

Gym Helper

Table of Contents

Page 3: Introduction

Page 4: Interface Information

Page 4: StrengthExercises Variables

Page 5: StrengthExercises Methods

Page 7: StoreExercises Variables

Page 8: StoreExercises Methods

Page 9: OneRep Variables

Page 10: OneRep Methods

Page 13: UML

Page 15: Text File Information

Page 15: OneRep File

Introduction

This project is meant to help people kickstart their gym journey with simple workout routines. It's also meant to help keep track of people's gym maxes for what many would consider the three major lifts: squat, deadlift, and bench press.

The project starts out by asking the user what they want to do, and they have the option of creating a workout routine, calculating their one-rep max, or seeing their previous maxes and workouts.

If the user chooses to create a workout routine, they get to choose the amount of time they plan to workout (30-90 minutes), and the number of exercises that they want to do (3 - 7 exercises). After that, the user has the opportunity to decide whether they want to do a push, pull, or legs workout. Finally, the program will output a workout plan that hits as many of the muscle groups associated with that specific workout possible. For example, if a user picks a push workout, but they only have 30 minutes to do 5 exercises, the program will generate 2 chest exercises, 2 shoulder exercises, and a tricep exercise. So, once this is completed, the user will have the option to either create another workout or exit strength exercises. If they exit, they have the option to save their workout to a file.

If the user chooses to calculate a one rep max, they will be prompted to choose an exercise, the amount of weight they did, and the amount of reps they did. For instance, as of April 6th 2023, I can deadlift 225 pounds for 10 reps. If I entered that into the program, the output would be "Deadlift max: 281 lbs". This is because, according to MasterClass's article titled *One-Rep Max Calculator: How to Calculate Your One-Rep Max*, a 225 pound deadlift for 10 reps would yield a 281 pound max if the O'Conner Formula is used ([MasterClass, 2022](#)). If

the user requests for the maxes to be saved to a file, they'll be organized within the three separate exercises and added to the file.

If the user wants to access all of their previous workouts or one rep maxes, the program will access the files to print those out.

Overall, the goal of this project is to give people a place learn about workouts they could do based on the amount of time they have, and a place to help track some of their progress in terms of one rep maxes.

Interface Information

The interface for this project, called UserExp, is made by me and is used to make the OneRep class implement methods that will help with the experience of the user.

The first method, called makeMenu(), is a public abstract non-static method. Its purpose is to provide the user with a menu based on the previous selections that they made. This method will return an integer because the user will enter an integer that goes along with the option they want.

The second method, called display(), is a public abstract non-static method that returns a String. Its purpose is to make the classes display either some workouts or a one-rep max.

The last method, called save(), is also a public abstract non-static method the returns a boolean. Its purpose is to help get all the workouts and maxes the user inputs into saved files. The reason that a boolean is returned is in case there isn't any values to store. In this case, false is returned.

StrengthExercises Variables

The StrengthExercises class will be a public abstract class whose purpose is to help users pick out a workout routine that fits their needs for the given day. The class's primitive and String

instance variables, *time*, *exercises*, and *type*, will all help generate a workout routine. Time is a protected integer variable that will keep track of the time that the workout will take. Exercises is also a protected integer variable that will store the number of exercises that the user will have in the routine, and type is a protected string variable that keeps track of the type of workout that will be performed. There is also a private final String variable, called *reps*, that is responsible for holding the rep ranges of the specific exercises. Since the amount of reps per exercise will be the same no matter what, this is kept at final.

In order to store the exercises, this class also has private final Arrays of Strings to store them all. These range from chestExercises to legExercises, and are kept final since the exercises provided will never change.

There is also a private final String called *workoutQuest*. The purpose of this variable is to store one of the workout questions that the user is asked.

The last variable within this class is called *workout*. It's a protected ArrayList of Strings that will hold all of the exercises for a given workout. The reason that it's protected is because it will be used by the child class, storeExercises, for saving in a file.

StrengthExercises Methods

The first method within StrengthExercises is the default constructor. This constructor is used strictly to give access to the class at the beginning of the program. The second constructor has a StrengthExercises object as a parameter because it will be used for the clone method. The clone method will be a public method that returns a StrengthExercises object. This method will be called if the user wants to save a second copy of the workout routine to a file.

The next method, called *makeMenu*, is a public non-static method that will prompt the user to enter a number that matches the option of what they want to do. Whether that's coming

up with a routine, accessing their previous saved routine, or simply exiting this menu, there's an option for it. The user's choice will be returned as an integer.

Another method in this class is the *workoutQuestions* method. It is a public non-static method that will not return anything. The purpose of this method is to gather information about what type of workout the user wants to have. There are two helper methods, *workoutHelper*, and *workoutHelper2*, that ask their own question about the workout. They are both private void methods. All these methods combined will set the time, exercises, and type variables based on the user's inputs.

The next two methods, *calculateMinCount*, and *calculateMaxCount*, are private methods that return integers. The purpose of these methods is to return the maximum and minimum number of exercises that the user can do.

The next method is the *setWorkout* method. This method is a private non-static void method that sets the workout ArrayList variable. This method will either call the *pushWorkout*, *pullWorkout*, or *legWorkout* method based on the type of workout the user wants.

For the *pushWorkout*, *pullWorkout*, and *legWorkout* methods, these are private non-static methods that return an ArrayList of Strings. The purpose of these methods is to use the given information about the *time*, *exercises*, and *type* variables to generate a workout routine. These workout routines are returned back to *setWorkout* for the user to see. These methods align with the *threeExs*, *fourExs*, *fiveExs*, *sixExs*, and *sevExs* methods. These methods are private void methods that will add various workouts to the *workout* ArrayList. These workouts are added based on whether or not the user wants a push, pull, or legs routine.

The next method is the display method, which is a protected method with an ArrayList of Strings parameter. This method will return the workout routine in a formatted String that allows the user to see what exactly they should do for the workout.

The toString method is a public non-static method that returns a String. It will make the StrengthExercises into a String type so that it can be printed out consistently. This method will use the display method to help with this process.

The save method is also in this class. However, it doesn't implement the UserExp interface. It's a public non-static method that returns a boolean value, and it will ask the user if they want to save this workout for later. If they do, the method will return true, and false otherwise.

The duplicate method is a public non-static method that returns a boolean. It will ask the user if they want to save this workout to their own file. If so, true is returned. Otherwise, false is returned.

StoreExercises Variables

The StoreExercises class is a child of the StrengthExercises class. The purpose of this class is to store workouts from the StrengthExercises objects into files. The reason that it's in a separate class is that I believed that it would be too many methods within a single class. Thus, making it harder to read.

There are only two instance variables for this class. The first one, called *workoutPlan*, is a private ArrayList of an ArrayList of StrengthExercise objects. The purpose of this variable is to store all of the user's previous workouts for later file printing. The second variable is called *file*, and it's a private File variable. This file will contain all of the workouts that the user has saved.

StoreExercises Methods

The two constructors for this class are the default constructor and the constructor with a `StoreExercises` parameter within it. The default constructor, like the `StrenthExercises`'s default constructor, is simply used to begin the program. Since the user doesn't have any workouts that can be saved, nothing will be in the `ArrayList`. As for the other constructor, it is used for the `clone` method to make a deep copy of the current *workoutPlan* `ArrayList` and store it in this new object. This will happen by making clones of each object in *workoutPlan*.

The first distinct method within this class is the *display* method, which is private and returns a `String`. This method will help format the workouts so that they can be displayed for the user as well as in the file. It will be called by this class's `toString` method.

The second distinct method is the *displayLast* method. This method is a public non-static void method that will display the user's last saved workout routine. This method will only return one set of exercises, and it must be saved by the user in order for it to be printed. If no files are saved, a *no saved workouts* message will be displayed.

The third method is the `toString` method. This method is a public non-static one that returns a `String` value. Its purpose is to show the user elements within the `StoreExercises` file. It will be used when the user asks to see their workouts from their previous use of this program. If nothing is stored, a *no saved workouts* message will be displayed.

The fourth method is the `setFile` method. This method is a public non-static void method that will be called when the user wants to save their workout to a separate file. It will set the `StoreExercise` file to whatever the user inputs as long as it's a valid text file.

Another method is the clone method. This method is a public non-static void method that will be used to create another StoreExercises object. The purpose of the method is to help the user create a duplicate object to save the workouts in a different file.

The *saveWorkout* method is a public non-static void method that will add a new workout to the *workoutPlan* ArrayList. The purpose of this method is to help the user save their workout so that it can be printed into a file.

The last method in this class is the *printFile* method. The purpose of this method is to print all of the workouts into the file. This method will only be called at the end of the program, but it could be also called when the user wants to save a routine into a separate file. This is a public non-static boolean method that will print any of the saved workout routines into a file. If there aren't any workouts yet, nothing will be printed and false is returned.

OneRep Variables

The OneRep class is a public class that will implement the UserExp interface. The objective of this class is to provide methods that will help the user calculate their one-rep max for three different exercises (Deadlift, squat, and bench), and save these to a file. These exercises each have their own set of private ArrayLists of Integers. The *allDeadLifts*, *allBenchBress*, and *allSquats* are the ArrayList variables that will store the values of the inputted one rep maxes.

For the primitive and String types, there are three private variables: *weight*, *reps*, and *type*. *Type* is a String that's responsible for keeping track of the exercise that the user wants to calculate. *Weight* and *reps* are integers that represent how much (in lbs) the lift is and how many times the user lifted the weight.

The last variable is called *file*. This private File instance variable will be used to store the one rep maxes within a file.

OneRep Methods

This class has two constructors: a default constructor and a constructor with a oneRep object as the parameter. The default constructor's intended purpose is to be used to create a base object for the user to be asked about their maxes. On the other hand, the constructor with the oneRep parameter is meant to be used for a clone method. This constructor would be used if the user wants to save everything to a separate file. The three ArrayLists for the separate exercises are deep copied in this constructor.

The first distinct method within this class is the makeMenu method. It is a public non-static method that returns an Integer. It is also from the UserExp interface. The goal of this method is to prompt the user to decide whether or not they want to calculate their one rep max, display previous maxes, or exit the menu. The user can select a number for their answer, and this would be returned to a front-end class.

The next method is called *expAsker*. This method is a public non-static void method that will prompt the user to enter in their desired exercise. This is saved as a variable, and then this method will call the *weightAsker* method, which is a private void method. This will ask the user to enter the weight performed for the given exercise. Finally, this will call the *oneRepAsker*, a private void method that will ask for the reps performed for the given exercise. These will also be saved as variables. This method also calls the class's *calculate* method. **Disclaimer:** The maximum weight that can be entered is 1000 lbs, and the maximum reps that can be entered is 10. This is to maintain clarity within the OneRep file.

The calculate method is a private void method that will return an integer. Its purpose is to calculate the one rep max of a given exercise using the [O'Conner](#) Formula. The method will also save this value into one of the ArrayLists that matches the exercise being calculated. Finally, the method displays the user's max for them to see.

The display method is a public non-static method that is an implementation of the UserExp's display method. The method first calls the *calculate* method, which will return the one rep max based on user input. Then, the method will print a formatted version of the max for the user to see.

The next method is the save method. This method is a public non-static method that implements UserExp's save method. This method will print all of the maxes into the OneRep file. To print the file elements, the *printElem* method is called. This is a public void method that will format a file to print the maxes inside it. There are also two helper methods, *printElem2* and *printElem3*, which are private non-static methods that return strings. They will display the actual maxes within the file. If there are no maxes for a specific exercise, a message will appear. This happens at the end of the program.

The next method is the *separate* method. It is a public non-static method that returns a boolean. This method will ask the user to save their current data into another file. If so, true is returned. Otherwise, the method will return false. The purpose of this method is to facilitate the opportunity for the user to create a backup file for storing their information.

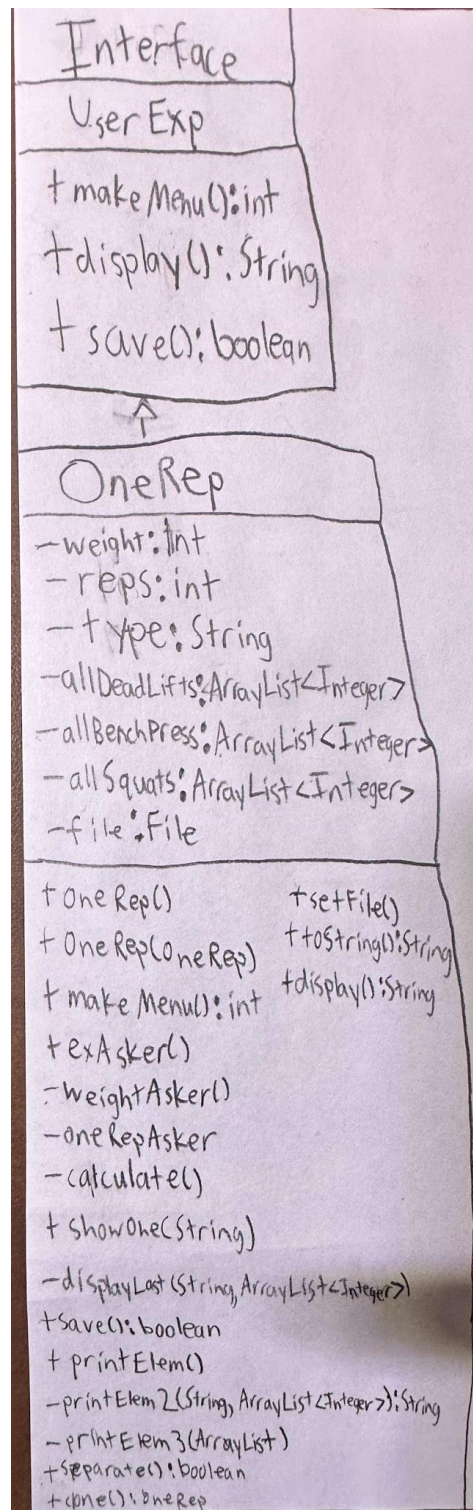
The next method is the setFile method. This public non-static void method will ask the user for a file name, which should be a text file. The file variable will then initialize a new text file with this file name.

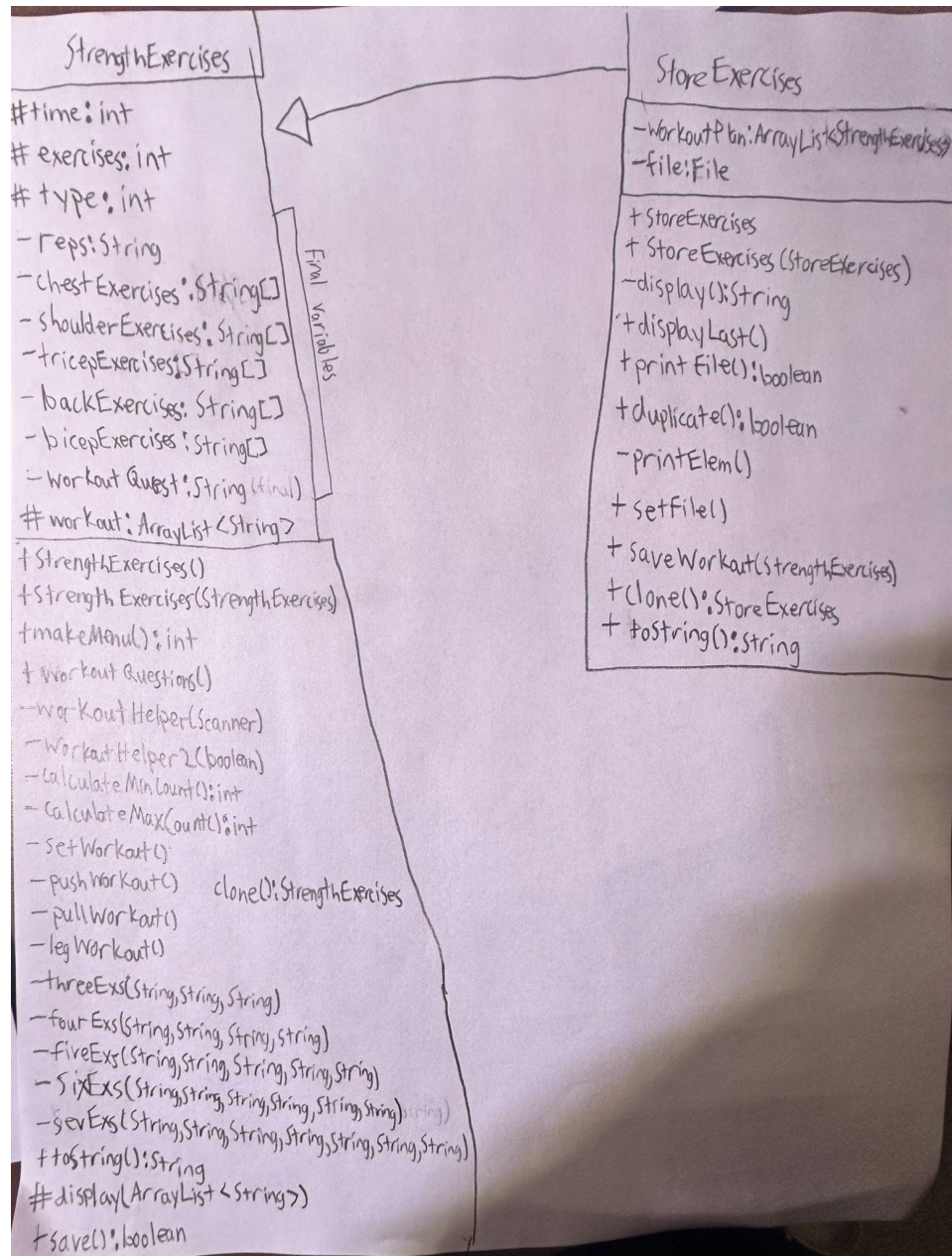
Another method in this class is the clone method. This is a public, non-static method that returns a OneRep object. This method's purpose is to help create a new object for the user. This object will be used for storing the maxes in a different file.

The toString method is a public, non-static method that returns a String. The purpose of this method is to formally display the maxes that are stored within the OneRep file. If there's nothing in the file, a "Nothing stored within the file yet!" message will be displayed.

The last method within this class is the showOne method. This public non-static void method is responsible for showing the user a set of the one rep maxes that they choose. If no one rep maxes are saved, an empty message will be displayed. Otherwise, the displayLast method is called, which is a private void method that displays a formatted version of the user's one rep maxes for a given exercise.

UML





File Information

StoreExercises File

The file for storing the workouts will be a text file titled *workouts.txt*. The purpose of this file is to save and display the workouts that the user created. The values stored will be the ArrayList elements from the *workoutPlan* ArrayList of objects. The photo below shows an example of what the inside of the file will look like. At the top, there's a workout saved message followed by the number of workouts within the file. This number comes from the length of the *workoutPlan* ArrayList. The file then skips a line to show the first workout. The “workout *num*: *type*” header is formatted to show the amount of exercises for this workout for the *num*, and the type of workout for the *type*. Then, the actual workout will be on the next line. These are just the elements of the *StrengthExercises* objects in the ArrayList. The user's selections for their workout will determine these elements. Then, the file will skip a couple of lines to display any of the following workouts. This is for clarity purposes.

```

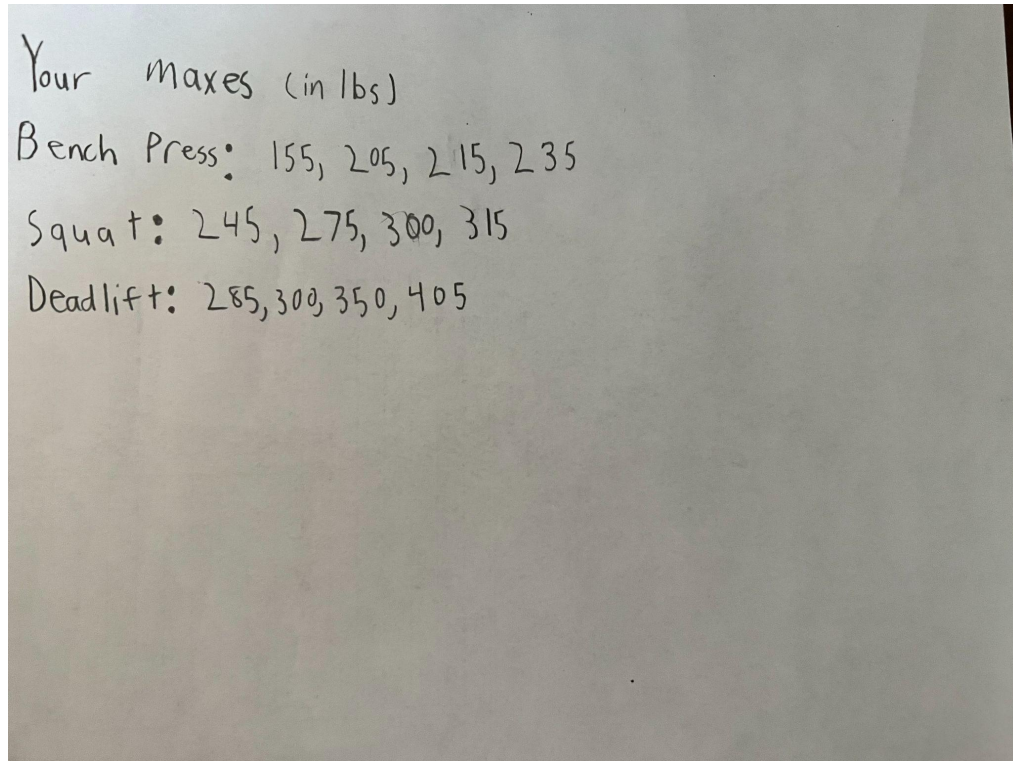
1 Your workouts saved: 3
2
3 Workout 1 (Legs): This workout has 5 exercises and should take 60 minutes
4 Squats: 4 x 8-12, RDL: 4 x 8-12, Leg Extensions: 3 x 8-12, Hamstring Curls: 3 x 8-12, Standing Calf Raise: 3 x 8-12
5
6 Workout 2 (Push): This workout has 5 exercises and should take 90 minutes
7 Incline Bench Press: 4 x 8-12, DB Bench Press: 4 x 8-12, DB Side Raise: 4 x 8-12, DB Shoulder Press: 4 x 8-12, Dips: 4 x 8-12
8
9 Workout 3 (Push): This workout has 3 exercises and should take 30 minutes
10 Incline Bench Press: 4 x 8-12, DB Bench Press: 4 x 8-12, DB Side Raise: 4 x 8-12

```

OneRep File

The file for storing the one-rep maxes will be a text file titled *oneMax.txt*. The purpose of this file is to save and display the one rep maxes that the user wanted to calculate. The values

stored are from either the *allDeadlifts*, *allBenchPress*, or *allSquats* ArrayList in a formatted manner. All of the values for an exercise will be displayed on a single line. The picture below is a sample output. Although the numbers are ascending, the values will be in the order of when they're calculated. For example, if a calculated squat max of 340 was made before a calculated squat max of 250, the 340lb max squat will appear first in the output.



Handwritten sample output showing maxes for Bench Press, Squat, and Deadlift. The text is written on a piece of paper and lists the maximum values for each exercise in a single line, separated by commas. The values are ascending, but the order of calculation is preserved.

Your maxes (in lbs)

Bench Press: 155, 205, 215, 235

Squat: 245, 275, 300, 315

Deadlift: 285, 300, 350, 405

References

“One-Rep Max Calculator: How to Calculate Your One-Rep Max - 2023.”

MasterClass, 19 Jan. 2022,

<https://www.masterclass.com/articles/one-rep-max-calculator>.

[https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#copy\(java.util.List,%20java.util.List\)](https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#copy(java.util.List,%20java.util.List)) (ArrayList Copy method)