# System design document for Ripped

## Contents

# Introduction

Ripped is an Android application which will let you log your exercises from the gym. You can create your own workouts with exercises already from the database or create your own exercises.

## Design goals

An application with good looking and uniform design so that the look does not differ depending on which android version the user has.
Easy way of handling workouts and exercises, editing workouts (even during a workout) and easy to input results.

## Definitions, acronyms and abbreviations

**WorkoutTemplate**
A workouttemplate is a routine where the user can add and/or remove the exercises that would be performed during the visit to the gym.

**Workout**
A workout is a created when the user has performed a workouttemplate. A workout is used to display the history.

**Exercise**
Something the user will be doing at the gym. There is three types; cardio, dynamic and static.

**Cardio exercise**
A cardio exercise could be running, row or cycling. Will allow the user to unput the duration of the exercise and the distance.

**Dynamic exercise**
An exercis where the user can enter number of repetitions and the weight. For example, bench press, military press or dead lift.

**Static exercise**
An exercise such as the plank bridge or side bridge. This type of exercise will allow the user to input duration and weight.

**Reps**
Short for repetitions.

**Sets**
A set contains either the repetition and weight, time and weight or time and distance. So when the user has finished his bench press session of 10 reps with X kg, that will complete a set. The next number of reps and weight X a new set has been performed.

### References
N/A

## Proposed system architecture
The android application will show both some example exercises and workouts, and also the exercises and workouts created by the user.
This data will be fetched from a local content provider (SQLite Database).
The main design pattern will be the model-view-controller.

### Software decomposition

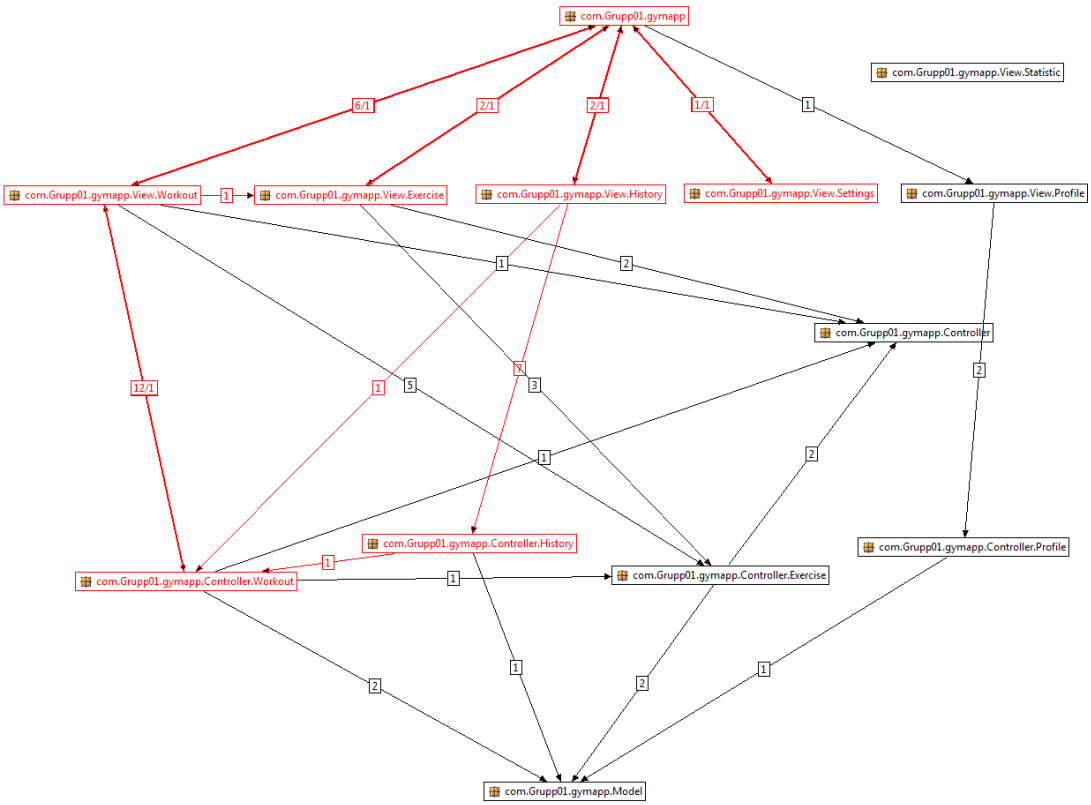
### General
The application is divided up in three parts which is divided up in more sub parts.
The three major parts is Model, Controller and View.
The view is what the user will see and use. The view is divided up in more subparts, one for each part of the GUI.
The model is our local SQLite database which contains several tables for storing workouts workouttemplates, exercises, sets, profile data etc.
The controller is also divided up in sub parts depending on which GUI will use them. The controller lets the view parts to write and get data to and from the database.

### Tiers
Only the application.

### Communication
No communication from the phone to the outside world

### Layering (anpassas efter indelning, tex MVC)
The application is divided up in three layers and is designed with the MVC model in mind.
The model is our local SQLite database.
The controller is classes with method that will access the database and return the data.
The view is the applications activities (and some help classes) and is what the user will see. The view will fetch data from the model via the access methods in the controller classes.

## Dependency analysis




## Concurrency issues

Everything runs in the same thread, since there is not so much data that is fetched from the database there is no problem so far. However, there may a need to move that to a separate thread if we notice that there is a big delay between opening an activity and the data is displayed in the activity.

## Issue handling

As soon as an issue or a bug has been discovered, it has been added to ‘Issues’ in our github. Everyone in the group has regularly been checking this and if someone nows how to solve the issue, they have created a new branch, fixed the issue and then merged the branch into the master.

## Persistent data management

The user selected settings (such as language and units for weight an distance) is managed by Android default way of handle this. By creating a SettingsActivity the system it self will manage these in the SharedPreferences storage.
All other information, such as workout templates, exercises, sets for exercises etc is stored in a local SQLite database.

## Access control and security

Since all the activity in the application is on the local phone (no central server) and there is no sensitive information beeing stored there is no access control or security implemented.

## Boundary conditions

**N/A**

## References

developer.android.com
actionbarsherlock.com/

# Build Procedure

Branches which are finished are merged into the master branch. When test cases is finished we compile it (using Eclipse) and bugs is reported as mentioned above.

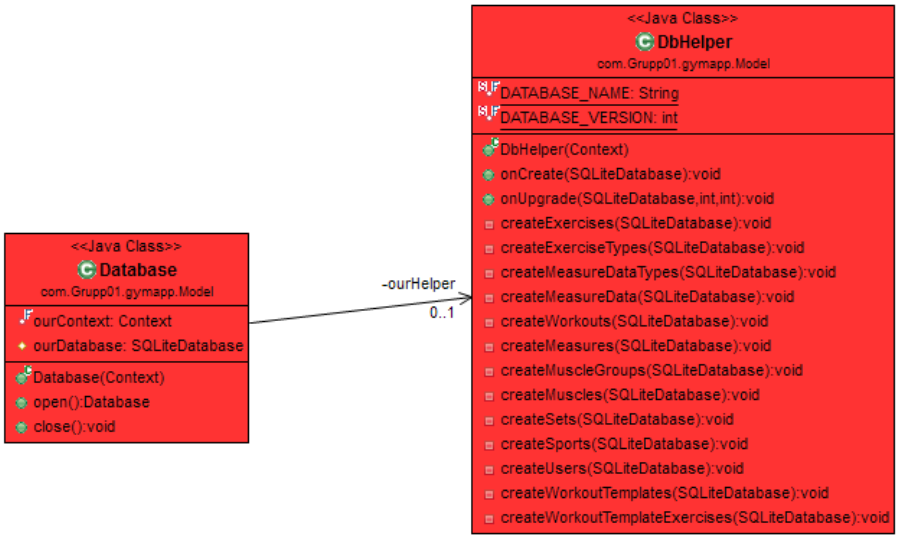
# Release Procedure

Before implementing any new feature or a bugfix, the feature has had its own branch, where all the developing has taken place. Before merging this feature or bugfix into the master branch it should pass Junit, robotium and accaptence tests, after that the branch was built and pushed to git together with a document containing releasnotes.

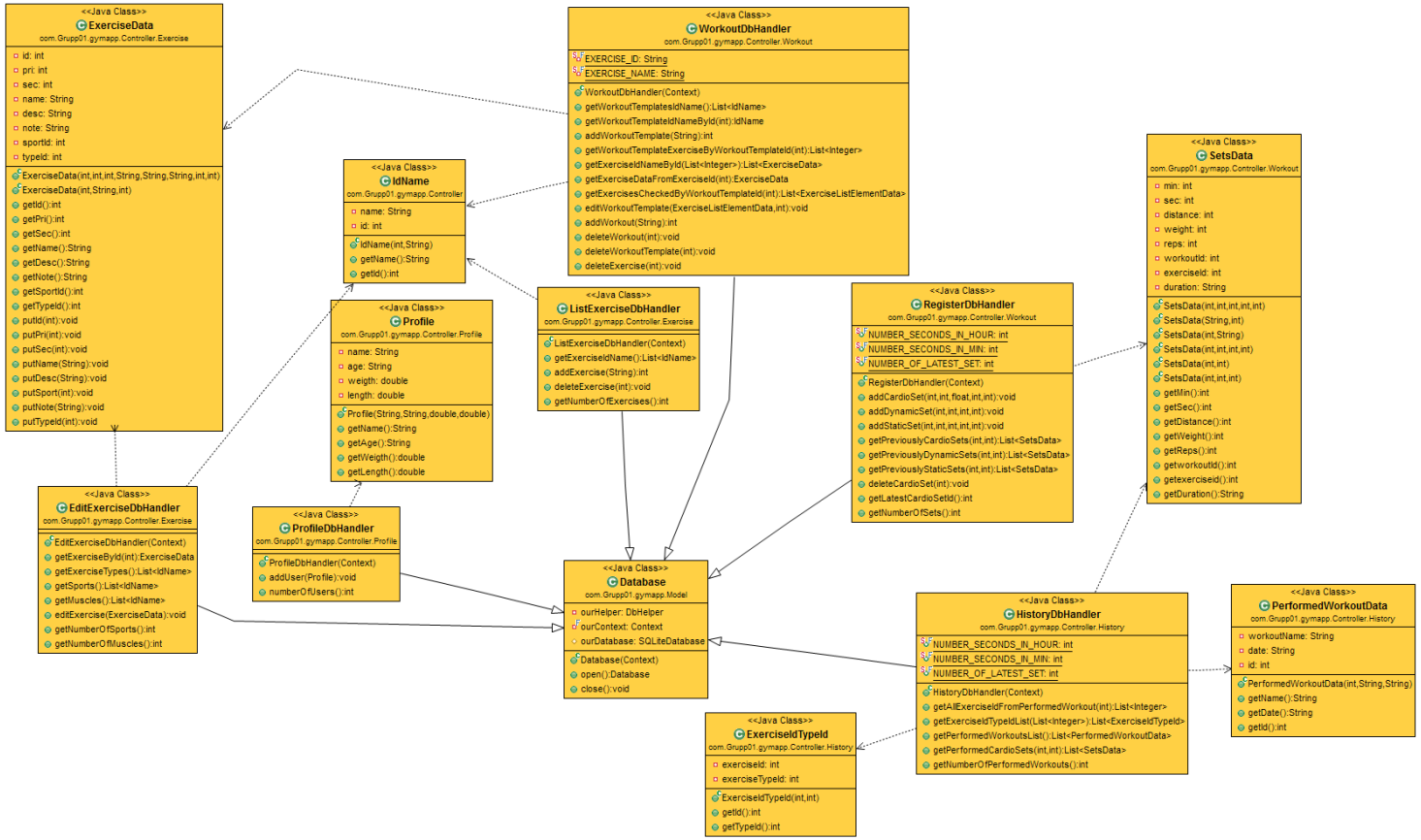
# Apendix A

## UML Diagrams

### Model



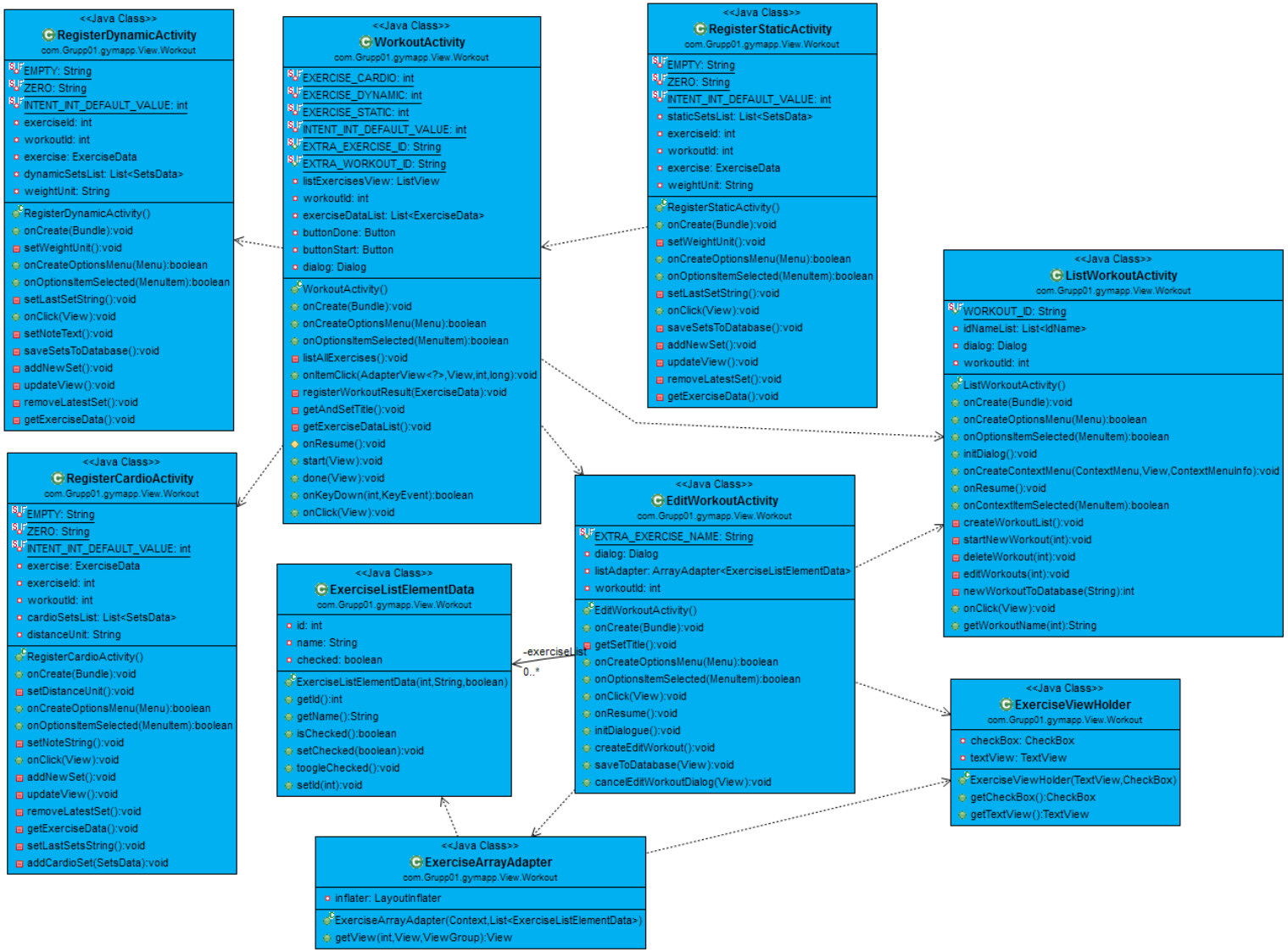
## Controller


<<Java Class>>
ExerciseData
com.Grupp01.gymapp.Controller.Exercise
id: int
pri: int
sec: int
name: String
desc: String
note: String
sportId: int
typeId: int
ExerciseData(int,int,int,String,String,String,int,int)
ExerciseData(int,String,int)
getId():int
getPri():int
getSec():int
getName():String
getDesc():String
getNote():String
getSportId():int
getTypeId():int
putId(int):void
putPri(int):void
putSec(int):void
putName(String):void
putDesc(String):void
putSport(int):void
putNote(String):void
putTypeId(int):void
<<Java Class>>
WorkoutDbHandler
com.Grupp01.gymapp.Controller.Workout
EXERCISE_ID: String
EXERCISE_NAME: String
WorkoutDbHandler(Context)
getWorkoutTemplatesIdName():List<IdName>
getWorkoutTemplateIdNameById(int):IdName
addWorkoutTemplate(String):int
getWorkoutTemplateExerciseByWorkoutTemplateId(int):List<Integer>
getExerciseIdNameById(List<Integer>):List<ExerciseData>
getExerciseDataFromExerciseId(int):ExerciseData
getExercisesCheckedByWorkoutTemplateId(int):List<ExerciseListElementData>
editWorkoutTemplate(ExerciseListElementData,int):void
addWorkout(String):int
deleteWorkout(int):void
deleteWorkoutTemplate(int):void
deleteExercise(int):void
<<Java Class>>
IdName
com.Grupp01.gymapp.Controller
name: String
id: int
IdName(int,String)
getName():String
getId():int
<<Java Class>>
SetsData
com.Grupp01.gymapp.Controller.Workout
min: int
sec: int
distance: int
weight: int
reps: int
workoutId: int
exerciseId: int
duration: String
SetsData(int,int,int,int,int)
SetsData(String,int)
SetsData(int,String)
SetsData(int,int,int,int)
SetsData(int,int)
SetsData(int,int,int)
getMin():int
getSec():int
getDistance():int
getWeight():int
getReps():int
getworkoutId():int
getexerciseId():int
getDuration():String
<<Java Class>>
Profile
com.Grupp01.gymapp.Controller.Profile
name: String
age: String
weigth: double
length: double
Profile(String,String,double,double)
getName():String
getAge():String
getWeigth():double
getLength():double
<<Java Class>>
ListExerciseDbHandler
com.Grupp01.gymapp.Controller.Exercise
ListExerciseDbHandler(Context)
getExerciseIdName():List<IdName>
addExercise(String):int
deleteExercise(int):void
getNumberOfExercises():int
<<Java Class>>
RegisterDbHandler
com.Grupp01.gymapp.Controller.Workout
NUMBER_SECONDS_IN_HOUR: int
NUMBER_SECONDS_IN_MIN: int
NUMBER_OF_LATEST_SET: int
RegisterDbHandler(Context)
addCardioSet(int,int,float,int,int):void
addDynamicSet(int,int,int,int):void
addStaticSet(int,int,int,int,int):void
getPreviouslyCardioSets(int,int):List<SetsData>
getPreviouslyDynamicSets(int,int):List<SetsData>
getPreviouslyStaticSets(int,int):List<SetsData>
deleteCardioSet(int):void
getLatestCardioSetId():int
getNumberOfSets():int
<<Java Class>>
EditExerciseDbHandler
com.Grupp01.gymapp.Controller.Exercise
EditExerciseDbHandler(Context)
getExerciseById(int):ExerciseData
getExerciseTypes():List<IdName>
getSports():List<IdName>
getMuscles():List<IdName>
editExercise(ExerciseData):void
getNumberOfSports():int
getNumberOfMuscles():int
<<Java Class>>
ProfileDbHandler
com.Grupp01.gymapp.Controller.Profile
ProfileDbHandler(Context)
addUser(Profile):void
numberOfUsers():int
<<Java Class>>
Database
com.Grupp01.gymapp.Model
ourHelper: DbHelper
ourContext: Context
ourDatabase: SQLiteDatabase
Database(Context)
open():Database
close():void
<<Java Class>>
HistoryDbHandler
com.Grupp01.gymapp.Controller.History
NUMBER_SECONDS_IN_HOUR: int
NUMBER_SECONDS_IN_MIN: int
NUMBER_OF_LATEST_SET: int
HistoryDbHandler(Context)
getAllExerciseIdFromPerformedWorkout(int):List<Integer>
getExerciseIdTypeIdList(List<Integer>):List<ExerciseIdTypeId>
getPerformedWorkoutsList():List<PerformedWorkoutData>
getPerformedCardioSets(int,int):List<SetsData>
getNumberOfPerformedWorkouts():int
<<Java Class>>
PerformedWorkoutData
com.Grupp01.gymapp.Controller.History
workoutName: String
date: String
id: int
PerformedWorkoutData(int,String,String)
getName():String
getDate():String
getId():int
<<Java Class>>
ExerciseIdTypeId
com.Grupp01.gymapp.Controller.History
exerciseId: int
exerciseTypeId: int
ExerciseIdTypeId(int,int)
getId():int
getTypeId():int

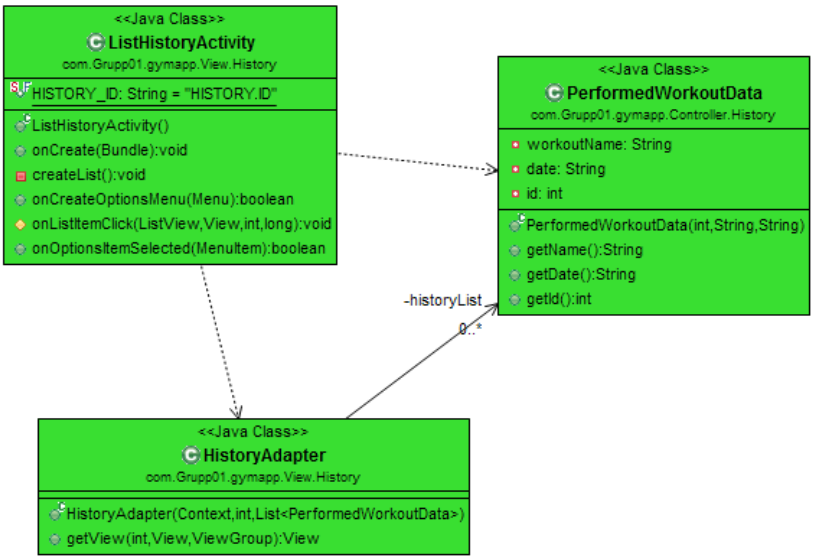
## View

### *Exercises*

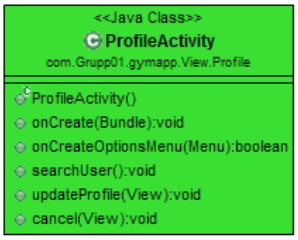
## *Workout*


<<Java Class>>
RegisterDynamicActivity
com.Grupp01.gymapp.View.Workout
EMPTY: String
ZERO: String
INTENT_INT_DEFAULT_VALUE: int
exerciseId: int
workoutId: int
exercise: ExerciseData
dynamicSetsList: List<SetsData>
weightUnit: String
RegisterDynamicActivity()
onCreate(Bundle):void
setWeightUnit():void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
setLastSetString():void
onClick(View):void
setNoteText():void
saveSetsToDatabase():void
addNewSet():void
updateView():void
removeLatestSet():void
getExerciseData():void
<<Java Class>>
WorkoutActivity
com.Grupp01.gymapp.View.Workout
EXERCISE_CARDIO: int
EXERCISE_DYNAMIC: int
EXERCISE_STATIC: int
INTENT_INT_DEFAULT_VALUE: int
EXTRA_EXERCISE_ID: String
EXTRA_WORKOUT_ID: String
listExercisesView: ListView
workoutId: int
exerciseDataList: List<ExerciseData>
buttonDone: Button
buttonStart: Button
dialog: Dialog
WorkoutActivity()
onCreate(Bundle):void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
listAllExercises():void
onItemClick(AdapterView<?>,View,int,long):void
registerWorkoutResult(ExerciseData):void
getAndSetTitle():void
getExerciseDataList():void
onResume():void
start(View):void
done(View):void
onKeyDown(int,KeyEvent):boolean
onClick(View):void
<<Java Class>>
RegisterStaticActivity
com.Grupp01.gymapp.View.Workout
EMPTY: String
ZERO: String
INTENT_INT_DEFAULT_VALUE: int
staticSetsList: List<SetsData>
exerciseId: int
workoutId: int
exercise: ExerciseData
weightUnit: String
RegisterStaticActivity()
onCreate(Bundle):void
setWeightUnit():void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
setLastSetString():void
onClick(View):void
saveSetsToDatabase():void
addNewSet():void
updateView():void
removeLatestSet():void
getExerciseData():void
<<Java Class>>
ListWorkoutActivity
com.Grupp01.gymapp.View.Workout
WORKOUT_ID: String
idNameList: List<IdName>
dialog: Dialog
workoutId: int
ListWorkoutActivity()
onCreate(Bundle):void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
initDialog():void
onCreateContextMenu(ContextMenu,View,ContextMenuInfo):void
onResume():void
onContextItemSelected(MenuItem):boolean
createWorkoutList():void
startNewWorkout(int):void
deleteWorkout(int):void
editWorkouts(int):void
newWorkoutToDatabase(String):int
onClick(View):void
getWorkoutName(int):String
<<Java Class>>
RegisterCardioActivity
com.Grupp01.gymapp.View.Workout
EMPTY: String
ZERO: String
INTENT_INT_DEFAULT_VALUE: int
exercise: ExerciseData
exerciseId: int
workoutId: int
cardioSetsList: List<SetsData>
distanceUnit: String
RegisterCardioActivity()
onCreate(Bundle):void
setDistanceUnit():void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
setNoteString():void
onClick(View):void
addNewSet():void
updateView():void
removeLatestSet():void
getExerciseData():void
setLastSetsString():void
addCardioSet(SetsData):void
<<Java Class>>
EditWorkoutActivity
com.Grupp01.gymapp.View.Workout
EXTRA_EXERCISE_NAME: String
dialog: Dialog
listAdapter: ArrayAdapter<ExerciseListElementData>
workoutId: int
EditWorkoutActivity()
onCreate(Bundle):void
getSetTitle():void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
onClick(View):void
onResume():void
initDialogue():void
createEditWorkout():void
saveToDatabase(View):void
cancelEditWorkoutDialog(View):void
<<Java Class>>
ExerciseListElementData
com.Grupp01.gymapp.View.Workout
id: int
name: String
checked: boolean
ExerciseListElementData(int,String,boolean)
getId():int
getName():String
isChecked():boolean
setChecked(boolean):void
toogleChecked():void
setId(int):void
-exerciseList
0..*
<<Java Class>>
ExerciseViewHolder
com.Grupp01.gymapp.View.Workout
checkBox: CheckBox
textView: TextView
ExerciseViewHolder(TextView,CheckBox)
getCheckBox():CheckBox
getTextView():TextView
<<Java Class>>
ExerciseArrayAdapter
com.Grupp01.gymapp.View.Workout
inflater: LayoutInflater
ExerciseArrayAdapter(Context,List<ExerciseListElementData>)
getView(int,View,ViewGroup):View

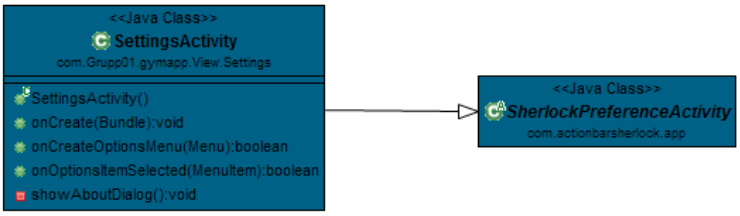
## *History*




## *Profile*



### *Settings*


<<Java Class>>
SettingsActivity
com.Grupp01.gymapp.View.Settings
SettingsActivity()
onCreate(Bundle):void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
showAboutDialog():void
<<Java Class>>
SherlockPreferenceActivity
com.actionbarsherlock.app