CS22120: Software Engineering Group Project 10 Maintenance Manual

Author: Keean Griffith, Juan Manuel Palma, Alex Clive

Conf. Ref: MaintenanceManualGroup10_doc

Date: 05/05/21

Version: 1.0

Status: Release

Department of Computer Science

Aberystwyth University

Aberystwyth
Ceredigion

SY23 3DB

Contents

1 Introduction

1.1 Purpose of this document

The purpose of this document is to provide an overview of how the program has been developed so that maintainers can easily modify the current app or add new features in the future.

1.2 Scope

This document provides important information for modifications to be made by a developer who is not familiarized with the program itself.

This document is intended to be read by program maintainers. It is assumed the reader is already familiar with the Design Specification document [1] as well as the Test specification [2] and the User Interface Specification [3]. It is important the reader is already familiar with the Quality Assurance plan [4] to maintain the formalisation of the program development.

1.3 Objectives

The objective of this document is to provide a system description of the following aspects:

- Program structure and relation between the different components.
- Further clarification on complex algorithms.
- Data management and files.
- Suggested improvements and parts of the program susceptible to generate side effects when making changes.
- Physical limitations.
- Instructions for rebuilding and testing the program.

2 Program Description

The main requirement of this program is to guide a user through a series of timed exercises with pauses between each exercise whilst explaining or displaying what the user should be doing at any point. The program does this by allowing the user to select some workout parameters e.g., No. of exercises, which the program then takes a random list of 30 exercises and randomises to fit the user's parameters e.g., if the user selects the number of exercises of 15, the program takes 15 random exercises from the list of 30.

3 Program Structure

A list of program modules and their purpose can be found:
Design Specification - 2.2.1 Significant classes in Program 1.
A list of methods and a brief description of what each does can be found:
Design Specification - 4 Interface Description.

4 Algorithms

Design Specification - 5.2 Significant Algorithms.

5 Main Data Areas

NodeList and Node are used a lot throughout the program. NodeLists are generated by the xpath library that is used (javax.xml.xpath). The return of this is type casted into an easily manipulatable NodeList (from org.w3c.dom)
NodeList can be converted to Node through the use of *NodeList.item(int)* and further to a String by *Node.getTextContent()*.

WorkoutHistory is a class that contains all of the necessary information to populate one row of the history table. A list of this class is populated by WorkoutRead and used to show all of the previous Workouts performed.

Enumerations are used to denote the intensity of workouts and exercises. Unfortunately, neither is used to the extent that was originally intended.

6 Files

There are 2 .XML files that are accessed by the program. These into generic data (regarding the exercises and routines) and history data.The current directory is src/uk/ac/aber/cs221/group10/app/data/

They are split this way to isolate the two file functions present: storing and reading previously done workouts and reading (but not writing) the information regarding possible workouts. (further expanded on in section 8).

7 Interfaces

Sliders and buttons are the only form of user input. This was done to both lower the need for type conversions that have the potential to cause problems and have an easy-to-use, aesthetically pleasing user interface.

8 Suggestions for improvement

Throughout the completion of this program there were multiple features and implementations that had to be omitted due to time constraints. Firstly, I believe an improvement that could be made would be implementing some way for the user to save their Custom workout configurations to the “Preconfigured Workouts” section of the program allowing them to go back

and complete the workout again without having to remember and re input the parameters. Another improvement I would suggest to a future programmer would be implementing some way of allowing the user to choose workouts that are created based on the specific intensity of the exercises it contains e.g. Having preconfigured workouts whose exercise intensity is different like Low Intensity workouts that contain only lower intensity exercises or vice versa. I believe this would be a lot more options for different users.
The last suggestion for improvement I would make is ensuring the video demonstration of each exercise plays automatically when the rest stage begins, we thought allowing the user to choose whether they play or pause the video would be a good idea. However, we failed to consider the user would have to be at the computer to watch the video which they may not be able to do whilst working out.

9 Things to watch when making changes

Despite not being used, the Intensity attributes are populated in the respective representation classes (Exercise and WorkoutRoutineConfig). If more values are to be added to Data.xml, make sure that intensity matches the appropriate enum.

The XML was split into two files: data and history. The split done here will need to be looked at during future development as the saving logic was placed into WorkoutStorage and not XMLio. Because the logic was placed here and written without too much future consideration, the names of the attributes were hard-coded, meaning that it will also need to be generalised if ever moved. (This is avoiding using a new function in WorkoutRead that would be very similar to the one in WorkoutStorage).

10 Physical limitations of the program

We had problems with the .Jar not working properly at first.

When it comes to the accuracy of inputs in our program, we tried our best to eliminate the majority of potential human/user error when it came to needing input from a user. We did this using sliders, this ensured the user could not enter an input that was out of bounds.

11 Rebuilding and testing

Rebuilding the program requires form different libraries and configurations to be made to the project structure and runtime configuration panel. The given system relies on the following libraries.

- Java JDK 14.0.2.
- Javafx 16.
- JUnit 4 and 5.7.0

- maven: org.testfx:testfx-junit5:4.0.16-alpha

The runtime configuration panel requires of adding VM options to add the following favafx modules:

- Javafx.controls
- Javafx.fxml
- Javafx.media

Within the source folder can be found a *tests* folder including three testing classes (TestExercise, TestRunner and TestWorkoutRoutinesConfig). The TestRunner class is prepared to run through all the tests included in the other two classes. It prints their results into a text file dated in the format yyyy-mm-dd_v_x_module_test_report.txt inside the Module Tests folder. If more tests are to be added to the project, it will be possible using the Junit testing framework and including the new testing classes within the TestRunner class.

12 References

[1] CS12120 Group Project 10 - Design Specification
[2] CS12120 Group Project 10 - Test specification
[3] CS12120 Group Project 10 - User Interface Specification
[4] QA Document SE.QA.01 - Quality Assurance Plan

13 Document Change History

Version	*Issue No.*	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
0.2	N/A		03/05/21	Filled the Program Description and Suggestions for Improvement sections	keg21
0.3	N/A		03/05/21	Introduction Rebuilding and testing	jmp16
0.4	N/A		03/05/21	Data Areas and Files	alc72
0.5	N/A		05/05/21	Files and things to look for in future development (xml)	alc72
0.6	N/A		05/05/21	Interfaces	alc72
1.0				Finished suggestion for Improvements, Program Structure, Algorithms and Physical Limitations	keg21