

PROJECT OOP2 SCHALTERASSISTENT



Harun Mustafa

24.02.2023 OOP2 Tutor Alain Rohr

Table of contents

Introduction	2
Development of the environment	3
Requirements.....	3
Design.....	4
Project structure	6
Programming	7
Global	7
Konfiguration	7
Printer	8
QR.....	8
QRWithList	8
QRSchweiz.....	8
Standort	8
Websocket	8
Main	8
Primary Controller	9
Primary fxml.....	9
Testing.....	10
GlobalTest	10
KonfigTest	10
PrimaryControllerTest	10
FXBuilder Fail	10
Closing	12

Introduction

We were given a project as an assignment by our instructor Alain Rohr, covering the last few months of school material and culminating in a project. We had the opportunity to review and apply everything we have learned. For my assignment, I chose the "Schalterassistent". An application for the Office of Population Services of the Canton of Bern (BUND).

I created the application back in October within three weeks. As sources of information, I had one document and weekly one call with the customer. The Schalterassistent is an application with an interface via websocket.

There are two processes and a configuration section. The application finally prints a receipt with some information and a QR code. During the evaluation of the technology, the software architect ruled out a web application so that in return there will be a desktop application. The reasoning behind this decision was to ensure that the application remained fast, simple and efficient.

Since websites still have trouble with printers and you can't get around pdf's, this means that the process becomes longer. In .NET and java there is a drawing method, which draws the print and then passes it directly to the printer, so no pdf or other files have to be created.

I will now rewrite the project from C# to Java and follow the rules for clean code. When I implemented the project for the client, I had little time and not really all the information together. So as I rewrite I will challenge myself with the following: I take into account Uncle Bob and his Clean Code bible, SOLID, Git Workflow and of course the documentation in English.

Development of the environment

In the last project OOP1 I worked as usual with Eclipse for Java. Now I wanted to implement it, as suggested by the school, with VSC and Scenebuilder.

For reference (also from school) I have the Holy Bible of Java, which is always a help for beginners as well as advanced users.

Requirements

The application shall have a graphical interface implemented with JavaFX. There should be the possibility that the user can enter something in the program and this input is then processed accordingly.

- a) Create at least four unit tests that check important parts of your application logic.
- b) Enumeration type with Enum
- c) Use at least three of the following techniques: Inheritance, casting, interfaces, abstract classes, or generics.
- d) Collections and sorting
- e) Certain data should be able to be stored by the user with Java Serialisierung that these also still available after restarting the application.
- f) Use the possibilities of Exceptions purposefully, in order to handle exceptions in your application cleanly. Make sure that an error can be analyzed in detail with the help of log data.
- g) The project should be delivered as a Maven project and can be generated via the build process of Maven. An already compiled and directly executable version of the program should also be included in the delivery. Use packages to structure your classes according to a chosen architecture approach.

Design

For the design I had a framework of how it should look like.

During the weekly meetings with the customer, the design was adjusted accordingly until the end.

Here is a screenshot of the original solution in .NET and below the Java/Scenebuilder solution.

The screenshot shows a Windows application window titled 'Schalterassistent'. The interface is divided into several sections:

- Unterlagen**: A large empty rectangular box for document upload, with a 'Kopieren' button below it.
- Folgende Unterlagen sind noch einzureichen**: A grid of 12 checkboxes for required documents:
 - Einwilligungserklärung
 - Ausweiskopie Vater
 - Ausweiskopie Mutter
 - Verlustmeldung
 - Pass/ID zur Entwertung
 - Rechtskraftbescheinigung
 - Sorgerechtsentscheidung
 - Erklärung über die gemeinsame
 - Entscheid KESB
 - Bestätigung der Personendaten
 - Niederlassungsausweis
 - Geburtsurkunde oder Familienausweis
 - Ausweis des Heimatlandes
 - Gesuch Austauschpass
 - Abklärung
- Zusätzliche Produkte**: A list of three checkboxes for additional products:
 - Verlustmeldung ID
 - Verlustmeldung Pass
 - Notfall Pass
- Produkte für ausländische Staatsangehörige**: A section for foreign citizens with input fields for 'Zemis Nummer (NAA) / Auftragsnummer (ISR)' and 'Name, Vorname', and buttons for 'Reisedokument SEM', 'NAA Drittstaaten', and 'NAA EU/EFTA'. Below these are checkboxes for 'Geburtsurkunde oder Familienausweis' and 'Pass des Heimatlandes'.
- Portal ID**: An input field.
- Sprache**: A dropdown menu.
- Standort**: A dropdown menu.
- Drucker**: A dropdown menu.
- Konfig Datei**: A button.

Figure 1 .NET solution

Schweiz QR

Folgende Unterlagen sind noch einzureichen

☐ Einwilligungserklärung

☐ Rechtskraftbescheinigung

☐ Niederlassungsausweis

☐ Ausweiskopie Vater

☐ Pass/ID Entwertung

☐ Sorgerechtsentscheidung

☐ Geburtsurkunde oder Familienausweis

☐ Erklärung

☐ Ausweiskopie Mutter

☐ Bestätigung der Personendaten

☐ Ausweis des Heimatlandes

☐ Verlustmeldung

☐ Entscheid KESB

☐ Gesuch Austauschpass

☐ Abklärung

Zusätzliche Produkte

☐ Verlustmeldung ID

☐ Verlustmeldung Pass

☐ Notfall Pass

Produkte für ausländische Staatsangehörige

Zemis Nummer (NAA) / Auftragsnummer (ISR)

Name, Vorname

Folgende Unterlagen sind noch einzureichen (NAA)

☐ Geburtsurkunde oder Familienausweis

☐ Pass des Heimatlandes

Reisedokument SEM

NAA Drittstaaten

NAA EU/EFTA

Konfiguration

PortalID

Standort

Drucker

Konfig

Figure 2 Java solution

Project structure

After installing and configuring all the necessary components and establishing the connection to the scene builder, the structure is adjusted.

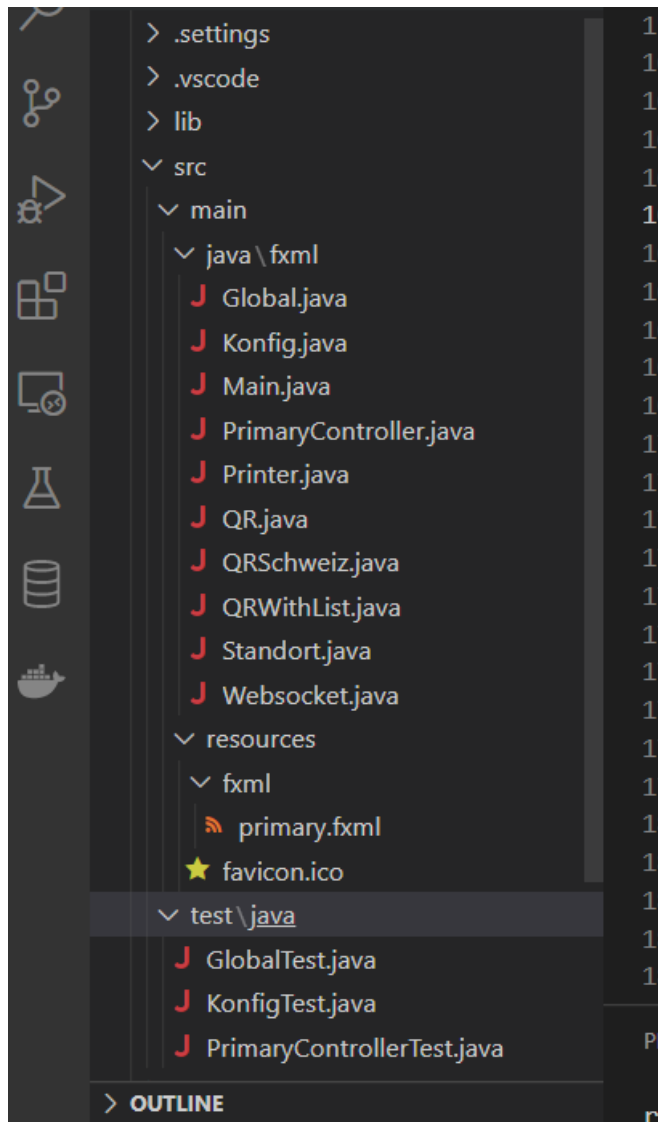


Figure 3 project structure

Programming

As mentioned before, I will prioritize the clean programming and analyze my project from .NET with reverse engineering.

As I work in the project business I have usually a deadline and need to fulfill that as soon as possible with the requirements from the customer. This can lead to redundant or less efficient codes.

First I will explain the classes used in Java and then compare them with the .NET project.

Global

The Global class is a collection of static functions that are reused.

An example:

```
public static String getItemInCsvFile(String artikel) {
    try {
        String path = getCsvFile();
        List<String> lines = Files.readAllLines(Paths.get(path));
        for (String line : lines) {
            String[] parts = line.split(",");
            if (parts[0].equals(artikel)) {
                return parts[1];
            }
        }
        return "Konfig File has an Error";
    } catch (IOException e) {
        JOptionPane.showMessageDialog(null, "ERROR - the file is missing! " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
    return "Konfig File has an Error";
}
```

So, when the user presses a button to trigger the print, the button is the first item to be processed by the function. This button gets an ID, which is stored in the Article.CSV. The ID is iterated through the CSV until it has a match. This works from key and then refers to the value and returns this. This ensures that the correct article is selected for the cash register and is understood by the system.

Konfiguration

All functions for the configuration of the application are stored there.

Also, the Portal ID, location and printer are configured there. It is intended that the user sets these at the very first time and basically never has to change it. The input is written to a JSON file and is stored in the Windows client under Path appFolderPath = Paths.get(System.getProperty("user.home"), "AppData", "Roaming", APP_FOLDER_NAME); App folder Name = Schalterassistent.

But if necessary he can change it and the program handles it.

Printer

The Printer class provides a method to print an image file using the default printer on the user's system. It uses the Java Print Service API to find and select the default printer and sends the image file to be printed using a print job. The class also sets attributes such as orientation and media size to customize the printing.

QR

The QR class represents a QR code with a title, order number, name and footer. It has getter and setter methods for each of these properties.

QRWithList

QRWithList class extends the QR class and adds a list of selected checkboxes as a new field, allowing for the creation of QR codes with additional information from QR if documentation needs to be submitted.

QRSchweiz

The QR Switzerland class is an auxiliary class for the websocket message. Once a message has been received via the websocket, the class packs it into a list. In the QR Switzerland class the list is iterated through and the important values are extracted and prepared for the QR.

After that the composition of the QR follows and is released for printing.

Standort

An enum which returns the correct URL for the websocket connection. The URL will be displayed attached to the combobox location.

Websocket

The Websocket class creates a WebSocket client and handles WebSocket events such as opening, receiving messages, closings, and errors. It also provides methods for connecting, disconnecting, and sending messages to the WebSocket server.

Main

The Main class is the entry point for the JavaFX application. It loads the primary FXML file, sets up the application scene, and creates an instance of the Websocket class. The setRoot method is used to switch between different FXML scenes, and the loadFXML method is used to load FXML files. The main method launches the JavaFX application.

Primary Controller

This code is a JavaFX application that is used to create checkboxes, combo boxes, text fields and buttons. It also uses external libraries to generate QR codes and read barcodes.

The initialize() method sets up the printer function and selects the current location from the ComboBoxStandort. The handleCheckBoxSelection() method is called when a checkbox is clicked and it adds or removes the item from the selectedCheckboxes list.

The handleButtonClick() method is called when one of the three buttons (naaEuEftaButton, naaDrittstaatenButton, reisedokumentSemButton) is clicked. It checks whether both txtAuftrag and txtNameVorname have values, and if not, it displays an error message. If both fields have values, it retrieves an item from a CSV file based on which button was clicked, and uses that item for further processing.

Primary fxml

This is a JavaFX code for a form with checkboxes and labels. The form has two sections, one for the required documents and one for additional products. The checkboxes are used to select the required documents and additional products.

The form has a controller named "PrimaryController" which handles the selection of the checkboxes using the handleCheckBoxSelection method. The form has an AnchorPane as the root node and several CheckBox, Label, and Line nodes as its children. The CheckBox nodes have fx:id attributes to identify them in the controller.

The required documents section has checkboxes for the consent declaration, proof of legal force, copies of parents' identification, residence permit, parental responsibility declaration, birth certificate or family register, identification from the home country, custody decision, application for exchange pass, decision by KESB, and loss report. The additional products section has checkboxes for loss report for ID card, loss report for passport, and emergency passport

Testing

Some J Unit tests were written to test some functions.

GlobalTest

The test method verifies that the result of calling getCSVFile method matches the expected path by using the Assertions.assertEquals method. If the result matches the expected value, the test passes; otherwise, it fails.

Overall, this test method ensures that the getCSVFile method in the Global class returns the expected path to the CSV file, which is necessary for the proper functioning of the project.

KonfigTest

This code defines test cases for the Konfig class. The first test case checks if the createAppFolder() method correctly creates a folder in the user's AppData/Roaming directory. The second test case checks if the getPortalId() method returns a non-null and non-empty String. If an IOException is thrown, the test case fails with a message.

PrimaryControllerTest

This is a unit test for the handleButtonClick method in the PrimaryController class. It uses the Mockito library to create a mock(ActionEvent) object and verify that the getSource method is called on it. It also includes a placeholder assertion to ensure that the test is not marked as a failure.

FXBuilder Fail

In the middle of the development, the big scare came.

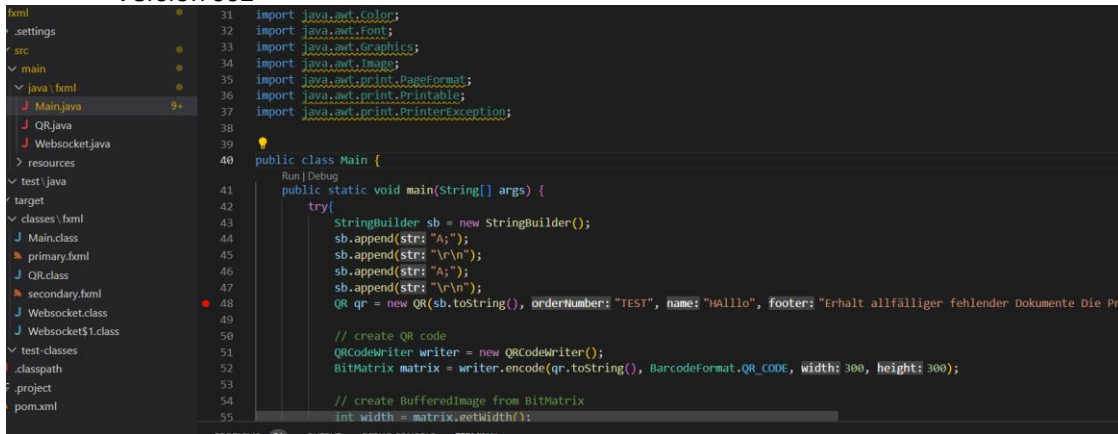
I could no longer execute and debug my program. The error message was: the FXML file could not be found/loaded.

I hoped for a solution when I had a look at it with Alain, but unfortunately without success.

Now I had two possibilities:

- 1) I recreate my FXML file and try to find out where what went wrong.
- 2) or my idea: I develop all logic in an external untouched project and insert it into the project.
 1. I decided to use the 2 method because I don't have enough time to restart the project and find the error. HALELULJA FXML!

Version 002



```
31 import java.awt.Color;
32 import java.awt.Font;
33 import java.awt.Graphics;
34 import java.awt.Image;
35 import java.awt.print.PageFormat;
36 import java.awt.print.Printable;
37 import java.awt.print.PrinterException;
38
39
40 public class Main {
41     public static void main(String[] args) {
42         try {
43             StringBuilder sb = new StringBuilder();
44             sb.append(str: "A;");
45             sb.append(str: "\r\n");
46             sb.append(str: "A;");
47             sb.append(str: "\r\n");
48             QR qr = new QR(sb.toString(), orderNumber: "TEST", name: "Hallo", footer: "Erhalt allfälliger fehlender Dokumente Die Pro
49
50             // create QR code
51             QRCodeWriter writer = new QRCodeWriter();
52             BitMatrix matrix = writer.encode(qr.toString(), BarcodeFormat.QR_CODE, width: 300, height: 300);
53
54             // create BufferedImage from BitMatrix
55             int width = matrix.getWidth();
```

Unfortunately, this resulted in my inability to deliver an executable.

Closing

In conclusion, the Schalterassistent project was a great opportunity to apply the concepts and principles learned throughout the OOP1 and OOP2 courses, as well as the Clean Code and SOLID principles. It also allowed me to gain experience in working with JavaFX and Scenebuilder.

But what I find extremely unfortunate, we should use a proper IDE for java, such as Netbeans or Eclipse. It does not have any less compatibility problems than mtidem VSC. I really had huge troubles configuring it and it was very tedious and demotivating. What I also find a pity is that debugging is not really looked at. Maybe it doesn't fit in here and there should be a separate course for it. However, my experience and a survey of my developer colleagues has shown that we spend about 30-40% with debugging. Some non computer scientists among my student colleagues could not use debugging correctly and did not understand the difference between f10 and f11. I hope that this message will be taken really seriously.

Through the development process, I paid particular attention to writing a clean, efficient, and readable code, which would be easily maintainable and scalable in the future. I also utilized various Java concepts and techniques, such as Enums, Collections as well as Java Serialization and Exception Handling.

I was also able to gain deep insights and comparisons between C# and Java. as far as the frontend is concerned, for example, the .NET environment is much more user-friendly and easier to use. In terms of libraries, I also had more choice in .NET, but this is not always positive, it also has rather less good libraries. Otherwise, these two programming languages are remarkably similar and easy to read/understand.

Additionally, I followed the project requirements, including the implementation of unit tests, the use of packages for structure, and the creation of a Maven project. The resulting application provides an intuitive and user-friendly interface for the Office of Population Services of the Canton of Bern (BUND) to assist with "Schalterassistent", while also allowing for easy configuration.

Overall, this project has been a valuable learning experience, and I am grateful for the opportunity to work on it.