Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Отчёт

по лабораторной работе №4

на тему:

**ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ (ООП)**

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф.В. Усенко

(подпись)

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. Н. Кубицкая

(подпись) гр. 214302

Минск, 2024

**Задание:** Разработайте систему для создания интерактивных графиков данных, включающую классы Chart, LineChart, BarChart, и PieChart. Реализуйте методы для динамического обновления данных, настройки параметров отображения и взаимодействия с пользователем (например, масштабирование, фильтрация данных).

**Решение:**

import org.knowm.xchart.\*  
import org.knowm.xchart.style.Styler  
import org.knowm.xchart.SwingWrapper  
import javax.swing.JFrame  
  
abstract class Chart {  
 abstract fun show()  
}  
  
// Класс для линейного графика  
class LineChart(private val title: String, private val xAxisTitle: String, private val yAxisTitle: String) : Chart() {  
 private val chart: XYChart = XYChart(800, 600).apply {  
 this.title = this@LineChart.title  
 this.xAxisTitle = this@LineChart.xAxisTitle  
 this.yAxisTitle = this@LineChart.yAxisTitle  
 }  
  
 private var swingWrapper: SwingWrapper<XYChart>? = null // Сохраняем SwingWrapper для обновления окна  
 private var chartFrame: JFrame? = null // Для хранения открытого окна  
  
 private val seriesData = mutableMapOf<String, Pair<DoubleArray, DoubleArray>>() // Хранение исходных данных  
  
 // Метод для добавления данных в график  
 fun addSeries(seriesName: String, xData: DoubleArray, yData: DoubleArray) {  
 seriesData[seriesName] = Pair(xData, yData)  
 chart.addSeries(seriesName, xData, yData)  
 }  
  
 // Метод для обновления данных  
 fun updateSeries(seriesName: String, xData: DoubleArray, yData: DoubleArray) {  
 chart.updateXYSeries(seriesName, xData, yData, null)  
 repaintChart() // Перерисовываем график после обновления данных  
 }  
  
 // Метод для фильтрации данных  
 fun filterData(minY: Double, maxY: Double) {  
 // Удаляем все серии вручную  
 val seriesNames = chart.seriesMap.keys.toList() // Получаем имена всех серий  
 for (seriesName in seriesNames) {  
 chart.removeSeries(seriesName) // Удаляем серию по имени  
 }  
  
 // Применяем фильтрацию к данным  
 for ((seriesName, data) in seriesData) {  
 val (xData, yData) = data // Извлекаем x и y данные для серии  
 val filteredYData = yData.filter { it in minY..maxY }.toDoubleArray() // Фильтруем y данные  
 val filteredXData = xData.take(filteredYData.size).toDoubleArray() // Соответствующее количество x данных  
  
 if (filteredYData.isNotEmpty()) {  
 chart.addSeries(seriesName, filteredXData, filteredYData) // Добавляем обновленную серию  
 }  
 }  
 repaintChart() // Перерисовываем график после фильтрации  
 }  
  
 // Метод для изменения диапазона осей  
 fun setAxisRange(xMin: Double, xMax: Double, yMin: Double, yMax: Double) {  
 chart.styler.xAxisMin = xMin  
 chart.styler.xAxisMax = xMax  
 chart.styler.yAxisMin = yMin  
 chart.styler.yAxisMax = yMax  
 repaintChart() // Перерисовываем график после изменения диапазона  
 }  
  
 // Метод для обновления графика без открытия нового окна  
 private fun repaintChart() {  
 if (chartFrame != null) {  
 chartFrame?.repaint() // Перерисовываем уже существующее окно  
 }  
 }  
  
 // Переопределяем метод show для отображения графика  
 override fun show() {  
 if (chartFrame == null) {  
 // Если окно еще не создано, создаем его  
 swingWrapper = SwingWrapper(chart)  
 chartFrame = swingWrapper?.displayChart() // Отображаем окно с графиком  
 } else {  
 repaintChart() // Перерисовываем график, если окно уже было открыто  
 }  
 }  
}  
  
// Класс для столбчатого графика  
class BarChart(private val title: String, private val xAxisTitle: String, private val yAxisTitle: String) : Chart() {  
 private val chart: CategoryChart = CategoryChartBuilder()  
 .width(800)  
 .height(600)  
 .title(this.title)  
 .xAxisTitle(this.xAxisTitle)  
 .yAxisTitle(this.yAxisTitle)  
 .build()  
  
 private var swingWrapper: SwingWrapper<CategoryChart>? = null  
 private var chartFrame: JFrame? = null  
 private val seriesData = mutableMapOf<String, Pair<List<String>, List<Double>>>() // Хранение исходных данных  
  
 // Метод для добавления данных в столбчатый график  
 fun addSeries(seriesName: String, xData: List<String>, yData: List<Double>) {  
 seriesData[seriesName] = Pair(xData, yData)  
 chart.addSeries(seriesName, xData, yData)  
 }  
  
 // Метод для обновления данных  
 fun updateSeries(seriesName: String, xData: List<String>, yData: List<Double>) {  
 chart.updateCategorySeries(seriesName, xData, yData, null)  
 repaintChart() // Обновляем график без открытия нового окна  
 }  
  
 // Метод для изменения диапазона осей  
 fun setAxisRange(yMin: Double, yMax: Double) {  
 chart.styler.yAxisMin = yMin  
 chart.styler.yAxisMax = yMax  
 repaintChart()  
 }  
  
 // Метод для фильтрации данных по Y  
 fun filterData(minY: Double, maxY: Double) {  
 chart.seriesMap.clear() // Удаляем все серии  
  
 for ((seriesName, data) in seriesData) {  
 val (xData, yData) = data  
 val filteredYData = yData.filter { it in minY..maxY } // Фильтруем y-данные  
 val filteredXData = xData.take(filteredYData.size) // Соответствующее количество x-данных  
  
 if (filteredYData.isNotEmpty()) {  
 chart.addSeries(seriesName, filteredXData, filteredYData) // Добавляем обновленную серию  
 }  
 }  
 repaintChart()  
 }  
  
 // Метод для перерисовки графика  
 private fun repaintChart() {  
 if (chartFrame != null) {  
 chartFrame?.repaint() // Обновляем график  
 }  
 }  
  
 // Переопределяем метод show для отображения графика  
 override fun show() {  
 if (chartFrame == null) {  
 swingWrapper = SwingWrapper(chart)  
 chartFrame = swingWrapper?.displayChart() // Отображаем окно с графиком  
 } else {  
 repaintChart() // Перерисовываем график  
 }  
 }  
}  
  
  
// Класс для круговой диаграммы  
class CustomPieChart(private val title: String) : Chart() {  
 private val chart: PieChart = PieChartBuilder()  
 .width(800)  
 .height(600)  
 .title(this.title)  
 .build()  
  
 private val categories = mutableMapOf<String, Double>() // Хранение данных категорий  
 private var swingWrapper: SwingWrapper<PieChart>? = null  
  
 // Метод для добавления или обновления данных в круговой диаграмме  
 fun addOrUpdateData(seriesName: String, value: Double) {  
 categories[seriesName] = value  
 chart.removeSeries(seriesName) // Удаляем старую серию, если она существует  
 chart.addSeries(seriesName, value) // Добавляем новую серию  
 }  
  
 // Метод для обновления данных по всем категориям  
 fun updateAllData() {  
 // Создаем копию имен серий, чтобы избежать ConcurrentModificationException  
 val seriesNames = chart.seriesMap.keys.toList() // Копируем имена всех серий  
 for (name in seriesNames) {  
 chart.removeSeries(name) // Удаляем старые серии  
 }  
  
 // Добавляем обновленные серии  
 for ((name, value) in categories) {  
 chart.addSeries(name, value)  
 }  
 show() // Отображаем диаграмму после обновления  
 }  
  
 // Метод для фильтрации данных  
 fun filterData(threshold: Double) {  
 // Создаем копию имен серий, чтобы избежать ConcurrentModificationException  
 val seriesNames = chart.seriesMap.keys.toList() // Копируем имена всех серий  
 for (name in seriesNames) {  
 chart.removeSeries(name) // Удаляем старые серии  
 }  
  
 // Затем добавим только те, которые удовлетворяют условию  
 for ((name, value) in categories) {  
 if (value >= threshold) {  
 chart.addSeries(name, value)  
 }  
 }  
 show() // Обновляем отображение диаграммы  
 }  
  
 // Переопределяем метод show для отображения графика  
 override fun show() {  
 if (swingWrapper == null) {  
 swingWrapper = SwingWrapper(chart)  
 swingWrapper?.displayChart() // Отображаем окно с графиком  
 } else {  
 swingWrapper?.repaintChart() // Обновляем график  
 }  
 }  
}  
  
// Функция для отображения графиков  
fun displayCharts(lineChart: LineChart, barChart: BarChart, pieChart: CustomPieChart) {  
 // Обновляем данные в графике и показываем его  
 val xDataL = doubleArrayOf(1.0, 2.0, 3.0, 4.0, 5.0)  
 val yDataL = doubleArrayOf(2.0, 3.5, 5.0, 4.5, 6.0)  
 lineChart.addSeries("LineChart", xDataL, yDataL)  
 lineChart.show() // Показать график в одном окне  
  
 val xDataB = listOf("A", "B", "C", "D", "E")  
 val yDataB = listOf(3.0, 7.5, 4.5, 6.0, 8.0)  
 barChart.addSeries("BarChart", xDataB, yDataB)  
 barChart.show() // Показать график в одном окне  
  
 pieChart.addOrUpdateData("Категория A", 30.0)  
 pieChart.addOrUpdateData("Категория B", 20.0)  
 pieChart.addOrUpdateData("Категория C", 25.0)  
 pieChart.addOrUpdateData("Категория D", 25.0)  
 pieChart.show()  
}  
  
// Функция для динамического изменения данных  
fun dynamicUpdate(lineChart: LineChart, barChart: BarChart, pieChart: CustomPieChart) {  
 val xDataL = doubleArrayOf(1.0, 2.0, 3.0, 4.0, 5.0)  
 val xDataB = listOf("A", "B", "C", "D", "E")  
  
 while (true) {  
 println("Выберите график для изменения:")  
 println("1 - Линейный график")  
 println("2 - Столбчатый график")  
 println("3 - Круговая диаграмма")  
 println("4 - Фильтрация данных")  
 println("5 - Изменение параметров отображения")  
 println("0 - Вернуться в главное меню")  
  
 when (readLine()) {  
 "1" -> {  
 println("Введите новые данные для линейного графика (5 значений через запятую):")  
 val newYData = readLine()?.split(",")?.map { it.trim().toDouble() }?.toDoubleArray()  
 if (newYData != null && newYData.size == 5) {  
 lineChart.updateSeries("LineChart", xDataL, newYData)  
 } else {  
 println("Ошибка: Пожалуйста, введите ровно 5 значений.")  
 }  
 }  
 "2" -> {  
 println("Введите новые данные для столбчатого графика (5 значений через запятую):")  
 val newYData = readLine()?.split(",")?.map { it.trim().toDouble() }  
 if (newYData != null && newYData.size == 5) {  
 barChart.updateSeries("BarChart", xDataB, newYData) // Обновляем данные графика  
 } else {  
 println("Ошибка: Пожалуйста, введите ровно 5 значений.")  
 }  
 }  
 "3" -> {  
 // Запрос новых данных для всех категорий  
 val categories = listOf("Категория A", "Категория B", "Категория C", "Категория D")  
 for (category in categories) {  
 println("Введите новое значение для $category:")  
 val newValue = readLine()?.toDoubleOrNull()  
 if (newValue != null) {  
 pieChart.addOrUpdateData(category, newValue) // Обновляем диаграмму с новыми данными  
 } else {  
 println("Ошибка: Пожалуйста, введите корректное значение.")  
 }  
 }  
 pieChart.updateAllData() // Обновляем все данные в диаграмме  
 }  
 "4" -> {  
 println("Выберите график для фильтрации:")  
 println("1 - Линейный график")  
 println("2 - Круговая диаграмма")  
  
 when (readLine()) {  
 "1" -> {  
 println("Введите минимальное и максимальное значение Y (через запятую):")  
 val range = readLine()?.split(",")?.map { it.trim().toDouble() }  
 if (range != null && range.size == 2) {  
 lineChart.filterData(range[0], range[1])  
 } else {  
 println("Ошибка: Пожалуйста, введите корректные значения диапазона.")  
 }  
 }  
 "2"->{  
 println("Введите пороговое значение для круговой диаграммы:")  
 val threshold = readLine()?.toDoubleOrNull()  
 if (threshold != null) {  
 pieChart.filterData(threshold)  
 } else {  
 println("Ошибка: Пожалуйста, введите корректное значение.")  
 }  
 }  
 }  
 }  
 "5" -> {  
 println("Выберите график для изменения диапазона:")  
 println("1 - Линейный график")  
 println("2 - Столбчатый график")  
  
 val graphChoice = readLine()  
 when (graphChoice) {  
 "1" -> {  
 println("Введите новый диапазон для оси X (min и max через запятую):")  
 val xRange = readLine()?.split(",")?.map { it.trim().toDouble() }  
 println("Введите новый диапазон для оси Y (min и max через запятую):")  
 val yRange = readLine()?.split(",")?.map { it.trim().toDouble() }  
 if (xRange != null && yRange != null && xRange.size == 2 && yRange.size == 2) {  
 lineChart.setAxisRange(xRange[0], xRange[1], yRange[0], yRange[1])  
 } else {  
 println("Ошибка: Пожалуйста, введите корректные значения диапазона.")  
 }}  
 "2" -> {  
 println("Введите новый диапазон для оси Y (min и max через запятую):")  
 val yRange = readLine()?.split(",")?.map { it.trim().toDouble() }  
 if (yRange != null && yRange.size == 2) {  
 barChart.setAxisRange(yRange[0], yRange[1]) // Изменяем диапазон осей  
 } else {  
 println("Ошибка: Пожалуйста, введите корректные значения диапазона.")  
 }  
 }  
 }  
 }  
 "0" -> {  
 println("Возврат в главное меню.")  
 return  
 }  
 else -> println("Неверный выбор. Пожалуйста, введите 1, 2, 3, 4, 5 или 0.")  
 }  
}}  
  
fun main() {  
 // Создаем один график и используем его для отображения и изменения  
 val lineChart = LineChart("Линейный график", "Ось X", "Ось Y")  
 val barChart = BarChart("Столбчатый график", "Категории", "Значения")  
 val pieChart = CustomPieChart("Круговая диаграмма")  
  
 while (true) {  
 println("Выберите действие:")  
 println("1 - Отображение графиков")  
 println("2 - Динамическое изменение данных")  
 println("0 - Выход")  
  
 val choice = readLine()  
  
 when (choice) {  
 "1" -> displayCharts(lineChart, barChart, pieChart) // Отображение графиков  
 "2" -> dynamicUpdate(lineChart, barChart, pieChart) // Динамическое изменение данных  
 "0" -> {  
 println("Выход из программы.")  
 return  
 }  
  
 else -> println("Неверный выбор.")  
 }  
 }  
}

**Контрольные вопросы:**

1. Что такое класс в kotlin, и как он объявляется?

Класс в kotlin — это шаблон для создания объектов (экземпляров), который может содержать свойства и методы. Класс объявляется с помощью ключевого слова class, после чего следует имя класса и его тело в фигурных скобках.

1. Как создать объект класса в kotlin? Приведите пример.

Чтобы создать объект класса, используйте оператор val или var, за которым следует имя объекта и вызов конструктора класса.

Class person(val name: string, val age: int)

Fun main() {

val person = person("alice", 30) // создание объекта

println("${person.name} is ${person.age} years old.")

}

1. Что такое свойства класса, и как их объявить в kotlin?

Свойства класса — это переменные, которые хранят состояние объекта. Они объявляются внутри класса с помощью ключевых слов val (для неизменяемых свойств) или var (для изменяемых).

1. Как объявить и использовать метод класса? Приведите пример.

Метод класса — это функция, определенная внутри класса. Методы могут использоваться для выполнения действий или возвращения значений.