма также\n " +
                "\t\tпропорциональна скорости анимации и заданному
сообщению.\n\n"
            ){


            }
            label("\n\t\t•  Поле кнопки (нижний справый угол
интерфейса)\n" +
                "\t\t\t+ Кнопка «Sort»   для запуска алгоритма
пузырьковой сортировки\n" +
                "\t\t\t+ Кнопка «Shuffle» для перемешивания входных
данных\n" +
                "\t\t\t+ Кнопка «Reset» для сброса алгоритма\n\n"){
            }
            label("\n\t\t•  График ( верхний левый угол интерфейса )
представляет состояние\n" +
```

```
                "\t\tданных массива в настоящее время . Каждый
элемент  массива \n" +
                "\t\tпредставлен столбцом значения на графике.
График столбца значения\n" +
                "\t\tсостоит из 3 разных цветов, соответствующих
несортированному \n" +
                "\t\tэлементу, сортируемому элементу и
отсортированному элементу\n\n"){
            }
            label("\n\t\t•  Поле сообщения (верхний справый угол
интерфейса):отображается\n" +
                "\t\tсообщение, которое отображается при запуске
алгоритма, в то же\n" +
                "\t\tвремя выполняемый код выделяется\n"){

            }
            label("\n\t\t•  Строка меню “MenuBar” состоит из 2 частей:
'File' и 'Help':\n" +
                "\t\t\t\tПукт ‘File’ включает в себя  ‘Save File’ -
Сохраните результат\n" +
                "\t\t\talгоритма и график результатов будет сохранен в
виде файла .png,\n" +
                "\t\t\t‘Quit’ – выход программа.\n" +
                "\t\t\t\tПукт ‘Help’ включает в себя ‘About Program’ -
руководство по\n" +
                "\t\t\tпрограмме и ‘Authors’ - Информация об
авторах\n\n"
            ){
```

```
        }
    }
vbox{
    style{
        paddingTop = 150
    }
        imageview(Image("Control2.png")){
            style{
                paddingLeft = 70
            }
        }
        imageview(Image("Buttons2.png")){
            style{
                paddingTop = 230
                paddingLeft = 70
            }
        }
        imageview(Image("Chart2.png")){
            style{
                paddingTop = 100
                paddingLeft = 70
            }
        }
        imageview(Image("Message2.png")){
            style{
                paddingTop = 100
                paddingLeft = 70
            }
        }
```

```
            }
        }
    }

    borderpane{
        style{
            paddingTop = 20
        }
        top = label("3. Описание работы программы"){
            style{
                fontWeight = FontWeight.EXTRA_BOLD
                paddingBottom = 10
            }
        }
        center = vbox{
            label("\n\t\t• Входные данные: Пользователь вводит данные через кнопку «Enter Inputs». Вы можете выбрать один из следующих типов:\n" +
                    " цифры, буквы  или случайный. Если выбран «цифры», входные данные представляют собой строку чисел. Если выбран «буквы », входные\n" +
                    " данные представляют собой строку символов. Наоборот если «случайный» выбран, то тип данных и данные генерируются автоматически.\n"){

            }
```

```
                label("\n\t\t• Выходные  данные : Если  входные  данные
являются  числовыми, то выходные данные  представляют собой массив
чисел,\n" +
                "расположенных в порядке возрастания, а если входные
данные представляют собой буквы, выходные данные представляют собой
массив\n" +
                "символов, отсортированных в порядке возрастания ASCII
их значения\n"){

                }

                label("\n\t\t• Шаги  для  выполнения  программы : Выберите,
как  загрузить  входные  данные. Затем выберите скорость  программы,
которую\n" +
                "вы  хотите.  Нажмите  кнопку  “Sort” , чтобы  запустить
программу.  Если  вы  хотите  сбросить  алгоритм , нажмите  кнопку  “Reset”,
после\n" +
                "чего  исходные  данные  будут  восстановлены  и  снова
запустите программу и наоборот, если вы  хотите изменить положение
элементов\n" +
                "(постоянные значения), нажмите «Перемешать»\n"){

                }
            }
        }
    }
  }
}
```

Файл AuthorsView.kt

```
package com.example.demo.view

import com.example.demo.app.Styles
import tornadofx.*

class AuthorsView : View("Authors") {

    override val root = vbox (){
        style{
            padding = box(20.px)
        }

        label("Состав команды:"){
            addClass(Styles.heading)
        }
        label("Сани Заяд, группа 8304"){
            addClass(Styles.authorsInfo)
        }
        label("Нгуен Ши Хай, группа 8381"){
            addClass(Styles.authorsInfo)
        }
        label("Распределение ролей:"){
            addClass(Styles.heading)
        }
        label("Сани Заяд : лидер, алгоритмист, тестировщик, фронтенд:"){
            addClass(Styles.authorsInfo)
        }
```

```
        label("Нгуен Хай : алгоритмист, тестировщик, фронтенд,
документация"){
            addClass(Styles.authorsInfo)
        }
    }
}
```

Файл BarGraphView.kt

```
package com.example.demo.view

import com.example.demo.controller.Processor
import javafx.geometry.Orientation
import javafx.geometry.Pos
import javafx.scene.text.FontWeight
import tornadofx.*

class MyBar : View(){
    val processor: Processor by inject()

    override val root =  stackpane {

        setPrefSize(600.0, 400.0)
        listview(processor.sampleList) {
            style{
                backgroundColor += c("#ffffff")
                alignment = Pos.BOTTOM_CENTER
                paddingTop = 10.0
            }
            isEditable = true
```

```
        orientation = Orientation.HORIZONTAL

        cellFormat{

            prefWidth = 600.0 / processor.sampleList.size
            style{
                alignment = Pos.BOTTOM_CENTER
                backgroundColor += c("#FFFFFF")
            }
            graphic = vbox {
                var maxValue = processor.sampleList.maxBy { barItem ->
barItem.getValue() }?.getValue()
                maxHeight = (it.getValue() / maxValue!!) * 400
                style {
                    alignment = Pos.BOTTOM_CENTER

backgroundProperty().bindBidirectional((item.backgroundProperty))
                }
                label ("${item.value}"){
                    style{
                        textFill = c("#FFFFFF")
                        fontSize = 20.px
                        fontWeight = FontWeight.EXTRA_BOLD
                    }
                }
            }
        }
    }
```

```
        label ("List is empty!"){
            style{
                fontSize = 50.px
            }
            visibleWhen { booleanBinding(processor.sampleList) { isEmpty() } }
        }
    }

}
```

Файл InfoView.kt

```
package com.example.demo.view

import com.example.demo.controller.MenuBarController
import com.example.demo.controller.Processor
import com.example.demo.model.codeList
import javafx.geometry.Pos
import javafx.scene.text.FontWeight
import tornadofx.*

class InfoView : View(){
    val processor: Processor by inject()



    override val root = vbox(20.0) {
        vbox {
            setPrefSize(Double.MAX_VALUE, 200.0)
            alignment = Pos.CENTER
            style {
                paddingLeft = 10.0
```

```
        paddingRight = 10.0
        backgroundColor += c("#FFFFFF")
        borderColor += box(c("#000000"))
    }

    label{
        textProperty().bind(processor.codeInfo)
        style{
            fontSize = 18.px
            maxWidth = infinity
            textFill = c("#000000")
        }
    }
}
vbox(10){
    alignment = Pos.CENTER
    setPrefSize(Double.MAX_VALUE, 400.0)
    style {
        padding = box(5.px)
        backgroundColor += c("#FFFFFF")
        borderColor += box(c("#000000"))
    }
    listview (codeList){
        style{
            alignment = Pos.CENTER
        }
        cellFormat{
            prefHeight = 55.0
            style{
```

```
                backgroundColor += c("#FFFFFF")
                alignment = Pos.CENTER
            }
            graphic =label(item.value){
                style{
                    fontSize = 18.px
                    maxWidth = infinity
                    padding = box(5.px)
                    backgroundProperty().bindBidirectional(item.bgProperty)
                    textFillProperty().bindBidirectional(item.codeFillProperty)
                }
            }
        }
    }

  }
}
```

Файл InputDialog.kt

```
package com.example.demo.view
import com.example.demo.controller.Processor
import com.example.demo.model.BarItem
import javafx.collections.FXCollections
import javafx.collections.ObservableList
import javafx.geometry.Pos
import tornadofx.*

class InputDialog(nSamples: Int, dropDownItems: Any) : View("Input Dialog") {
```

```
val processor: Processor by inject()
var dropDown =  dropDownItems as ObservableList<Any>?
var buffList = FXCollections.observableArrayList<BarItem>();

override val root = vbox (25){
    style{
        padding = box(20.px)
    }
    label("Enter your inputs : ") {

    }
    hbox(10) {
        for (i in 0 until nSamples){
            if (dropDown?.get(0) is Int)
                buffList.add(BarItem(1))
            else
                buffList.add(BarItem('A'))
            combobox<Any> {
                items = dropDown
                selectionModel.selectFirst()
            }.valueProperty().addListener { observable, oldValue, newValue
                -> if(newValue != null) buffList[i] = BarItem(newValue)
            }
        }

    }

    hbox(15) {
        alignment = Pos.BASELINE_RIGHT
```

```
            button("Cancel") {
                action {
                    super.close()
                }
                setPrefSize(100.0, 40.0)
            }
            button("Ok") {
                action {
                    processor.reassignList(SortedFilteredList<BarItem>(buffList))
                    buffList.clear()
                    super.close()
                }
                setPrefSize(100.0, 40.0)
            }
        }
    }
}
```

Файл MainView.kt

```
package com.example.demo.view

import com.example.demo.controller.InputController
import com.example.demo.controller.Processor
import javafx.geometry.Pos
import javafx.scene.control.ComboBox
import javafx.scene.layout.BorderPane
import tornadofx.*
```

```
class MainView : View("Bubble Sort Visualization") {
    val processor: Processor by inject()
    val inputController: InputController by inject()
    override val root = BorderPane()



    init {
        val dropdown = ComboBox<String>()
        with(root) {
            top (MenuBarView::class)
            center(VisualView::class)

            bottom = form {
                borderpane {
                    style {
                        paddingTop = 10
                        paddingBottom = 10
                    }
                    left = hbox(10.0) {
                        vbox (30){
                            label("Sorting Speed: ") {
                                style {
                                    fontSize = 16.px
                                }
                            }
                            label("Number of Samples: ") {
                                style {
                                    fontSize = 16.px
                                }
```

```
    }
    label("Input Method: ") {
        style {
            fontSize = 16.px
        }
    }

}
vbox {
    setPrefSize(300.0, Double.MIN_VALUE)
    style{
        paddingBottom = 10.0
    }
    slider(1.0, 10.0) {
        isShowTickLabels = true
        isShowTickMarks = true
        majorTickUnit = 3.0
        valueProperty().bindBidirectional(processor.sortingSpeed)
        disableProperty().bind(processor.isSorting)
        style {
            paddingBottom = 15
        }
    }
    slider(1.0, 10.0) {
        isShowTickLabels = true
        isShowTickMarks = true
        majorTickUnit = 3.0
        valueProperty().bindBidirectional(processor.nSamples)
        disableProperty().bind(processor.isSorting)
```

```
                }
                hbox(25) {
                    style {
                        paddingTop = 8
                    }
                    dropdown.items = inputController.inputMethods
                    dropdown.selectionModel.selectFirst()
                    disableProperty().bind(processor.isSorting)
                    dropdown.valueProperty().addListener { observable, oldValue,
newValue
                        -> inputController.type =
inputController.changeType(newValue)}
                    borderpane {
                        center = dropdown
                    }
                    button("Enter Inputs") {
                        action {
                            inputController.getInput()
                        }
                    }
                }
            }
        }
        center = vbox {
            spacing = 5.0
            alignment = Pos.CENTER_RIGHT
            disableWhen{booleanBinding(processor.sampleList) { isEmpty() }}
            style{
                paddingRight = 60.0
```

```
}
button("Step Sort") {
    action {
        processor.stepSort()
    }
    disableProperty().bind(processor.isSorted)
    setPrefSize(240.0, 40.0)
}
vbox {
    style{
        alignment = Pos.CENTER_RIGHT
    }
    disableProperty().bind(processor.isSorting)
    button("Sort") {
        action {
            processor.sort()
        }
        disableProperty().bind(processor.isSorted)
        setPrefSize(240.0, 40.0)
    }
}

button("Shuffle") {
    action {
        processor.shuffle()
    }
    disableProperty().bind(processor.isSorting)
    setPrefSize(240.0, 40.0)
}
```

```
                button("Reset") {
                    action {
                        dropdown.selectionModel.selectFirst()
                        processor.reset()
                    }
                    setPrefSize(240.0, 40.0)
                }

            }
        }

    }
  }
}
```

Файл ManuBarView.kt

```
package com.example.demo.view
import com.example.demo.controller.MenuBarController
import tornadofx.*

class MenuBarView : View(){
   val menuBarController: MenuBarController by inject()

   override val root =  menubar {
      menu("File") {
         item("Save", "Shortcut+S").action {
            menuBarController.save()
         }
```

```
            separator()
            item("Quit", "Shortcut+Q").action {
                super.close()
            }
        }
        menu("Help") {
            item("About").action {
                AboutView().openModal()
            }
            item("Authors").action {
                AuthorsView().openModal()
            }
        }
    }
}
```

Файл VisualView.kt

```
package com.example.demo.view

import com.example.demo.app.Styles
import tornadofx.*

class VisualView : View(){

    override val root = vbox {
        style{
            padding = box(20.px)
        }
        hbox (20.0){
```

```
            setMaxSize(Double.MAX_VALUE, 400.0)
            vbox (20.0){
                label("Visualization") {
                    addClass(Styles.viewHeaders)
                }
                borderpane {
                    center(MyBar::class)
                }
            }
            vbox (20.0){
                setMaxSize(400.0, Double.MAX_VALUE)
                label("Code") {
                    addClass(Styles.viewHeaders)
                }
                borderpane {
                    center(InfoView::class)
                }

            }
        }

    }
}
```

ПРИЛОЖЕНИЕ В. UNIT-ТЕСТИРОВАНИЕ