CS22120: Software Engineering Group Project 10

Design Specification

Author: Group 10

Conf. Ref: DesignGroup10_doc

Date: 05/04/2021

Version: 1.3

Status: Released

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 the Design Specification which describes the different system components and how they fit together to accomplish the application requirements.

1.2 Scope

This document provides an accurate translation of the requirements into a clear description of the program components and the relationships between them.

The document should be read by all the group members who have any relation to the Design Specification document [2]. It is assumed the reader is already familiar with the introductory QA document [1] and the Operating Procedures and Configuration Management Standards document [3].

1.3 Objectives

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

- Decomposition description: Programs and modules that make up the system, along with requirements that meet
- Dependency description: Relationships between system components
- Interface description: Information required to use the facilities provided by a module
- Detailed design**:** Further details for the modules whose internal workings are not self-evident).

2 Decomposition description

2.1 Programs in system

The entire workout application will be developed within the homeExcerciseApp directory that can be found in the project repository.

2.2 Significant classes in each program

2.2.1 Significant classes in Program 1

The program will be made up using the following classes, which will be divided into three main categories:

Main functionality:

- **Main**: It initialises the program and contains the main logic to start-up workouts. This class serves as a bridge between the persistent storage using XML and the classes modelling Exercises and Workouts.

- **WorkoutHistory**: Allows us to keep track of the user workout history. It binds some interesting performance facts as the rest time or the number of exercises with the date on which the workout was performed.

Exercises and workouts:

- **Exercise:** This class models a typical exercise with the required parameters. An enumeration will be used to specify to which stage of the workout this exercise can be part of (Warm-up, Working exercises, Cold Down).

- **WorkoutRoutinesConfig:** Workouts with their required parameters will be represented using this class. An enumeration will be used to differentiate them into 4 categories (Beginner, Advanced, LIIT, HIIT and Custom). This classification will be used to attend to the different users described in the User Interface Specification [4].

Persistent storage:

- **XMLio<abstract>:**
  XMLio is intended to be a class that other io classes can extend from. It will contain a single simple function to abstract away the querying process.

- **WorkoutStorage**
  This class will use the function provided by XMLio to load information required by Exercise and WorkoutRoutineConfig.

- **WorkoutRead**
  This class will use the function provided by XMLio to get information about previously completed workouts.

User interface:

- **CustomWorkoutController**
  This class allows user to create their own custom workout.

- **ExerciseOverviewController**
  Class to display an overview from the exercise chosen.

- **HomeController**
  Class to display the home screen user interface.

- **PreconfiguredWorkoutsController**
  Class to allow to user to select preconfigured workouts.

- **WorkoutController**
  Class to run the workout.

2.3 Mapping from requirements to classes

Functional Requirements	*Classes*
FR 1 | Start-up	Application.
FR 2 | Configuring a set of exercises	Application, WorkoutRead, XMLio & WorkoutRoutineConfig.

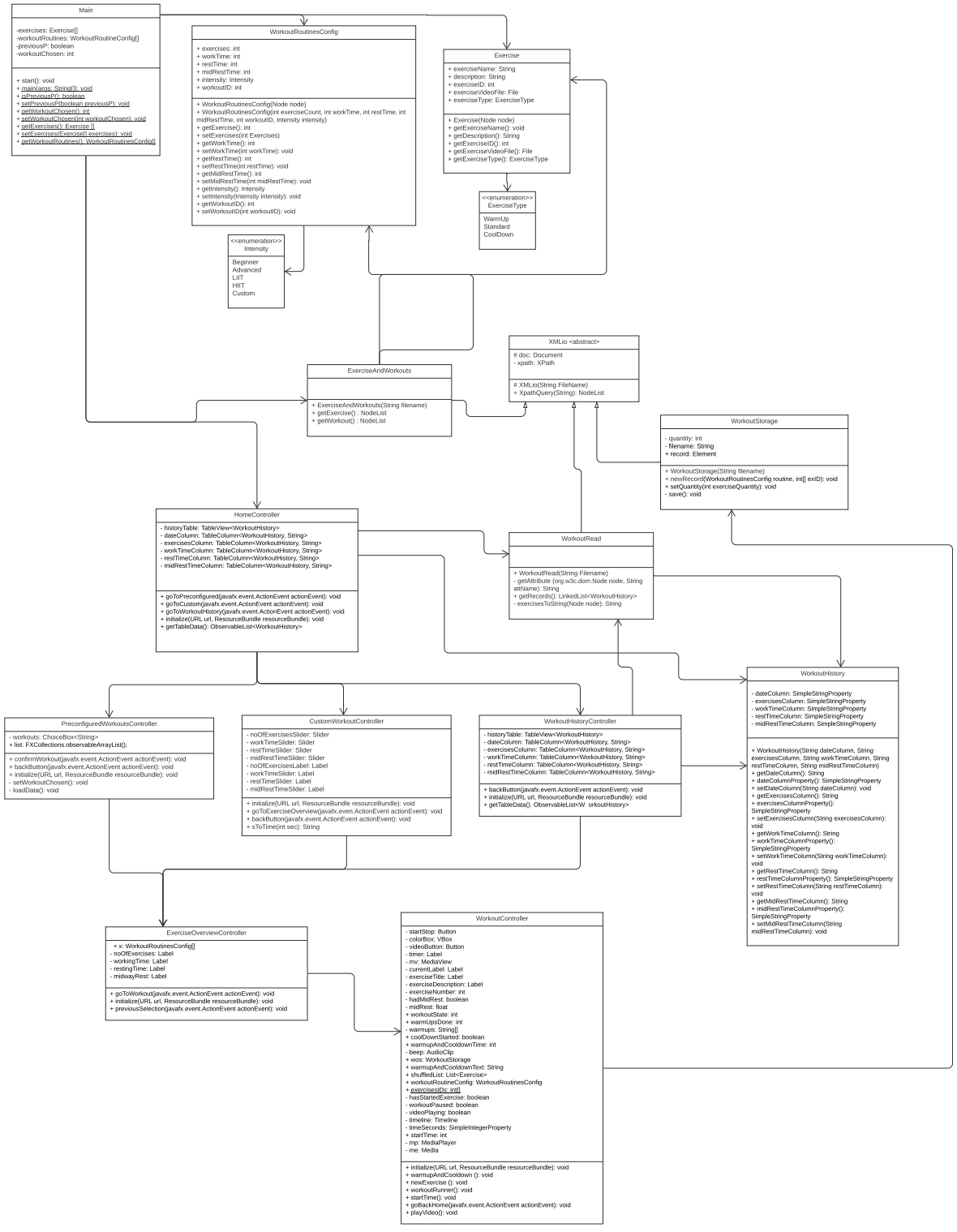
FR 3 | Options after selecting a set of exercises	Application, WorkoutRead & XMLio.
FR 4 | Warm up	Application, Exercise & WorkoutActivity.
FR 5 | Carrying out exercises	Application, Exercise, WorkoutActivity & WorkoutRead.
FR 6 | Cool down	Application, Exercise & WorkoutActivity.
FR 7 | Exercise guidance	Exercise.
FR 8 | Timing and progress guidance	Application & WorkoutRoutineConfig.
FR 9 | Pausing	Application.
FR 10 | Record Keeping	WorkoutRead & XMLio.

3 Dependency description

3.1 UML Diagram

3.1.1 UML Diagram for program 1

CS22120: Group project - Class Diagram



4 Interface description

4.1 Workout History Interface

Access: Public

Methods:

public void WorkoutActivity()
This is a constructor method

public void warmUp()
This method starts the warmup routine.

public void runWk()
This method starts the main workout routine.

public void coolDown()
This method starts the cooldown routine.

4.2 Workout Routine Config Interface

Access: Public

Methods:

public WorkoutRoutineConfig()
This is a constructor method, takes a node to populate instance variables.

public String toString()
A toString method used for test printing.

public getter()
Method to return values of variables in the class.

public setter()
Method to set values of variables in the class.

4.3 Exercise Interface

Access: Public

Extended Classes: No extended classes, this class is used to hold all the relevant information for each exercise. Used to allow the app class to access exercise information easily.

Methods:

public v Exercise(Node node)
This is a constructor method. Uses an exercise node to populate instance variables.

public String getExerciseName()
This method will return the name of the exercise.

public String getDescription()
This method will return the exercise description.

public File getExerciseVideoFile()
This method will return the exercise video file.

public ExerciseType getExerciseType()
This method will return the exercise type.

public int getExerciseID()
This method will return the exercise ID.

4.4 Workout Read Interface

Access: Public

Extended Classes: XMLio

Methods:

public WorkoutRead(String FileName)
Calls XMLio constructor.

public LinkedList<WorkoutHistory> getRecords()
Returns list of exercises for a given month and year.
This will be used for the drop-down selection.

private String getAttribute()
Returns information about a singular Record, including names of workouts performed.
This will be used for looking at a specific exercise.

4.5 Workout Storage Interface

Access: **Public**

Extended Classes: XMLio, it permits to save the data related to the workout history into an structured XML document.

Methods:

public void WorkoutStorage(filename)
Constructor of the class. It requires the filename where the data will be stored.

public void ExerciseSave(array exerciseListForWO)
It saves to the XML document the information already available when initialising a workout (exercises planned and rest times)

Public void incQuantity()
This method will be called once per rest period. It allows the system to modify a variable within the XML document responsible for keeping track of the number of exercises completed by the user.

4.6 App Interface

Access: Public

Extended Classes: No extended classes, this class contains the main logic to populate a workout with random exercises and is used to bridge the gap between the persistent storage XML and the classes which model workouts.

Methods:

public void runWk()
This is a method used to start a workout once the user has decided on either a preconfigured workout or configured their own workout.

public void pauseWk()
Method used to allow the user to pause the workout at any time after it has started.

Public void resumeWk()
Method used to allow the user to resume a workout anytime after it has been paused.

Public void exerciseCreation()
This method is used to populate a workout routine with however many random exercises specified by the user when configuring their workout. This method will use the Random class for randomising the exercises.

Public void workoutConfigCreation()
This method uses the workoutRoutinesConfig class to load preconfigured workouts to the application class so they can be run by the user.

4.7 XMLio Interface

Access: Public

Extended Classes:
A class designed to be inherited by all .XML input/output classes.
Includes a function to query a document that is read in on construction.

Methods:

Public void XMLio(String filename)
Constructor. This takes a directory as a string for the file to be read or written. DocumentBuilder and XPathFactory libraries and corresponding instance variables are set up for use.

Private nodeList xPathQuery(String xPath)
Queries the file with the xPath parameter given. It uses a single *XPath.evaluate()*, however it is nice to have for abstraction purposes.

4.8 ExerciseAndWorkouts

Access: Public

Extended Classes: XMLio

Methods:

Public ExerciseAndWorkouts()
This is a constructor method.

Public NodeList getExercise()
This is a getter method used to access exercises from the XML document and load to the Exercise class.

Public NodeList getWorkout()
This is a getter method used to access preconfigured workouts from the XML document and load to the WorkoutRoutineConfig class.

4.9 CustomWorkoutController

Access: Public

Extended Classes:

Methods:

public void initialize(URL url, ResourceBundle resourceBundle)
Method used to initialize the Controller. It uses url parameter as a location to resolve relative paths for the root object, or null if the location is not known. ResourceBundle is used to localize the root object, or null if the root object was not localized.

Public void goToExerciseOverview(javafx.event.Action actionEvent)
Method to go to the exercise overview scene. Throws IOExceotion if the file cannot be found.

public void backButton(javafx.event.ActionEvent actionEvent)
Method to go back to the previous scene. Throws IOException if the file cannot be found.

public String sToTime(int sec)
Method to return seconds and minutes and seconds. Converts a given integer representing second into a string as minutes and seconds.

4.10 ExerciseOverviewController

Access: Public

Extended Classes:

Methods:

public String sToTime(int sec)
Method to return seconds and minutes and seconds. Converts a given integer representing second into a string as minutes and seconds.

public void goToWorkout(javafx.event.ActionEvent actionEvent)
Button functionality to load the workout page. Throws IOException if the file cannot be found.

public void initialize(URL url, ResourceBundle resourceBundle)
Method used to initialize the Controller. It uses url parameter as a location to resolve relative paths for the root object, or null if the location is not known. ResourceBundle is used to localize the root object, or null if the root object was not localized.

public void previousSelection(javafx.event.ActionEvent actionEvent)
Button functionality to load either the preconfigured workouts page or the custom workouts page. Throws IOException if the file cannot be found.

4.11 HomeController

Access: Public

Extended Classes:

Methods:

public void goToPreconfigured(javafx.event.ActionEvent actionEvent)
Button functionality to load the preconfigured workouts page. Throws IOException if the file cannot be found.

public void goToCustom(javafx.event.ActionEvent actionEvent)
Button functionality to load the custom workouts page. Throws IOException if the file cannot be found.

public void goToWorkoutHistory(javafx.event.ActionEvent actionEvent)
Button functionality to load the workout history page. Throws IOException if the file cannot be found.

public void initialize(URL url, ResourceBundle resourceBundle)
Method used to initialize the Controller. It uses url parameter as a location to resolve relative paths for the root object, or null if the location is not known. ResourceBundle is used to localize the root object, or null if the root object was not localized.

public ObservableList<WorkoutHistory> getTableData()
Generates and returns a list of workout histories to be read into java fx tables.

4.12 PreconfiguredWorkoutsController

Access: Public

Extended Classes:

Methods:

public void confirmWorkout(javafx.event.ActionEvent actionEvent)

public void backButton(javafx.event.ActionEvent actionEvent)
Returns to the previous screen. Throws IOException if the file cannot be found.

public void initialize(URL url, ResourceBundle resourceBundle)
Method used to initialize the Controller. It uses url parameter as a location to resolve relative paths for the root object, or null if the location is not known. ResourceBundle is used to localize the root object, or null if the root object was not localized.

private void setWorkoutChosen()
Sets chosen workout.

private void loadData()

Loads the descriptions for the preconfigured workouts.

4.13 Workout Controller

Access: Public

Extended Classes:

Methods:

public void initialize(URL url, ResourceBundle resourceBundle)
Method used to initialize the Controller. It uses url parameter as a location to resolve relative paths for the root object, or null if the location is not known. ResourceBundle is used to localize the root object, or null if the root object was not localized.

public void warmupAndCooldown()
Method to load path to media for warmups and cooldowns.

public void newExercise()
Method to load path to media for exercises.

public void workoutRunner()
Method used to play the correct part of the workout routine.

public void startTime()
Method used to display the timer for the workout.

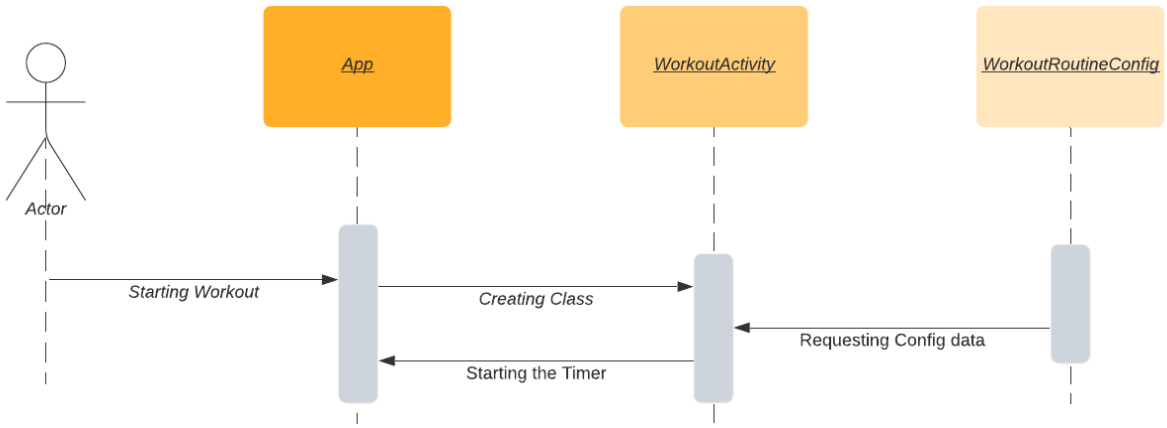
public void goBackHome(javafx.event.ActionEvent actionEvent)
Method use to go back to the home screen. Throws IOException if the file cannot be found.

public void playVideo()
Method to play and pause the exercise video.

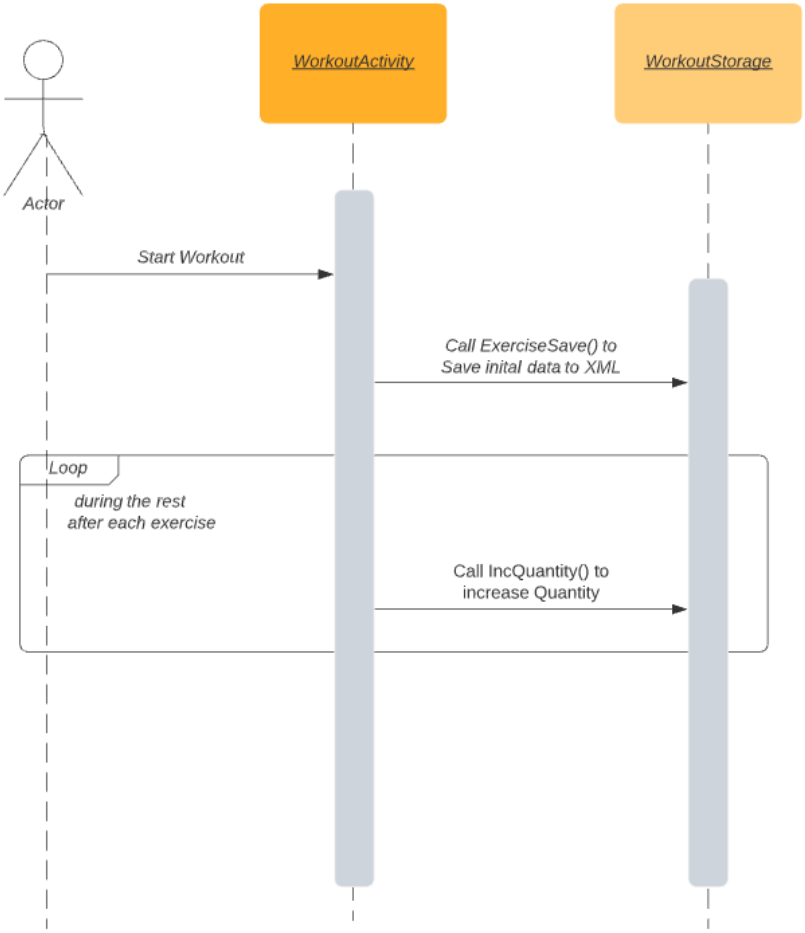
5 Detailed design

5.1 Sequence diagrams

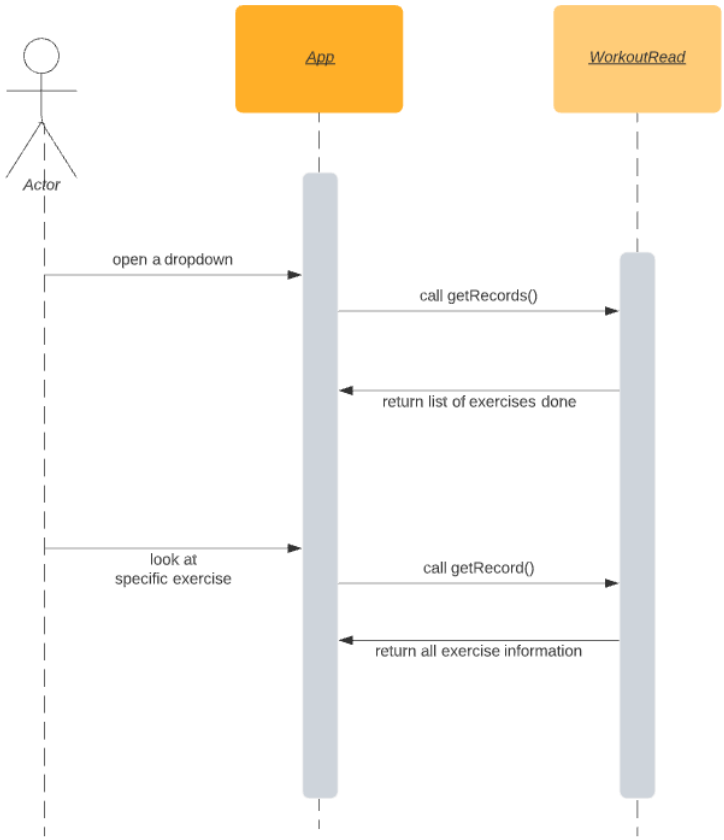
Sequence for starting a workout:



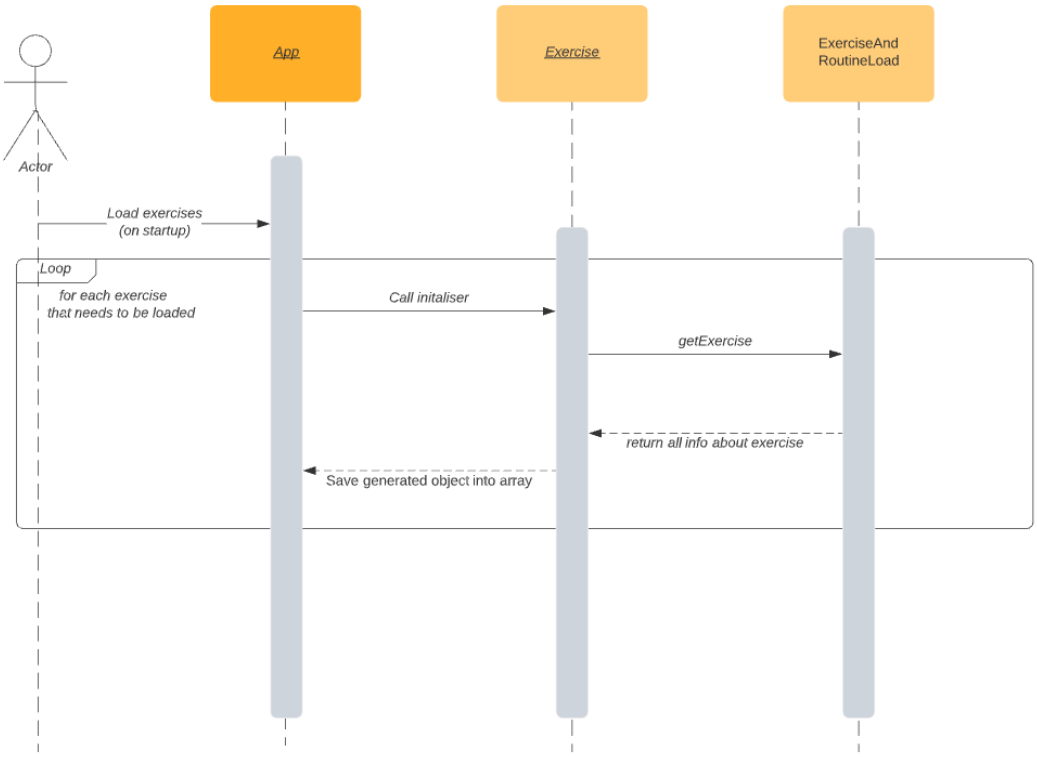
Saving information to File:



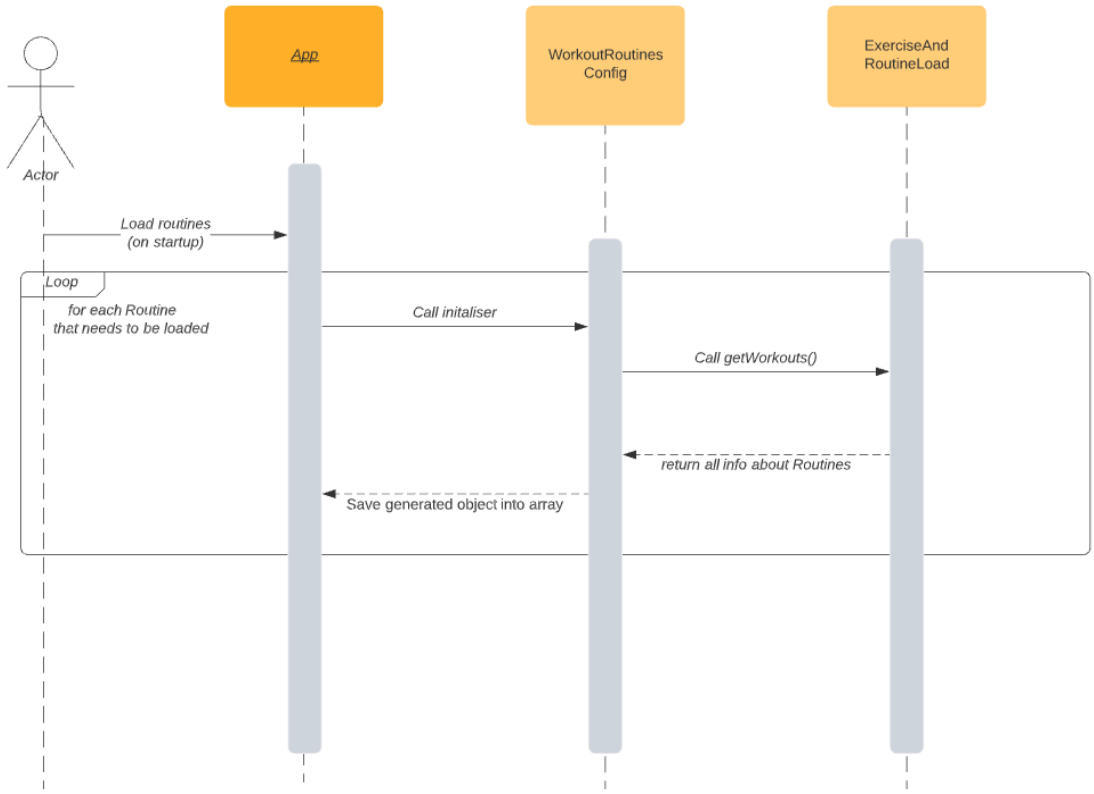
Loading workout history



Loading exercises from storage



Loading workout configs from storage



5.2 Significant algorithms

Random selection of exercises

Algo for random exercise in pseudo / structural code. (need to use random library in java)

1. Workout Starts
2. Randomize the nodelist of exercises.
3. Iterate through nodelist (or array) of exercises.

5.3 Significant data structures

NodeList:

While it is not going to be actively used, NodeList is an important data structure to mention. The primary way of querying the XML file is through the function *XPath.evaluate*, and parse the resulting NodeList into a more readable format.

Exercise and WorkoutRoutineConfig:

Exercise and WorkoutRoutineConfig are planned to be a struct-like classes, used to hold information. These will be put into an array in *App* for easy access.

Date:

obtaining the date will be necessary as it's part of the information to be saved in each record. This is likely going to be done through the use of a library and any data structures that come with it.

6 References

[1] QA Document SE.QA.01 – Quality Assurance Plan.
[2] QA Document SE.QA.05 – Design Specification Standards.
[3] QA Document SE.QA.08 – Operating Procedures and Configuration Management Standards.
[4] Software Engineering Group Project 10 - User Interface Specification

7 Document Change History

Version	*Issue No.*	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
0.1		N/A	23/02/21	keg21Created layout; Intro section completed.	jmp16
0.2		N/A	10/03/21	Intro modified; Decomposition description nearly done.	jmp16
0.3		N/A	13/03/21	Converted document to google docs and added interface sections.	ahz1
0.4	14	N/A	15/03/21	Added the Exercise interface specification section.	keg21
0.5		N/A	16/03/21	Added Component diagram Created 2.4 Table	isl7/ahz1
0.6	18, 20	N/A	17/03/21	Added App class and ExerciseAndWorkoutsLoad class interface specification sections.	keg21
0.7		N/A	18/03/21	Added 5.3 Significant data structures info	alc72
0.8		N/A	18/03/21	Added sequence diagram for starting a workout	isl7
0.9		N/A	19/03/21	Added signatures to methods	isl7
0.10		N/A	20/03/21	Added 2.2 Persistent storage	alc72
0.11	38, 39, 40, 44	N/A	24/03/21	Fixed header, capitalisation and numbering of headings	keg21
1.00	28,29,30	N/A	01/04/21	Added 5.1 Sequence diagrams	Alc72
1.1	43	N/A	26/04/21	Fixed spacings	pid8
1.2		N/A	04/05/21	Updating classes and methods	pid8
1.3		N/A	05/05/21	Reformatting	Keg21