**PROG24178** Project Proposal

Project Name: GoFit

Group: codeWizards

UX Developer, QA, Deployment: Chris Sarvghadi

I/O Developer, Tester, Debugger: Alex Park

March 25, 2015

**Project Specifications**

Our application is designed to track the progress of a user’s strength training fitness regimen. There are a number of fitness applications that already exist, which track a user’s fitness progress, or which allow them to calculate their strength progression, but we have not seen one which combine both of these ideas into a single application. Furthermore we hope to incorporate a graphing function which will take the user’s inputs and visually display their progression in various exercises. Depending on time constraints we may also add further functionality to input (and graph) a user’s bodyweight and body-fat changes over time, and to track a user’s cardiovascular fitness regimen.

The users for this application are individuals who wish to track their fitness progression using a simple and clean interface design, and see their progress displayed in a convenient and informative graph format. Our intended users will not need to be particularly tech-savvy or even very knowledgeable about fitness – we aim to give them a lot of information (graphs, one-rep-max, caloric expenditure, etc.) based on limited inputs (daily workouts). Users can choose to use this information to modify their training regimen, or they may simply use the application to keep track of the work that they have done.

**Screen Discussion**

Left-hand navigation pane allows switching between various tabs (CardLayout)

Welcome/Main Page:

* Create button: pops up a new SubGui window to create a new user account.
* Open button: pops up a JFileChooser window to allow loading of an existing account.

Create Account Page (SubGui):

* Name/Age/Sex JTextfields: allow user to input information about themselves for creation of a new account.

Entry Page:

* Body Part JComboBox: allows user to choose a body part, which will determine the exercises available in the next JComboBox.
* Exercise JComboBox: allows the user to choose an exercise (ArrayList) to load into the application.
* Date/Weight/Repetitions/Calories JTextField: allows user to input data about their exercise.
* Calculate JButton: uses the previous JTextFields to calculate the user’s One-Rep Maximum, and how much they can likely lift at various percentages of that maximum.
* Add/Edit JButtons: allow the user to add their inputs into their log, or to edit/delete previous entries.

Log Page:

* Exercise JComboBox: allows user to specify an exercise, of which logs will be displayed in a columned JList format.
* Edit JButton: will switch back to the Entry page and load the selected log entry.
* Save JButton: will save the new entries and/or edits to the user’s account file.

Graph Page:

* Exercise JComboBox: shows a list of exercises that the user has performed and constructs a graph based on this information.

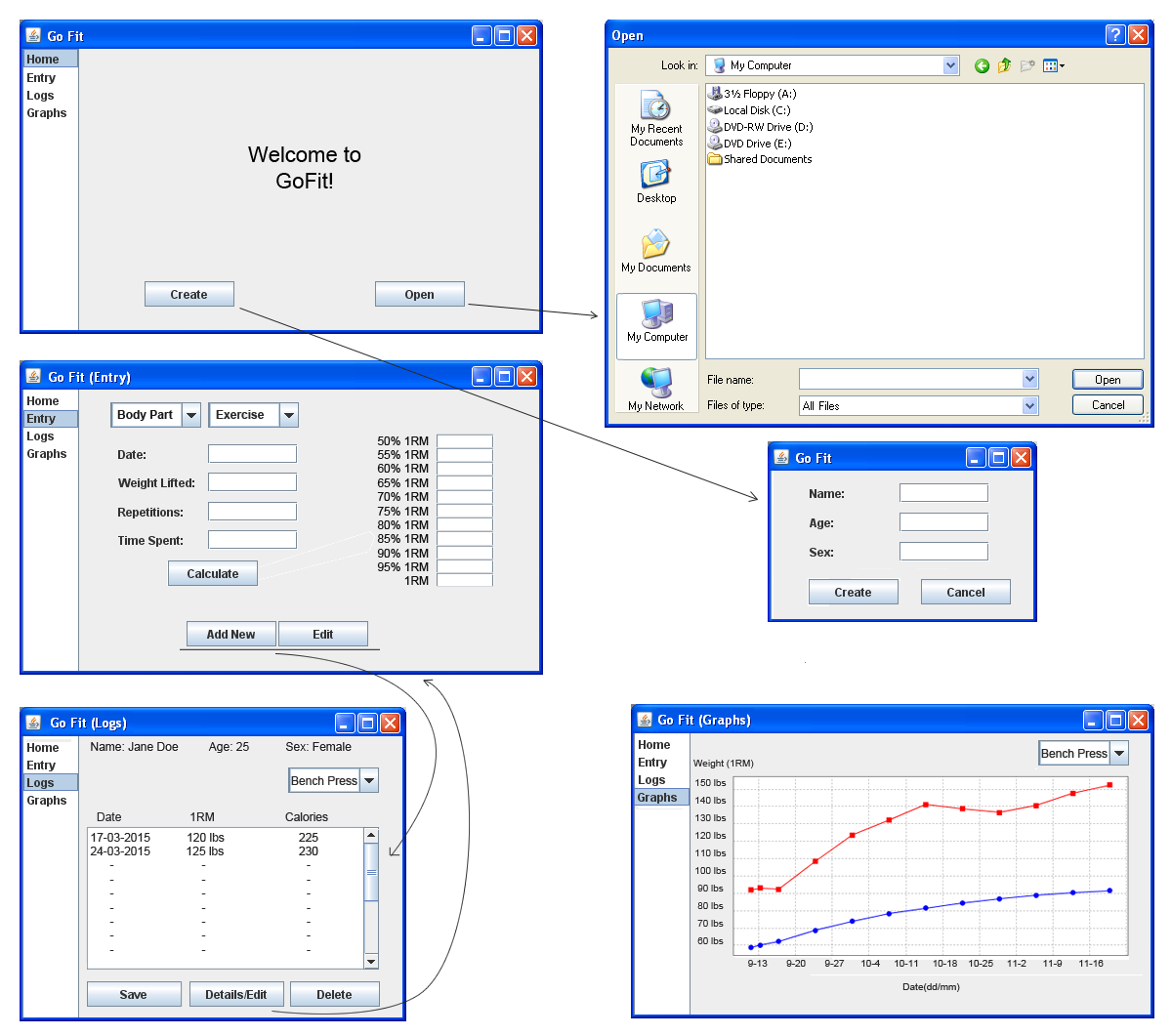
**OOP Discussion**

The object model will be divided into two distinct parts. The first part will involve the code necessary to instantiate the GUI. The MainGui class will use the CardLayout layout to allow convenient switching between the various tabs (Home, Entry, Logs, Graphs). The MainGui will also contain an instance of the SubGui class in order to implement pop-up menus (which will be used to create new user accounts). The MainGui class and SubGui class will have a hasA/composition relationship.

The MainGui and SubGui implement the ValidateInput interface to validate user inputs and ensure that there are no errors. The CreateAccount interface will be implemented by the MainGui class to allow the SubGui to pass variable information to the MainGui class when a user wishes to create a new account.

The second part will contain the programming logic necessary to run the application. The UserAccount class contains private variables of name, age, sex, and an ArrayList of Exercise(s). The Exercise class will extend ArrayList and contain instances of the Log class. The UserAccount and Exercise classes, and the Exercise and Log classes, both have a hasA/composition relationship. Also, the Exercise class inherits from the ArrayList class.

UI Design MockUp



Unified Modeling Language

|  |
| --- |
| *<<interface>>CreateAccount* |
| *+ onUpdate (name : String, age : int, sex : char,*  *selectedField : String[])* |

|  |
| --- |
| SubGui |
| - createAcc : CreateAccount |
| + SubGui() |

|  |
| --- |
| MainGui |
| - cards : CardLayout  - account : Account  - chart : JFreeChart  - index : int  - currentField : Field  - editMode : boolean  - subGui : SubGui |
| + MainGui ()  - open () : void  - addNew () : void  - edit () : void  - delete () : void  - save () : void |

|  |
| --- |
| *<<interface>>ValidateInput* |
| *+ validate () : boolean*  *+ reset () : void* |

|  |
| --- |
| UserAccount |
| - name : String  - age : int  - sex : char  - exerciseList : ArrayList<Exercise> |
| + Account (name : String, age : int, sex : char)  + getName() : String  + getAge() : int  + getSex() : char  + getSelectedExercise() : String[]  + getExerciseList : ArrayList  + setExerciseList (ArrayList<Field>) : void  + toString () : String |

|  |
| --- |
| Exercise \*extends ArrayList<Log> |
| - type : String |
| + Exercise (type : String)  + toString () : String |

|  |
| --- |
| Log |
| - date : String  - weight : int  - reps : int  - calories : int |
| + Log (date : String, weight : int, reps : int,  calories : int)  + getDate() : String  + getWeight() : int  + getReps() : int  + getRM() : RM  + getCalories() : int  + compareTo(log Log) : int  + toString() : String |

|  |
| --- |
| RM |
| - FOMULA : double  - oneRepMax : int |
| + RM (weight : int, reps : int)  + calculateRM(weight : int, reps : int) : int  + getRMatPercentages(percentage: int) : int  + toString() : String |

**File Structure Example - JSON**

{

"name": "Alex Park",

"age": "25",

"sex": "F",

"exercise": {

"bicepLift": [

{

"date": "Jan-11-2015",

"weight": 10,

"reps": 8,

“rm”: 13,

“calories”: 300

},

{

"date": "Jan-12-2015",

"weight": 15,

"reps": 9,

“rm”: 19,

“calories”: 400

},

{

"date": "Jan-13-2015",

"weight": 18,

"reps": 9,

“rm”: 24,

“calories”: 500

}

],

"tricepLift": [

{

"date": "Jan-11-2015",

"weight": 30,

"reps": 30,

“rm”: 49,

“calories”: 200

},

{

"date": "Jan-13-2015",

"weight": 35,

"reps": 30,

“rm”: 50,

“calories”: 300

},

{

"date": "Jan-16-2015",

"weight": "40",

"reps": "30",

“rm”: 53,

“calories”: 400

}

],

"squat": [

"null"

],

"deadLift": [

{

"date": "Jan-11-2015",

"weight": 100,

"reps": 8,

“rm”: 110,

“calories”: 300

},

{

"date": "Jan-14-2015",

"weight": 110,

"reps": 9,

“rm”: 123,

“calories”: 300

},

{

"date": "Jan-18-2015",

"weight": 110,

"reps": 10,

“rm”: 125,

“calories”: 300

}

],

“benchPress” [

“null”

]

}

}

**UML Discussion**

The instance of UserAccount class will be a root JSONObject.

This Object has four name-value pairs, which are “name”, “age”, “sex”, and “exercise”.

The “exercise” value represents another JSONObject. This “exercise” JSONObject has 5 JSONArrays(“ bicepsLift” ,”tricepsLift”, “squat”, “deadlift”, benchPress”) in it, which are instances of the Exercise class. There might be a change in the number of exercises. But, for now, there are 5 of them. These JSONArrays contain JSONObject Logs for each user entry. The log JSONObject has five name-value pairs, which are “date”, “weight”, “reps”, “rm”, and “calories”.