Министерство транспорта Российской Федерации

Федеральное агентство железнодорожного транспорта

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Дальневосточный государственный университет путей сообщения»

Естественно-научный институт

Кафедра «Вычислительная техника и компьютерная графика»

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ JAVA

Отчет к расчетно-графической работе №1

РГР.09.03.03. JAVA-П.01.07.000 – БО941ПРИ

Исполнитель  
студент Д. Е. Дуда

Руководитель  
преподаватель И. В. Кузнецов

Хабаровск 2022

Содержание

[Введение 3](#_Toc99445511)

[1 Сведения о программном обеспечении 4](#_Toc99445512)

[1.1 IntelliJ IDEA 4](#_Toc99445513)

[1.2 JavaFX 5](#_Toc99445514)

[2 Ход работы 6](#_Toc99445515)

[2.1 UML диаграмма 6](#_Toc99445516)

[2.2 Код программы 6](#_Toc99445517)

[Заключение 18](#_Toc99445518)

[Список использованных источников 19](#_Toc99445519)

Введение

Целью расчетно-графической работы является создание десктопного приложения, удовлетворяющего следующим требованиям:

* использовать возможности ООП: классы, наследование, полиморфизм, инкапсуляция;
* каждый класс должен иметь отражающее смысл название и информативный состав;
* наследование должно применяться только тогда, когда это имеет смысл;
* при кодировании должны быть использованы соглашения об оформлении кода java code convention;
* классы должны быть грамотно разложены по пакетам;
* консольное меню должно быть минимальным;
* для каждого класса должны быть созданы документированные комментарии с использованием дескрипторов
* реализовать обработчики исключений
* добавить возможность сохранения и загрузки информации в текстовые файлы. Процессы чтения из файла и записи в файл должны находиться в отдельных потоках выполнения.
* разработать систему протоколирования
* интерфейс должен быть реализован на JavaFX.

Определен следующий вариант на выполнение расчетно-графической работы:

* клиент может иметь несколько счетов в банке;
* учитывать возможность блокировки/разблокировки счета;
* реализовать поиск и сортировку счетов;
* вычисление общей суммы по счетам;
* вычисление суммы по всем счетам, имеющим положительный и отрицательный балансы отдельно.

1. Сведения о программном обеспечении
   1. IntelliJ IDEA

IntelliJ IDEA – это интеллектуальная IDE, учитывающая контекст. Она предназначена для разработки разнообразных приложений на Java и других языках JVM, например Kotlin, Scala и Groovy. Кроме того, IntelliJ IDEA Ultimate поможет в разработке веб-приложений: она предлагает эффективные встроенные инструменты, поддержку JavaScript и связанных с ним технологий, а также расширенную поддержку таких популярных фреймворков, как Spring, Spring Boot, Jakarta EE, Micronaut, Quarkus и Helidon. А бесплатные плагины, разработанные JetBrains, позволяют дополнительно расширить возможности IntelliJ IDEA и использовать ее для работы с другими языками программирования, в том числе Go, Python, SQL, Ruby и PHP.

IntelliJ IDEA продумана в каждом аспекте и готова к использованию сразу после установки. Среда обеспечивает быстрый доступ ко всем функциям и встроенным инструментам, необходимым разработчику, а также широкие возможности индивидуальной настройки. Вы можете полностью настроить среду в соответствии со своим рабочим процессом: задать сочетания клавиш, установить плагины, настроить интерфейс по своему усмотрению и т. д.

JetBrains стремится сделать свои продукты доступными для всех желающих. IntelliJ IDEA предлагает различные специальные возможности, которые могут вам понадобиться. Например, с IntelliJ IDEA совместимы программы чтения с экрана. Можно также настроить цвета разных элементов интерфейса, добавить контрастности полосам прокрутки, изменить размер окон и шрифта в редакторе и т. п.

IntelliJ IDEA была создана в первую очередь для разработки на Java, но она понимает и многие другие языки программирования, в том числе Groovy, Kotlin, Scala, JavaScript, TypeScript и SQL, и предлагает интеллектуальную помощь в написании кода на каждом из этих языков. Первоначальная индексация исходного кода позволяет IDE создать виртуальную карту проекта. Используя информацию виртуальной карты, она мгновенно обнаруживает ошибки, предлагает варианты автодополнения кода с учетом контекста, выполняет рефакторинг и т. д.

IntelliJ IDEA обладает широкими возможностями проверки качества и валидности кода с помощью инспекций, которые выполняются «на лету». Инспекции помогают быстрее писать код в соответствии с самыми строгими стандартами качества и чувствовать себя уверенно на протяжении всего процесса разработки. Мы предлагаем набор стандартных инспекций и десятки инспекций для отдельных фреймворков. Они позволяют выявить самые разные проблемы: например, ошибки при автоматическом связывании бинов Spring и т. п.

* 1. JavaFX

JavaFX – платформа на основе Java для создания приложений с насыщенным графическим интерфейсом. Может использоваться как для создания настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет-приложений (RIA), работающих в браузерах, и для приложений на мобильных устройствах. JavaFX призвана заменить использовавшуюся ранее библиотеку Swing. Платформа JavaFX конкурирует с Microsoft Silverlight, Adobe Flash и аналогичными системами.

Начиная с версии Java 11 больше не входит в Java SE и не разрабатывается компанией Oracle (как отдельный модуль поддерживается компанией Gluon). Oracle будет поддерживать JavaFX в составе Java JDK 8 до марта 2025.

Приложения JavaFX создаются с помощью декларативного языка программирования JavaFX Script. Для разработки приложений на языке JavaFX Script необходимо установить JavaFX SDK, который входит в поставку с Java SE 7. Из кода, написанного на языке JavaFX Script, можно обращаться к любым библиотекам Java. Поэтому совместное использование языков Java и JavaFX Script позволяет решать разнообразные задачи, например, логика бизнес-приложения может быть написана на Java, а графический интерфейс пользователя – на JavaFX Script.

1. Ход работы

Для выполнения задания необходимо составить диаграмму классов и написать программу по варианту.

* 1. UML диаграмма

Диаграмма представлена на рисунке 1.

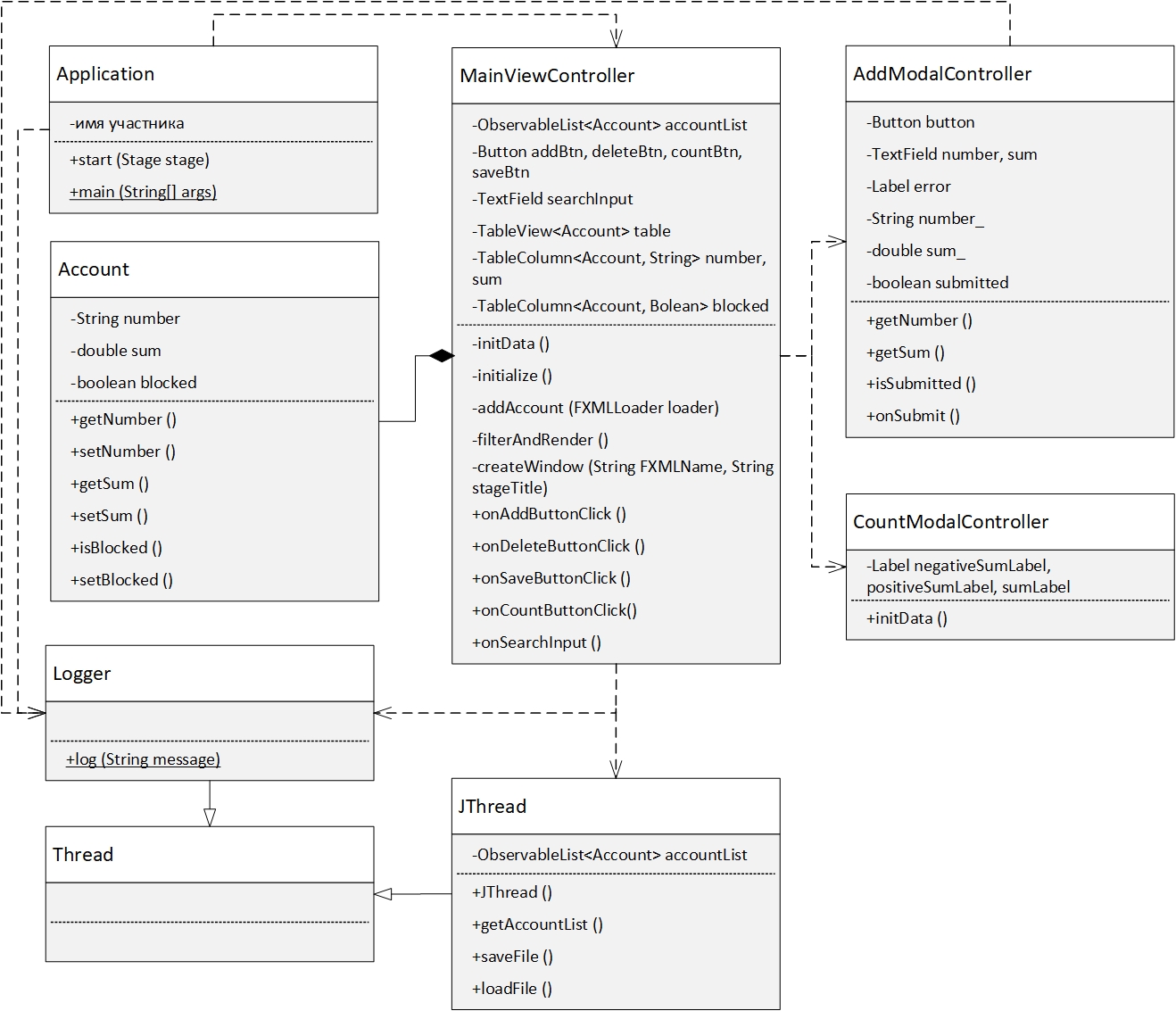


Рисунок 1 - UML диаграмма классов

* 1. Код программы

Листинг 1 – Application

package com.example.demo;

import javafx.application.Platform;

import javafx.fxml.FXMLLoader;

import javafx.scene.Scene;

import javafx.stage.Stage;

import java.io.IOException;

public class Application extends javafx.application.Application {

@Override

public void start(Stage stage) throws IOException {

FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("main\_view.fxml"));

Scene scene = new Scene(fxmlLoader.load(), 800, 600);

stage.setTitle("Hello!");

stage.setScene(scene);

stage.show();

stage.setOnCloseRequest(event -> {

Logger.log("Выход из приложения");

Platform.exit();

System.exit(0);

});

}

public static void main(String[] args) {

launch();

}

}

Листинг 2 – MainViewController

package com.example.demo;

import com.example.account.Account;

import javafx.beans.property.\*;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.CheckBoxTableCell;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.control.cell.TextFieldTableCell;

import javafx.stage.Modality;

import javafx.stage.Stage;

import java.io.IOException;

public class MainViewController {

private ObservableList<Account> accountList = FXCollections.observableArrayList();

@FXML

private Button addBtn, deleteBtn, countBtn, saveBtn;

@FXML

private TextField searchInput;

@FXML

private TableView<Account> table;

@FXML

private TableColumn<Account, String> number;

@FXML

private TableColumn<Account, String> sum;

@FXML

private TableColumn<Account, Boolean> blocked;

private void initData() {

JThread jThread = new JThread("load", accountList);

jThread.loadFile();

this.accountList = jThread.getAccountList();

Logger.log(String.format("Запуск приложения"));

}

@FXML

private void initialize() {

table.setPlaceholder(new Label("Банковские счета отсутствуют"));

// Загрузка данных из файла

initData();

// Привязка колонок таблицы

number.setCellValueFactory(new PropertyValueFactory<Account, String>("number"));

sum.setCellValueFactory(cellData -> new ReadOnlyStringWrapper(String.valueOf(cellData.getValue().getSum())));

blocked.setCellFactory(column -> new CheckBoxTableCell<>());

// Отрисовка сущностей в таблице

table.setItems(accountList);

// Редактирование полей в таблице

number.setCellFactory(TextFieldTableCell.<Account> forTableColumn());

number.setOnEditCommit((TableColumn.CellEditEvent<Account, String> event) -> {

TablePosition<Account, String> pos = event.getTablePosition();

String number = event.getNewValue();

int row = pos.getRow();

Account account = event.getTableView().getItems().get(row);

account.setNumber(number);

Logger.log(String.format("Номер счета %s изменился на %s", event.getOldValue(), account.getNumber()));

});

sum.setCellFactory(TextFieldTableCell.<Account> forTableColumn());

sum.setOnEditCommit( (TableColumn.CellEditEvent<Account, String> event) -> {

String oldValue = event.getOldValue();

String sum = event.getNewValue();

try {

Double.parseDouble(sum);

} catch (NumberFormatException e) {

table.refresh();

return;

}

TablePosition<Account, String> pos = event.getTablePosition();

int row = pos.getRow();

Account account = event.getTableView().getItems().get(row);

account.setSum(Double.parseDouble(sum));

Logger.log(String.format("Сумма на счету %s изменилась с %s на %s", account.getNumber(), oldValue, sum));

});

blocked.setCellValueFactory(cellData -> {

Account cellValue = cellData.getValue();

BooleanProperty property = new SimpleBooleanProperty(cellValue.isBlocked());

property.addListener((observable, oldValue, newValue) -> {

cellValue.setBlocked(newValue);

Logger.log(String.format("Статус блокировки счета %s изменился с %s на %s", cellValue.getNumber(), oldValue, newValue));

});

return property;

});

}

@FXML

void onAddButtonClick () {

FXMLLoader loader = createWindow("addModal.fxml", "Добавление счета");

addAccount(loader);

}

@FXML

void onDeleteButtonClick () {

int indexRemove = table.getSelectionModel().getSelectedIndex();

if (indexRemove != -1) {

Account account = table.getSelectionModel().getSelectedItem();

accountList.remove(account);

filterAndRender();

Logger.log(String.format("Удален счет с номером %s", account.getNumber()));

}

}

@FXML

void onSaveButtonClick () {

JThread jThread = new JThread("save", accountList);

jThread.saveFile();

Logger.log(String.format("Сохранение изменений"));

}

@FXML

void onCountButtonClick () {

FXMLLoader loader = new FXMLLoader(getClass().getResource("countModal.fxml"));

try {

Parent root = loader.load();

CountModalController controller = loader.getController();

controller.initData(accountList);

Stage stage = new Stage();

stage.setTitle("Сводка по счетам");

stage.setScene(new Scene(root));

stage.initModality(Modality.APPLICATION\_MODAL);

stage.showAndWait();

} catch (IOException e) {

e.printStackTrace();

}

}

@FXML

void onSearchInput () {

filterAndRender();

}

private FXMLLoader createWindow(String FXMLName,String stageTitle) {

FXMLLoader loader = new FXMLLoader(getClass().getResource(FXMLName));

try {

Parent root = loader.load();

Stage stage = new Stage();

stage.setTitle(stageTitle);

stage.setScene(new Scene(root));

stage.initModality(Modality.APPLICATION\_MODAL);

stage.showAndWait();

} catch (IOException e) {

e.printStackTrace();

}

return loader;

}

private void addAccount (FXMLLoader loader) {

AddModalController addModalController = loader.getController();

String number = addModalController.getNumber();

double sum = addModalController.getSum();

if (addModalController.isSubmitted()) {

Account account = new Account(number, sum, false);

accountList.add(account);

filterAndRender();

Logger.log(String.format("Добавлен новый счет с номером %s и суммой %s", account.getNumber(), account.getSum()));

}

}

private void filterAndRender() {

String searchStr = searchInput.getText();

table.setItems(accountList.filtered(account -> account.getNumber().contains(searchStr)));

}

}

Листинг 3 – Account

package com.example.account;

public class Account {

private String number;

private double sum;

private boolean blocked;

public Account (String number, double sum, boolean blocked) {

this.number = number;

this.sum = sum;

this.blocked = blocked;

}

public void setNumber (String number) {

this.number = number;

}

public void setSum (double sum) {

this.sum = sum;

}

public String getNumber () {

return this.number;

}

public double getSum () {

return this.sum;

}

public boolean isBlocked() {

return blocked;

}

public void setBlocked(boolean blocked) {

this.blocked = blocked;

}

@Override

public String toString(){

return number + " " + sum + " " + blocked;

}

}

Листинг 4 – JThread

package com.example.demo;

import com.example.account.Account;

import javafx.collections.ObservableList;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.PrintWriter;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.List;

public class JThread extends Thread {

ObservableList<Account> accountList;

JThread(String name, ObservableList<Account> accountList){

super(name);

this.accountList = accountList;

}

public ObservableList<Account> getAccountList() {

return accountList;

}

public void saveFile(){

System.out.printf("%s started... \n", Thread.currentThread().getName());

try{

PrintWriter pw = new PrintWriter(new FileOutputStream("D:\\Study\\8\_semester\\Java\\demo\\src\\main\\resources\\com\\example\\demo\\data.txt"));

for (Account account : accountList){

pw.println(account); // call toString() on club, like club.toString()

}

pw.close();

System.out.printf("ok");

}

catch(IOException e){

System.out.println("Thread has been interrupted");

}

System.out.printf("%s fiished... \n", Thread.currentThread().getName());

}

public void loadFile() {

System.out.printf("%s started... \n", Thread.currentThread().getName());

try {

List<String> list = null;

list = Files.readAllLines(Paths.get("D:\\Study\\8\_semester\\Java\\demo\\src\\main\\resources\\com\\example\\demo\\data.txt"));

String[] param;

for (String row : list){

try{

Thread.sleep(500);

}catch (Exception ex){

System.out.println(ex.getMessage());

}

param = row.split(" ");

accountList.add(new Account(param[0],

Double.parseDouble(param[1]),

param[2].equals("true")));

}

} catch (IOException e) {

System.out.println("Thread has been interrupted");

}

System.out.printf("%s fiished... \n", Thread.currentThread().getName());

}

}

Листинг 5 – Logger

package com.example.demo;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Date;

public class Logger extends Thread {

public static void log (String message) {

try {

PrintWriter pw = new PrintWriter(new FileOutputStream("D:\\Study\\8\_semester\\Java\\demo\\src\\main\\resources\\com\\example\\demo\\log.txt", true));

pw.append(new Date() + ": " + message + "\n");

pw.close();

} catch (IOException e) {

System.out.println("Logger thread has been interrupted");

}

}

}

Заключение

Язык программирования Java – это полностью объектно-ориентированный язык, который в отношении синтаксиса многое унаследовал от С++. Конечно, преимущества Java далеко не исчерпываются межплатформенностью. Язык Java в синтаксическом отношении проще и логичнее, чем С++. Java как платформа предоставляет в распоряжение программистов большое количество библиотек (пакетов), в которых содержится большое количество описаний классов и интерфейсов на все случаи жизни. С их помощью можно создавать стопроцентные приложения Java с возможностью обращения к базам данных, поддержкой передачи почтовых сообщений, с клиентской частью, которой необходим web-браузер, или, наоборот, с клиентской частью, обладающей изощренным интерфейсом.

Результатом расчетно-графической работы стало программное обеспечение, реализованное с помощью JavaFX.

Список использованных источников

|  |
| --- |
| 1. ГОСТ 7.0.5-2008, 2008.  2. Гопкало, В.Н. Выпускная квалификационная работа, / В.Н. Гопкало, О.А. Графский, Хабаровск: Изд-во ДВГУПС, 2018. |
|  |