WORKOUT PLANNER

1-Analysis

Background

Many students across multiple sixth forms in London make use of commercial gyms. Mostly these gyms are the ones that are situated the closest to school and students will go in large groups(friend groups). The result of this is extremely crowded gyms between 3:45-5:30 pm.

Gym-goers have the main goal of completing all the exercises in their workout in a suitable amount of time. Ideally, with a minimal amount of time wasted looking for an available machine to complete their exercise.

Identification of Problem

When there are large groups of students crowding the gym they tend to take a specific machine and stay at this machine for around 20-25 minutes which is an overinflated amount of time which is both inefficient for said group but also this method disturbs other gym-goers as they are unable to use a certain machine and this results in many people leaving workouts uncompleted as they cant be bothered to wait so long for one machine. Furthermore the optimal rest time between sets for an individual who is working out is around 3 minutes but when working out in a large group they must wait for other members of the group to have their go which can take up to 10 minutes. This makes the exercise that is done redundant.

Description of Current System

Nearly all students enter the gym and simply look for an available machine to complete their exercise and keep following this cycle of looking for an available machine until their workout is complete. This results in incomplete workouts which are not optimal for progression in the gym.

Identification of potential users

My target user group consists of year 12 and year 13 students across all secondary schools in London. However it can also be used by any group of individuals who plan on following the same list of exercises.

User Needs

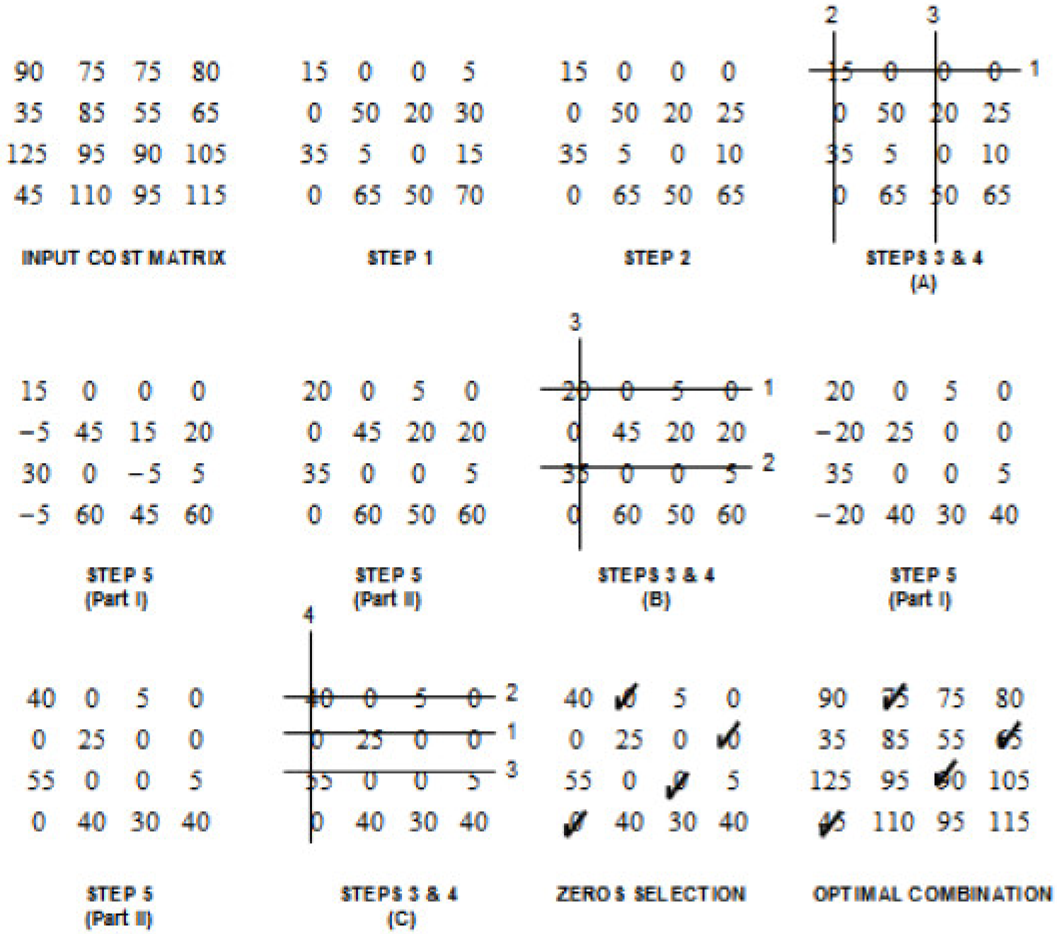
1. The program should be able to create individual plans for each member of the group
2. The users should be able to choose whether they want to follow an existing workout plan or create their own set of exercises
3. Users should be able to create multiple workout plans where they choose different exercises. Then they should be able to select which plan they want to follow on this occasion.

Research into potential solutions

With respect to what data should be taken into account, there are two key pieces of data which truly matter. The time it takes for each person to finish an exercise and the person's preferred order. There are two main potential solutions to this problem. One makes use of the Hungarian algorithm while the other uses the Gale-Shapley (or stable marriage) algorithm. The difference is that the Hungarian algorithm only takes in one set of inputs while the other uses two. Usage of the Hungarian algorithm would mean that the only data taken into consideration would be either how long each person takes to do each exercise OR each user's preferred order. However, using the Gale-Shapley algorithm would compromise the most optimal solution in order to maximise user-satisfaction.

Hungarian Algorithm

The Hungarian algorithm is a matching algorithm developed by Harold Kuhn in 1955. The algorithm relies on matrix manipulation to find optimal matches between two sets of objects based on a cost factor.



$$\begin{matrix} 90 & 75 & 75 & 80 \\ 35 & 85 & 55 & 65 \\ 125 & 95 & 90 & 105 \\ 45 & 110 & 95 & 115 \end{matrix}$$

INPUT COST MATRIX

$$\begin{matrix} 15 & 0 & 0 & 5 \\ 0 & 50 & 20 & 30 \\ 35 & 5 & 0 & 15 \\ 0 & 65 & 50 & 70 \end{matrix}$$

STEP 1

$$\begin{matrix} 15 & 0 & 0 & 0 \\ 0 & 50 & 20 & 25 \\ 35 & 5 & 0 & 10 \\ 0 & 65 & 50 & 65 \end{matrix}$$

STEP 2

$$\begin{array}{c|cccc|c} & 2 & & 3 & & \\ \hline 15 & 0 & 0 & 0 & & 1 \\ 0 & 50 & 20 & 25 & & \\ 35 & 5 & 0 & 10 & & \\ 0 & 65 & 50 & 65 & & \end{array}$$

STEPS 3 & 4
(A)

$$\begin{matrix} 15 & 0 & 0 & 0 \\ -5 & 45 & 15 & 20 \\ 30 & 0 & -5 & 5 \\ -5 & 60 & 45 & 60 \end{matrix}$$

STEP 5
(Part I)

$$\begin{matrix} 20 & 0 & 5 & 0 \\ 0 & 45 & 20 & 20 \\ 35 & 0 & 0 & 5 \\ 0 & 60 & 50 & 60 \end{matrix}$$

STEP 5
(Part II)

$$\begin{array}{ccccc} 3 & & & & \\ 20 & 0 & 5 & 0 & 1 \\ 0 & 45 & 20 & 20 & \\ 35 & 0 & 0 & 5 & 2 \\ 0 & 60 & 50 & 60 & \end{array}$$

STEPS 3 & 4
(B)

$$\begin{matrix} 20 & 0 & 5 & 0 \\ -20 & 25 & 0 & 0 \\ 35 & 0 & 0 & 5 \\ -20 & 40 & 30 & 40 \end{matrix}$$

STEP 5
(Part I)

$$\begin{matrix} 40 & 0 & 5 & 0 \\ 0 & 25 & 0 & 0 \\ 55 & 0 & 0 & 5 \\ 0 & 40 & 30 & 40 \end{matrix}$$

STEP 5
(Part II)

$$\begin{array}{ccccc} 4 & & & & \\ 40 & 0 & 5 & 0 & 2 \\ 0 & 25 & 0 & 0 & 1 \\ 55 & 0 & 0 & 5 & 3 \\ 0 & 40 & 30 & 40 & \end{array}$$

STEPS 3 & 4
(C)

$$\begin{matrix} 40 & \checkmark 0 & 5 & 0 \\ 0 & 25 & 0 & \checkmark 0 \\ 55 & 0 & \checkmark 0 & 5 \\ \checkmark 0 & 40 & 30 & 40 \end{matrix}$$

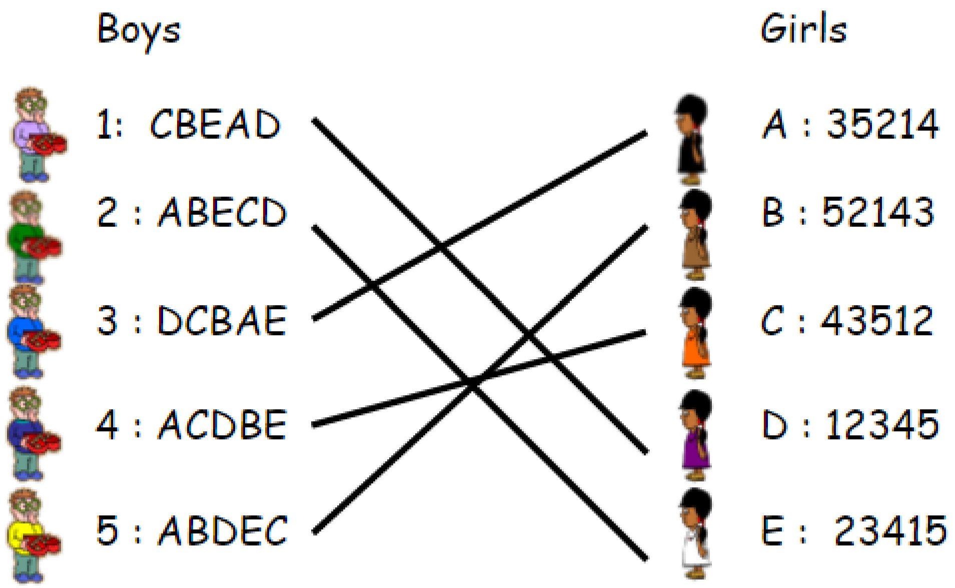
ZEROS SELECTION

$$\begin{matrix} 90 & \checkmark 75 & 75 & 80 \\ 35 & 85 & 55 & \checkmark 65 \\ 125 & 95 & \checkmark 90 & 105 \\ \checkmark 45 & 110 & 95 & 115 \end{matrix}$$

OPTIMAL COMBINATION

Gale-Shapley Algorithm

The Gale-Shapley algorithm was created as a solution to the stable marriage problem. A simple way of modelling this algorithm is by visualising it as having a group of n bachelors and a group of n bachelorettes. Each person in these two groups will have a list of preferences ranking who they would like to marry from most to least. A stable pairing always exists and the point of the algorithm is to find it. A pairing is defined as unstable if: Person A from set 1 prefers Person B from set 2 over Person A's current pairing AND Person B prefers Person A over their current pairing. Put alternatively, there is no pair (A, B) where these people prefer each other over their current pairings.



User Interview

I will be interviewing Aden who is a year 13 student who uses the gym every day after school. Key: Roshan G. Ganithi: **Q**, Aden Grandcourt: **A**

Q: Good morning Aden, today I just want to ask you some questions about your gym experience

A: Alright then. Let us begin.

Q: Firstly, I just want to make sure that you go to the gym with a group of friends.

A: Yes that is correct. Whenever I go to the gym I go with 4 of my friends. We all decided to start going to the gym at the beginning of year 12. It does seem that many other friend groups had the same idea. We always see many groups of people of a similar age to us.

Q: So there are many people who are similar to you at the gym. What do you think is the biggest hindering factor when you go to the gym? What I mean is that I want to know what you think could change to make your workouts more efficient.

A: If I'm being honest I feel like we each get too much rest time between sets because I read a research paper online that stated that we should be getting a maximum of around 2 and a half minutes. Basically, what happens is we each take turns on the machine until we have each finished our 3-4 sets. By the time my 4 other friends have finished their set around 5 minutes have passed. What this causes is my workouts are not optimal and my gym progress is facing some drawbacks. Furthermore we normally spend a lot of time waiting around to find an available machine. Sometimes we are forced to skip a certain exercise altogether because we need to get back home quickly

Q: Do you believe that having each person at a different machine and swapping machines would be better as it would allow you to have full control over your rest time?
A: Yes. That would be much better as it would make my workouts more optimal. Besides not being able to talk to my friends between sets is a very minor drawback as I talk to them before and after the workouts as well as at school.

Q: Would there be any other benefits for you if there was code that created individual workouts for each of you?

A: Of course. It would make planning my workouts much easier if I could click a few buttons and I would be able to see my schedule for the day.

Q: Is there anything you think would be very important to have in this program?

A: Personally, I think there should be a main menu where I can select what i want to do such as see everyone's schedule for the workout or change the workout we are doing on the day

Q: And finally would there be anything else you would like to see in a potential code?

A: It would be pretty cool if I could see maybe a table that stored my previous exercises along with the weight/resistance I used when doing each set.

Q: Thank you very much for your input. I hope to please you with my project.

A: Goodbye.

Chosen Solution

Given the users' needs and taking the user interview into account I have decided that the most apt solution would be the use of the Gale-Shapley algorithm as it can take into consideration two sets of preferences which I feel would make pairings that will make as many group members as happy as possible. However I need to account for the fact that, naturally, it is very unlikely that I have the same number of people and exercises in a workout. To generate the order of exercises for each user I will modify their 'preference' array and the exercises preference array to ensure that the same pairing does not occur twice.

Objectives of this Project

1 - User Need: The user should be able to manage all users of the system

1.1 - The User should be able to add users to the database
1.2 - The User should be able to delete users from the database

1.3 - The User should be able to modify users in the database
1.4 - The User should be able to view all users in the database
1.5 - Whenever a User is created they should be made to enter their preference and time taken for each exercise that is currently in a workout

2 - User Need: The user should be able to manage the workouts in the system
2.1 - The User should be able to add workouts to the database
2.2 - The User should be able to delete workouts from the database
2.3 - The user should be able modify the exercises in the workout in the database
2.3.1 - The user should be able to add an exercise to an existing workout
2.3.2 - The user should be able to delete an exercise from an existing workout
2.3.3 - The user should be able to modify the number of sets of an exercise in an existing workout
2.4 - The User should be able to view all workouts in the database
2.5 - The user should be able to view all exercises and their respective number of sets in an existing workout
3 - User Need: The user should be able to get timetables for their workouts
3.1 - The Gale-Shapley algorithm should be used to create the routines.
3.2 - The User should be able to get routines for every person in the group
3.3 - The User should be able to enter a specific user's name and get their routines for a workout

2 - Design

General Explanation of Solution

I will be programming the system in Java and the interface being used will be the command line. I have decided to use Java due to its object-oriented nature which will help me structure my code better. I will use an SQLite3 database to store all data that will be used by the system.

Basic Idea of the Gale-Shapley Algorithm

Let us assume we have the arrays for the gym-goers and machines along with their preferences here. Preferences are placed in two two-dimensional arrays where we create parallel arrays where the index of a gym-goer in the 'gym-goers' array will be the same as the index of their preference in the gymGoerPreferences array. We will also assume that the gym-goers will 'propose'. Stable pairings would still be made if the machine 'proposed'.

```
WHILE(There exists a person or machine who isn't paired)
    FOR(Each person in the gym-goer array)
        IF(The person doesn't have a pairing)
            The person will propose to the next machine in its
            preference array which hasn't rejected them yet.
            IF(The machine isn't paired)
                Create pairing between the person and machine
            ENDIF
            ELSE
                IF(Machine prefers proposer over person in
                its current pairing)
                    Create pairing between proposer and
                    machine while the machine ‘rejects’ its
                    previous partner
                ENDIF
                ELSE
                    Machine rejects proposer
                ENDELSE
            ENDELSE
        ENDIF
    ENDFOR
END WHILE
```

PseudoCode of the Gale Shapley Algorithm

```
people = [*names go here*]
exercises = [*exercises go here*]
userPreferences = *2 dimensional array with each user’s
preferences for each exercise*
exercisePreferences = *2 dimensional array with each exercise’s
preferences for each user. These are based on the time taken for
the user to complete each exercise*
userMatchIndexes = [-1, -1, -1, . . . . ]
excerciseMatchIndexes = [-1, -1, -1, . . . . ]
userNextProposal = [0, 0, 0, . . . . ]
matchedUsers = [false, false, false, . . . .]
matchedExcercises = [false, false, false, . . . . ]

WHILE(containsFalse(matchedUsers)):
    FOR x in range (len(people)):
        IF(!matchedUsers[x]):
            exerciseIndex = indexFromID(userPreferences[i]
            [userNextProposal[i]], exerciseIDs)
            //indexFromID is a linear search
            IF(!matchedExercises[exerciseIndex]):
                matchedExercises[exerciseIndex] = true
                matchedUsers[x] = true;
                userMatchIndexes[x] = indexFromID
                (exerciseIDs[exerciseIndex],
                userPreferences[x])
                exerciseMatchIndexes[exerciseIndex] =
                indexFromID(exerciseIDs[userIDs[i],
                exercisePreferences[exerciseIndex])
            ELSE:
                userIndex = indexFromID(userIDs[i],
                exercisePreferences[exerciseIndex])
```

```
                    if(exerciseMatchIndexes[exerciseIndex] >
                    userIndex):
                         int rejectedUserIndex =
                         indexFromID(exercisePreferences
                         [exerciseIndex][exerciseMatchIndexes
                         [exerciseIndex]], userIDs);
                         userMatchIndexes[rejectedUserIndex] = -1
                         matchedUsers[rejectedUserIndex] = false
                         exerciseMatchIndexes[exerciseIndex] =
                         userIndex;
                         userMatchIndexes[x] = indexFromID
                         (exerciseIDs[exerciseIndex],
                         userPreferences[x])
                         exerciseMatchIndexes[exerciseIndex] =
                         indexFromID(exerciseIDs[userIDs[i],
                         exercisePreferences[exerciseIndex])
                    ENDIF
               ENDIF
          ENDFOR
     ENDWHILE
//This is the pseudocode for when there are an equal or greater
number of exercises than people. If there are less exercises then
the exercises will be the 'proposers'
```

Tracing of Gale-Shapley Algorithm

Input Data:
Gym-Goers: [Alhassan, Bogdan, Cai, Daniel, Edgar]
Exercise: [Bench Press, Tricep Curls, Pec fly, Shoulder Press, Lateral Raises]
//Preferences are put in a two-dimensional array where each list of preferences has the same index as the person themselves.
//As mentioned above in analysis we will have each person's preferred order as one set of preferences and use the time they take for each workout to make the machine's preferences for the other set of preferences. Machines will prefer people who take less time on them.

PeoplePreferences:[[Shoulder Press, Tricep Curls, Pec Fly, Bench Press, Lateral Raises], [Lateral Raises, Pec Fly, Tricep Curls, Bench Press, Shoulder Press], [Tricep Curls, Lateral Raises, Bench Press, Shoulder Press, Pec Fly], [Lateral Raises, Tricep Curls, Shoulder Press, Pec Fly, Bench Press], [Shoulder Press, Bench Press, Tricep Curls, Pec Fly, Lateral Raises]]

Exercise Preferences: [[Daniel, Bogdan, Edgar, Cai, Alhassan], [Bogdan, Alhassan, Daniel, Cai, Edgar], [Alhassan, Cai, Edgar, Daniel, Bogdan], [Daniel, Alhassan, Cai, Bogdan, Edgar], [Bogdan, Edgar, Alhassan, Cai, Daniel]]

Since the algorithm requires one side to be the proposers and the other the acceptors/rejectors, we will let the men be proposers.

People Next Proposal					Pairings				
Alhassan	Bogdan	Cai	Daniel	Edgar	Alhassan	Bogdan	Cai	Daniel	Edgar
Shoulder Press	Lateral Raises	Tricep Curls	Lateral Raises	Shoulder Press					
Tricep Curls					Shoulder Press				
	Pec Fly					Lateral Raises			
		Lateral Raises					Tricep Curls		
			Tricep Curls						
				Bench Press					
			Shoulder Press				Null	Tricep Curls	
				Tricep Curls					Bench Press
		Bench Press							
		Shoulder Press							
		Pec Fly							
							Pec fly		

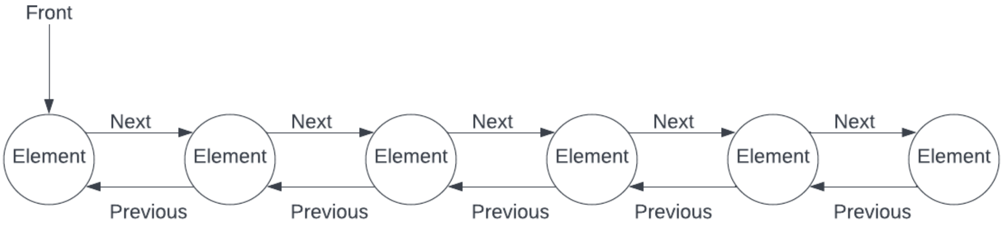
Final Matches: Alhassan: Shoulder Press, Bogdan: Lateral Raises, Cai: Pec Fly, Daniel: Tricep Curls, Edgar, Bench Press.

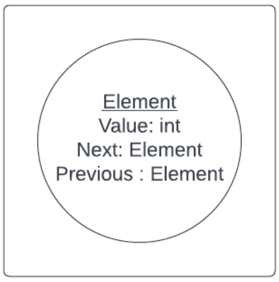
Data Structures

Linked List

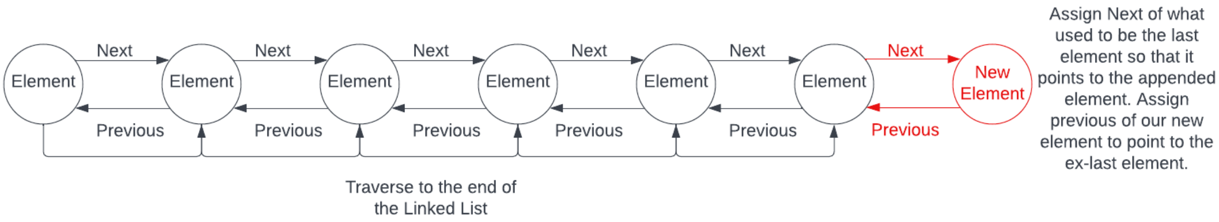
To make my linked list more thorough I am using a doubly linked list. I use two unique constructors when initialising my linked list. One of these constructors doesn't take in any arguments and creates an empty linked list. The other constructor has one argument which is an array of integers. This converts the array into a linked list which later allows me to manipulate the linked list by rearranging the elements. There are three main functions which I will need. A function to add to the linked list, one to delete an element from the linked list, and a function that converts the linked list to an array of integers and returns said array.

Linked List Structure



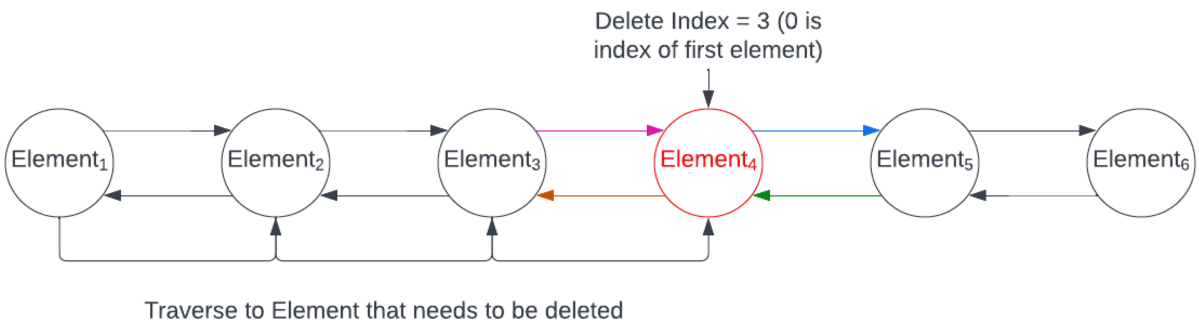


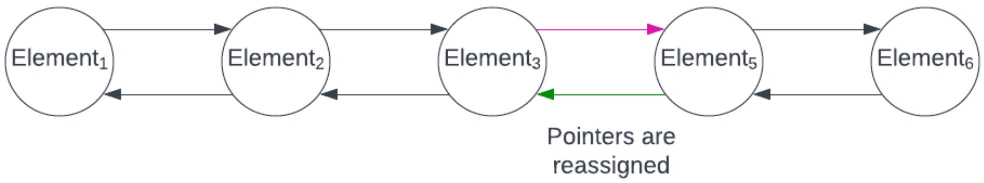
Append Function


Element
Element
Element
Element
Element
Element
New Element
Next
Next
Next
Next
Next
Next
Previous
Previous
Previous
Previous
Previous
Previous
Assign Next of what used to be the last element so that it points to the appended element. Assign previous of our new element to point to the ex-last element.
Traverse to the end of the Linked List

Delete Function

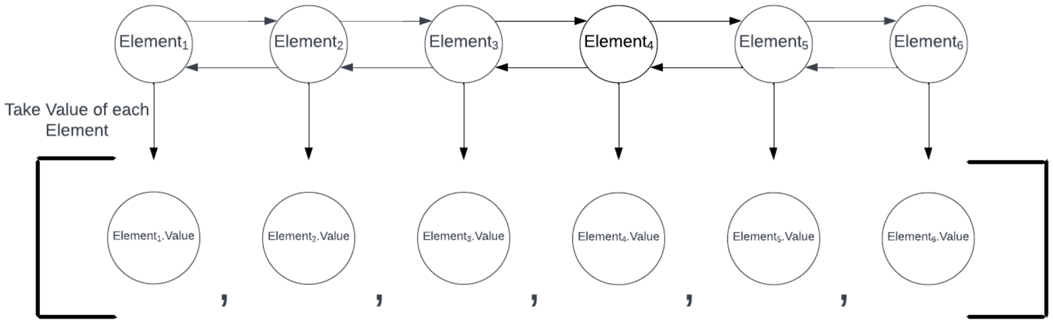
This function takes in one argument which is the index of the element which needs to be deleted. First, we traverse to the element which needs to be deleted. Then we reassign the next pointer of the deleted element's previous element so that it points to the deleted element's next element. We do the same to the previous pointer of the deleted element's next element.


Delete Index = 3 (0 is index of first element)
Element1
Element2
Element3
Element4
Element5
Element6
Traverse to Element that needs to be deleted



As Array Function

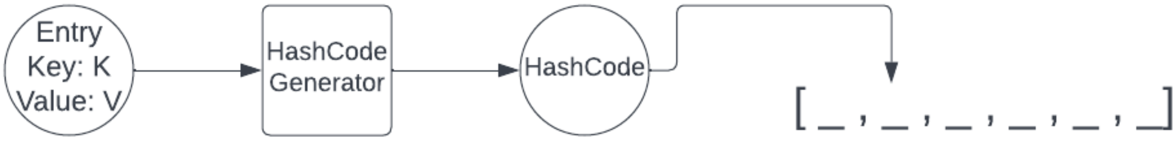
This function takes every element's value in the linked list and adds them to an array of integers in the same order that the elements exist in the linked list.



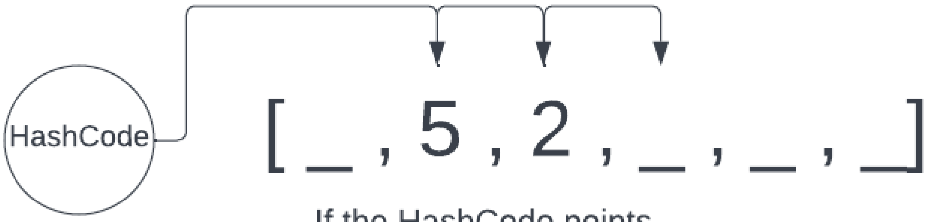
Hash Map

The purpose of my hash map is to improve the look up time when a user wants to fetch a time table for their workout.

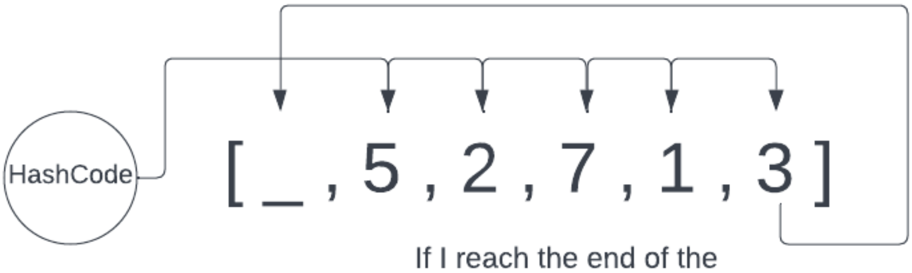
Add Function



The HashCode generator creates an integer based on the key. I then take the remainder of the HashCode when divided by the size of the HashMap. This remainder is the index of the array I will store the value in.



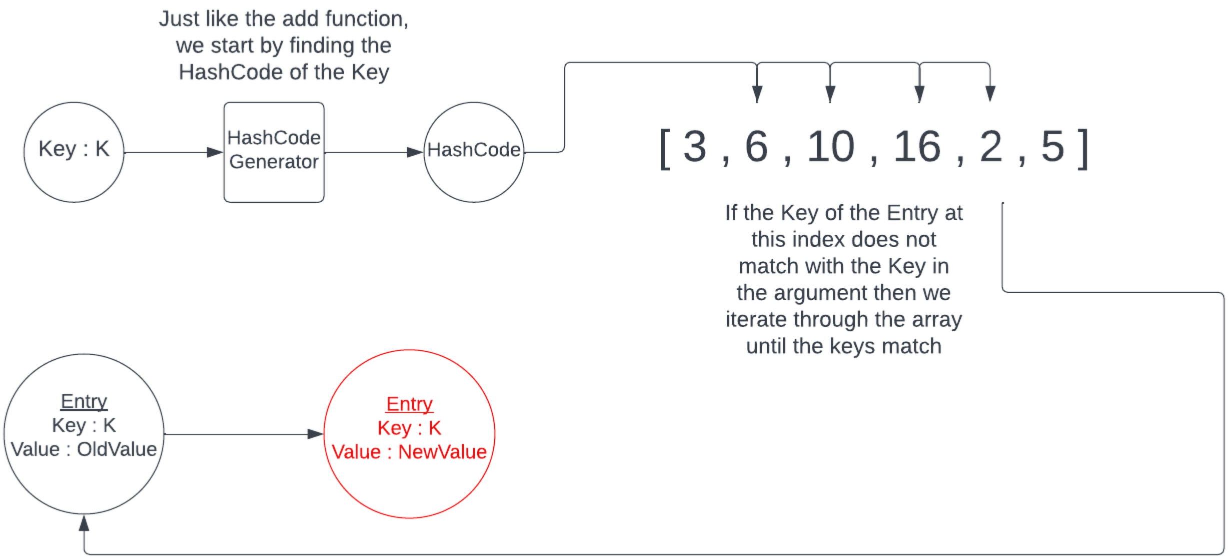
If the HashCode points to an occupied index in the array then I iterate through the array until I find an expty index.



If I reach the end of the array then I cycle back to the beginning of the array

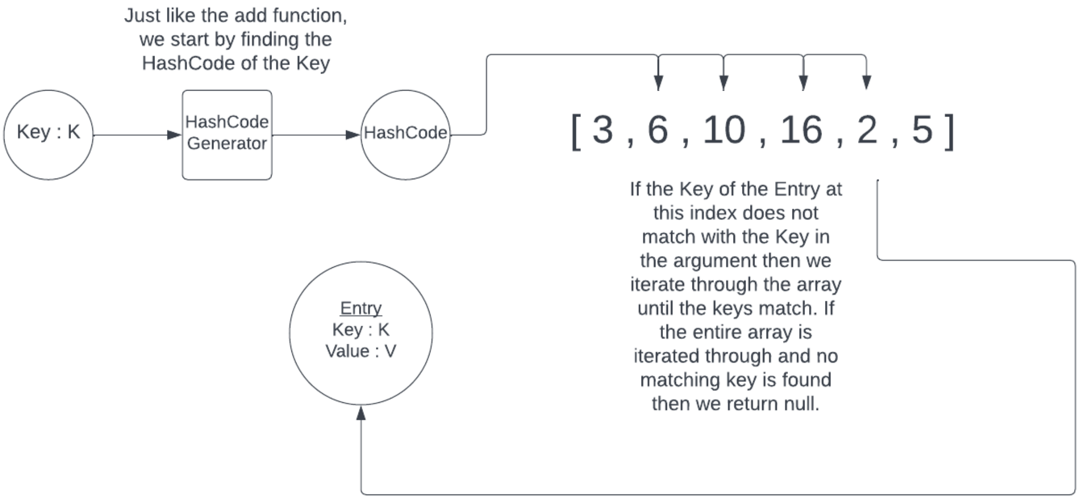
Alter Function

As stated by the name, the purpose of the alter function is to change the value of an entry that is specified by the given key.


Just like the add function, we start by finding the HashCode of the Key
Key : K
HashCode Generator
HashCode
[3 , 6 , 10 , 16 , 2 , 5]
If the Key of the Entry at this index does not match with the Key in the argument then we iterate through the array until the keys match
Entry
Key : K
Value : OldValue
Entry
Key : K
Value : NewValue

Peek Function

The peek function simply returns the value of an entry where the key is the same as the given key


Just like the add function, we start by finding the HashCode of the Key
Key : K
HashCode Generator
HashCode
[3 , 6 , 10 , 16 , 2 , 5]
If the Key of the Entry at this index does not match with the Key in the argument then we iterate through the array until the keys match. If the entire array is iterated through and no matching key is found then we return null.
Entry
Key : K
Value : V

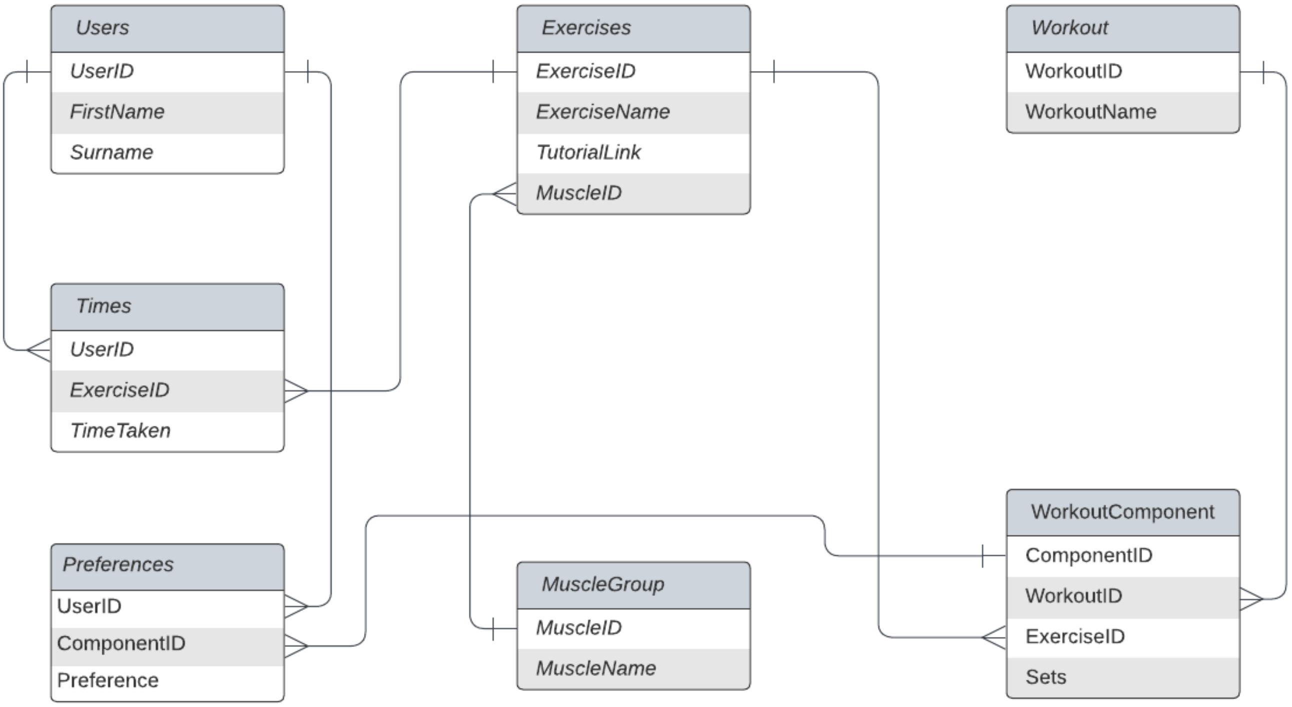
Contains Function

This function relies on the peek function. If the peek function returns a null value then we can say that there is no Entry in our HashMap with the given key. If any non-null value is returned then the function returns true because there is an Entry with the given key.

Index In Table Function

The principle of this function is similar to the peek function in the way we search for the desired Entry. However, instead of returning the value of the Entry we return the Entry's index in the HashMap table

Entity Relationship Diagram



Data Definition Language For Database Tables

User Table

```
CREATE TABLE Users (
      UserID INTEGER PRIMARY KEY AUTOINCREMENT
      NOT NULL,
      FirstName STRING NOT NULL,
      Surname STRING NOT NULL
)
```

MuscleGroup Table

```
CREATE TABLE MuscleGroup (
      MuscleID INTEGER PRIMARY KEY AUTOINCREMENT
      NOT NULL,
      MuscleName STRING
)
```

Exercises Table

```
CREATE TABLE Exercises (
      ExerciseID INTEGER PRIMARY KEY NOT NULL,
      ExerciseName STRING NOT NULL,
      TutorialLink STRING,
      MuscleID STRING REFERENCES MuscleGroup (MuscleID)
)
```

Workout Table

```
CREATE TABLE Workout (
      WorkoutID INTEGER PRIMARY KEY AUTOINCREMENT,
      WorkoutName STRING
)
```

WorkoutComponent Table

```
CREATE TABLE WorkoutComponent (
      ComponentID INTEGER PRIMARY KEY AUTOINCREMENT,
      WorkoutID INTEGER NOT NULL
      REFERENCES Workout (WorkoutID),
      ExerciseID INTEGER NOT NULL
      REFERENCES Exercises (ExerciseID),
      Sets INTEGER NOT NULL
)
```

Preferences Table

```
CREATE TABLE Preferences (
      UserID INTEGER NOT NULL,
      ComponentID INTEGER NOT NULL,
      Preference INTEGER NOT NULL,
      PRIMARY KEY (UserID, ComponentID),
      FOREIGN KEY (UserID)
      REFERENCES Users (UserID),
      FOREIGN KEY (ComponentID)
```

```
        REFERENCES WorkoutComponent (ComponentID)
);
```

Times Table

```
CREATE TABLE Times (
        UserID INTEGER NOT NULL,
        ExerciseID INTEGER NOT NULL,
        TimeTaken INTEGER NOT NULL,
        PRIMARY KEY (UserID,ExerciseID),
        FOREIGN KEY (UserID) REFERENCES Users (UserID),
        FOREIGN KEY (ExerciseID)
        REFERENCES Exercises (ExerciseID)
)
```

Final Database Structures

User Table

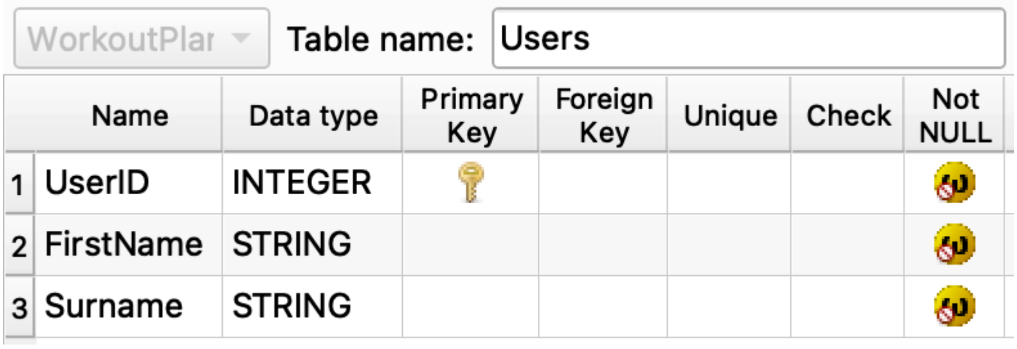


Table name: Users

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL
1	UserID	INTEGER	🔑				✓
2	FirstName	STRING					✓
3	Surname	STRING					✓

MuscleGroup Table

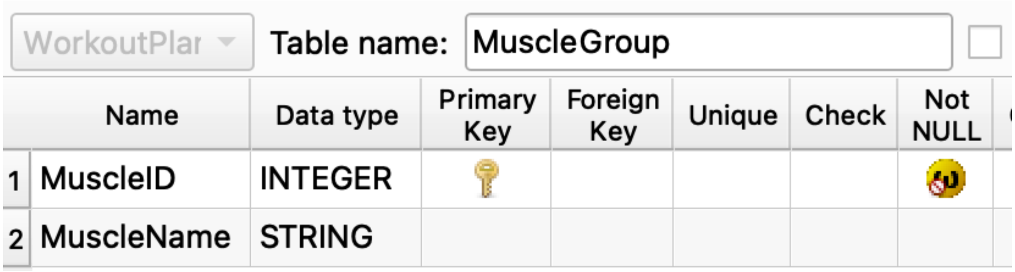
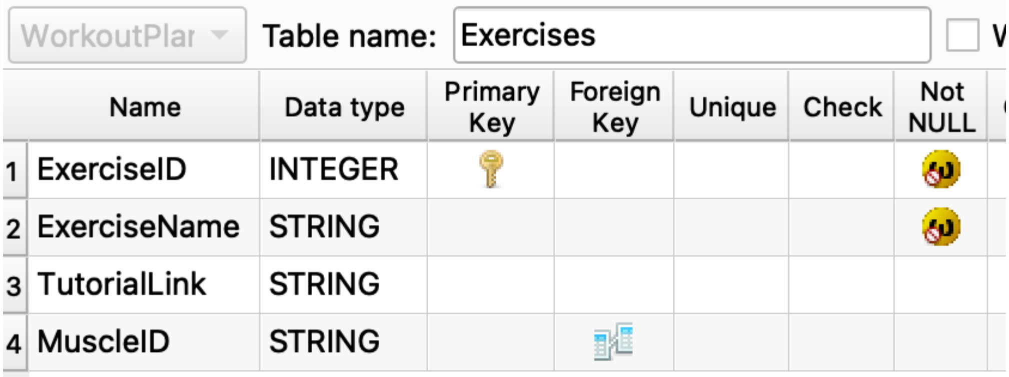


Table name: MuscleGroup □

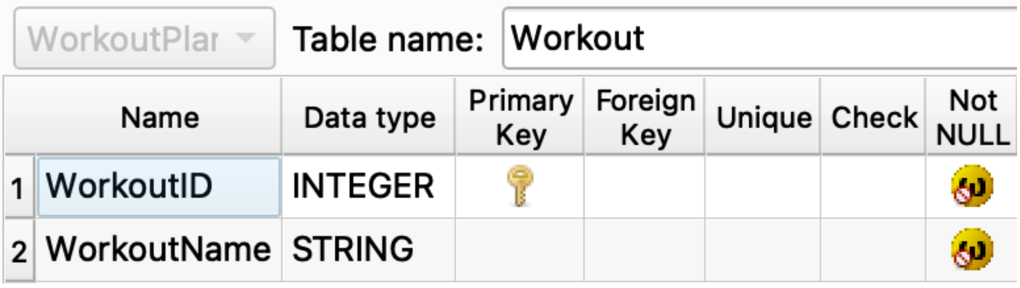
	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL
1	MuscleID	INTEGER	🔑				✓
2	MuscleName	STRING					

Exercises Table


WorkoutPlar Table name: Exercises □ V

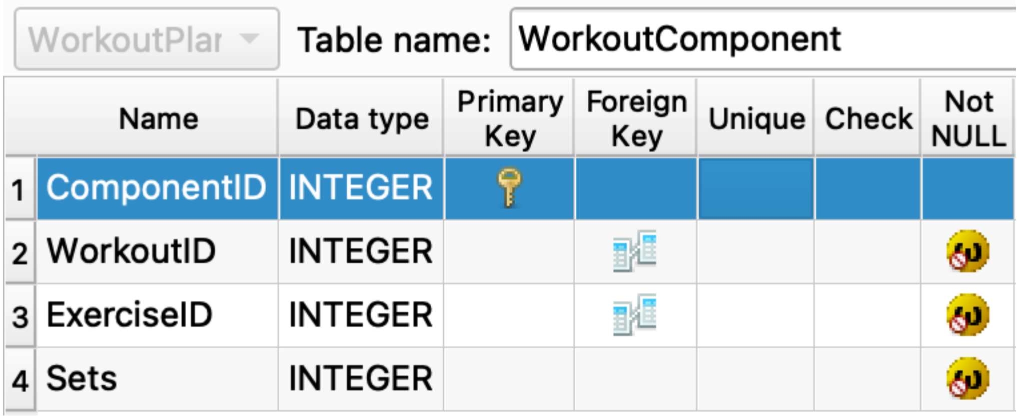
	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL
1	ExerciseID	INTEGER	✓				✓
2	ExerciseName	STRING					✓
3	TutorialLink	STRING					
4	MuscleID	STRING		✓			

Workout Table


WorkoutPlar Table name: Workout

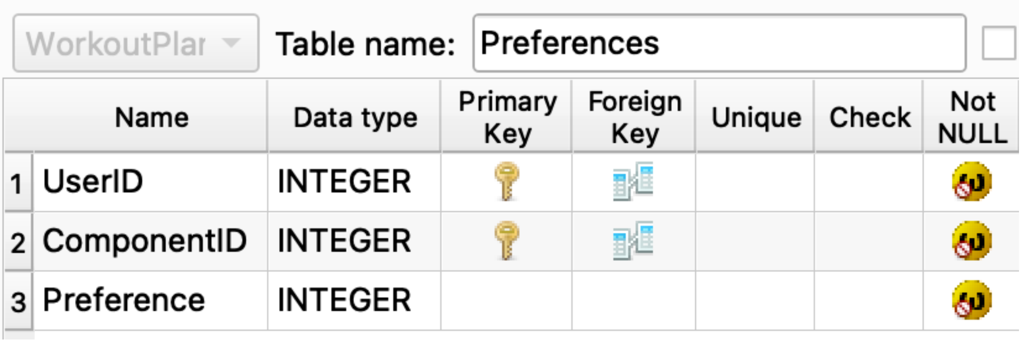
	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL
1	WorkoutID	INTEGER	✓				✓
2	WorkoutName	STRING					✓

WorkoutComponent Table


WorkoutPlar Table name: WorkoutComponent

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL
1	ComponentID	INTEGER	✓				
2	WorkoutID	INTEGER		✓			✓
3	ExerciseID	INTEGER		✓			✓
4	Sets	INTEGER					✓

Preferences Table


WorkoutPlar Table name: Preferences □

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL
1	UserID	INTEGER	✓	✓			✓
2	ComponentID	INTEGER	✓	✓			✓
3	Preference	INTEGER					✓

Times Table

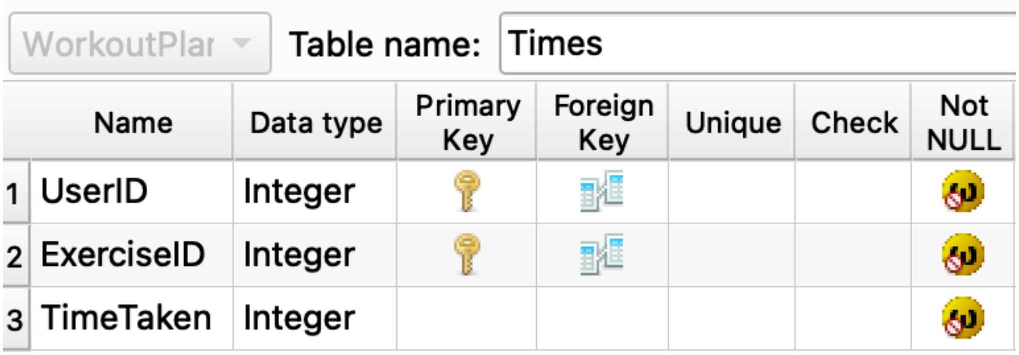


Table name: Times

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL
1	UserID	Integer	✓	✓			✓
2	ExerciseID	Integer	✓	✓			✓
3	TimeTaken	Integer					✓

SQL Statements

Function Name	Returned Data Type	SQL Statement
getUserNames()	String[]	SELECT FirstName, Surname FROM Users ORDER BY UserID ASC
getUsers()	User[]	SELECT UserID, FirstName, Surname FROM Users ORDER BY UserID ASC
getUserFromID(int uID)	User	SELECT UserID, FirstName, Surname FROM Users WHERE UserID = uID
getUserFromName(String firstName)	User	SELECT UserID, FirstName, Surname FROM Users WHERE FirstName = firstName
getNoOfUsers()	int	SELECT COUNT(UserID) From Users
zeroUsers()	boolean	SELECT COUNT(UserID) From Users
addUser(String firstName, String surname)	void	INSERT INTO Users(firstName, surname) VALUES(FirstName, Surname)
deleteUser(int uID)	void	DELETE FROM Users WHERE UserID = uID
modifyUser(int uID, String newFirst, newSurname)	void	UPDATE Users SET FirstName = firstName, Surname = surname WHERE UserID = uID
userExists(int uID)	boolean	SELECT COUNT(UserID) FROM Users WHERE UserID = uID
displayUsers()	void	SELECT UserID, FirstName, Surname FROM USERS
userHasTime(int uID, int exID)	boolean	SELECT COUNT(TimeTaken) FROM Times WHERE UserID = uID AND ExerciseID = exID

deleteUserTimesAndPreferences(int uID)	void	DELETE FROM Times WHERE UserID = uID;
		DELETE FROM Preferences WHERE UserID = uID

getExerciseFromID(int exID)	String	SELECT ExerciseName FROM Exercises WHERE ExerciseID = exID
getExercisesFromIDs(int[] exIDs)	String[]	SELECT ExerciseName FROM Exercises WHERE ExerciseID = exID
displayExercises()	void	SELECT ExerciseID, ExerciseName FROM Exercises
zeroWorkouts()	boolean	SELECT COUNT(WorkoutID) FROM Workout
workoutExists(int wID)	boolean	SELECT COUNT(ComponentID) WHERE WorkoutID = wID
workoutHasExercises(int wID)	boolean	SELECT COUNT(ComponentID) WHERE WorkoutID = wID
getWorkoutIDFromName (String WorkoutName)	int	SELECT WorkoutID FROM WORKOUT WHERE WorkoutName = workoutName
addWorkout(String workoutName)	void	INSERT INTO WORKOUT(WorkoutName) VALUES(workoutName)

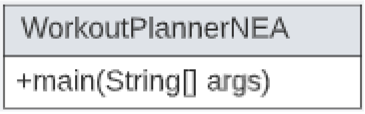
deleteWorkout(int wID)	void	DELETE FROM Workout WHERE WorkoutID = wID;
		DELETE FROM WorkoutComponent WHERE WorkoutID = wID;
		DELETE FROM Preferences WHERE WorkoutID = wID

addExToWorkout(int exID, int wID, int sets)	void	INSERT INTO WorkoutComponent(ExerciseID, WorkoutID, Sets) VALUES(exID, wID, sets)
deleteExerciseFrom Workout(int exID, int wID)	void	DELETE FROM WorkoutComponent WHERE ExerciseID = exID AND WorkoutID = wID
exerciseInWorkout(int exID, int wID)	boolean	SELECT COUNT(ComponentID) FROM WorkoutComponent WHERE ExerciseID = exID AND WorkoutID = wID
changeSetsOfExercise(int exID, int wID, int newSets)	void	UPDATE WorkoutComponent SET Sets = newSets WHERE ExerciseID = exID AND WorkoutID = wID
getComponentID(int exID, int wID)	int	SELECT ComponentID FROM WorkoutComponent WHERE WorkoutID = " + wID + " AND ExerciseID = " + exID
displayWorkouts()	void	SELECT WorkoutID, WorkoutName FROM Workout
deleteWorkoutTimesAnd Preferences(int wID)	void	DELETE FROM Preferences WHERE ComponentID = (SELECT ComponentID FROM WorkoutComponent WHERE WorkoutID = deleteID)
displayExercisesInWorkout (int wID)	void	SELECT Exercises.ExerciseName, WorkoutComponent.Sets, WorkoutComponent.ExerciseID FROM Exercises, WorkoutComponent WHERE Exercises.ExerciseID = WorkoutComponent.ExerciseID AND WorkoutComponent.WorkoutID = wID
getExerciseIDsFrom Workout(int wID)	int[]	SELECT ExerciseID FROM WorkoutComponent WHERE WorkoutID = wID
getExercisesInWorkout	String[]	SELECT Exercises.ExerciseName FROM Exercises INNER JOIN

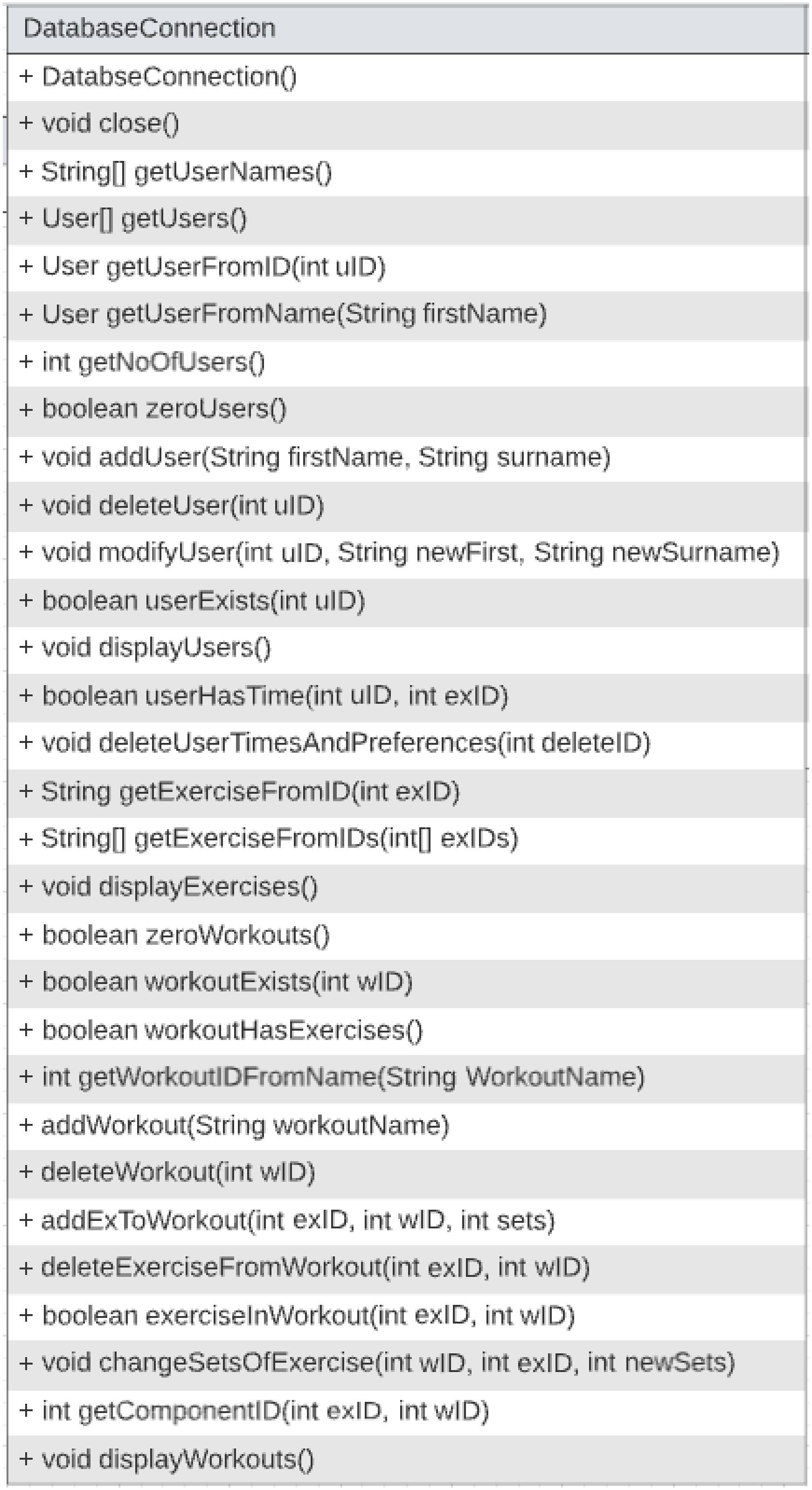
(int wID)		WorkoutComponent ON WorkoutComponent.ExerciseID = Exercises.ExerciseID AND WorkoutComponent.WorkoutID = wID
getNoOfExercisesInWorkout(int wID)	int	SELECT COUNT(ExerciseID) FROM WorkoutComponent WHERE WorkoutID = wID
addTime(int uID, int exID)	void	INSERT INTO Times(UserID, ExerciseID, TimeTaken) VALUES(uID, exID, time)
userHasTime(int uID, int exID)	boolean	SELECT COUNT(TimeTaken) FROM Times WHERE UserID = uID AND ExerciseID = exID
addPreference(int uID, int compID, int preference)	void	INSERT INTO Preferences(UserID, ComponentID, Preference) VALUES(uID, compID, preference)
getPreferences(int wID)	int[][]	SELECT ExerciseID FROM WorkoutComponent, Preferences WHERE Preferences.UserID = users[x].getID() AND Preferences.ComponentID = WorkoutComponent.ComponentID AND WorkoutComponent.WorkoutID = wID ORDER BY Preference ASC");
getTimes(int wID)	int[][]	SELECT UserID FROM Times WHERE ExerciseID = currentexID ORDER BY TimeTaken ASC
getRequiredTimes()	int[]	SELECT DISTINCT(ExerciseID) FROM WorkoutComponent
getNoOfComponents()	int	SELECT COUNT(ComponentID) FROM WorkoutComponent
getWorkoutComponents()	Workout Component[]	SELECT ComponentID, WorkoutID, ExerciseID FROM WorkoutComponent

Class Models

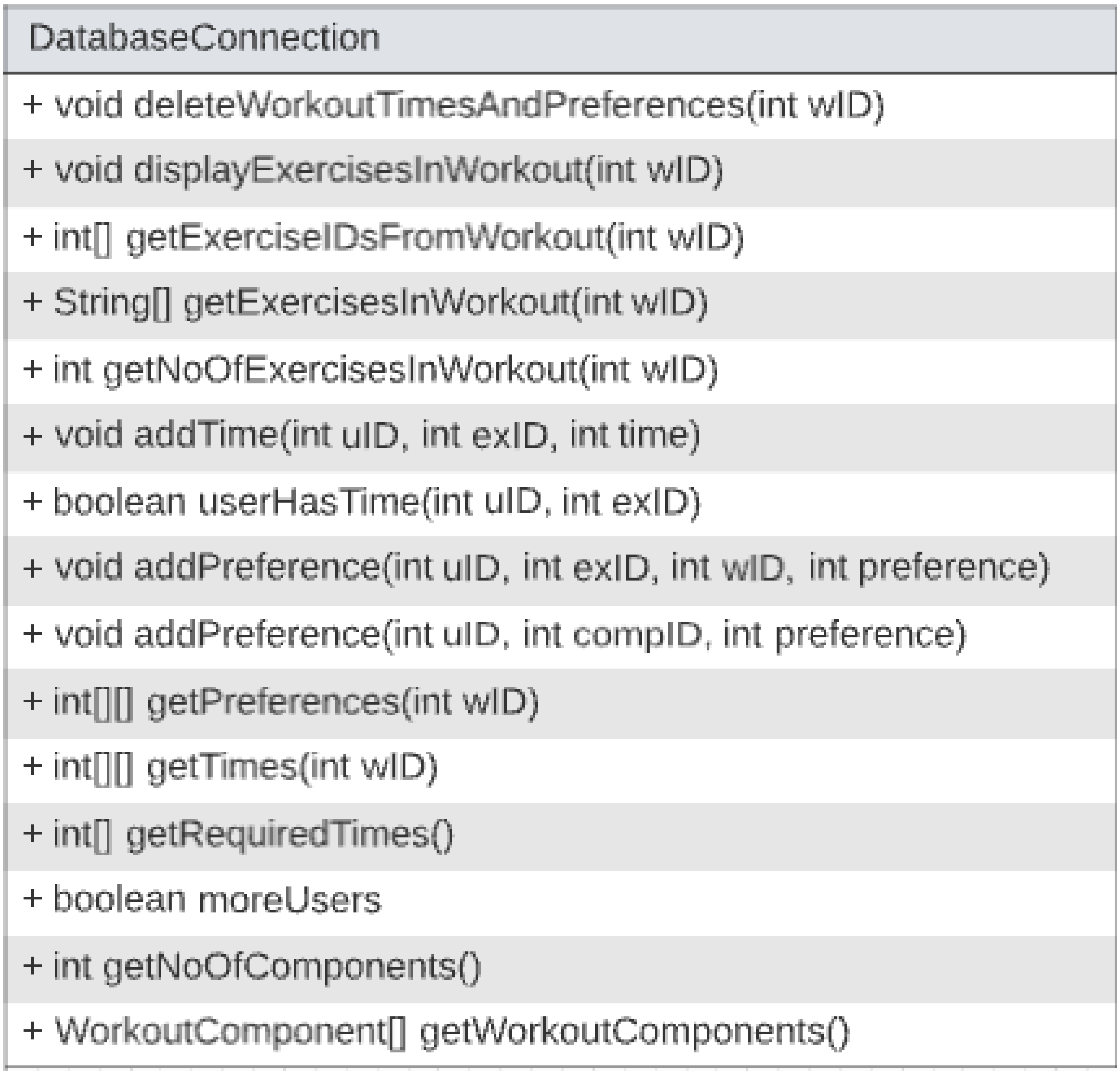
WorkoutPlannerNEA



DatabaseConnection

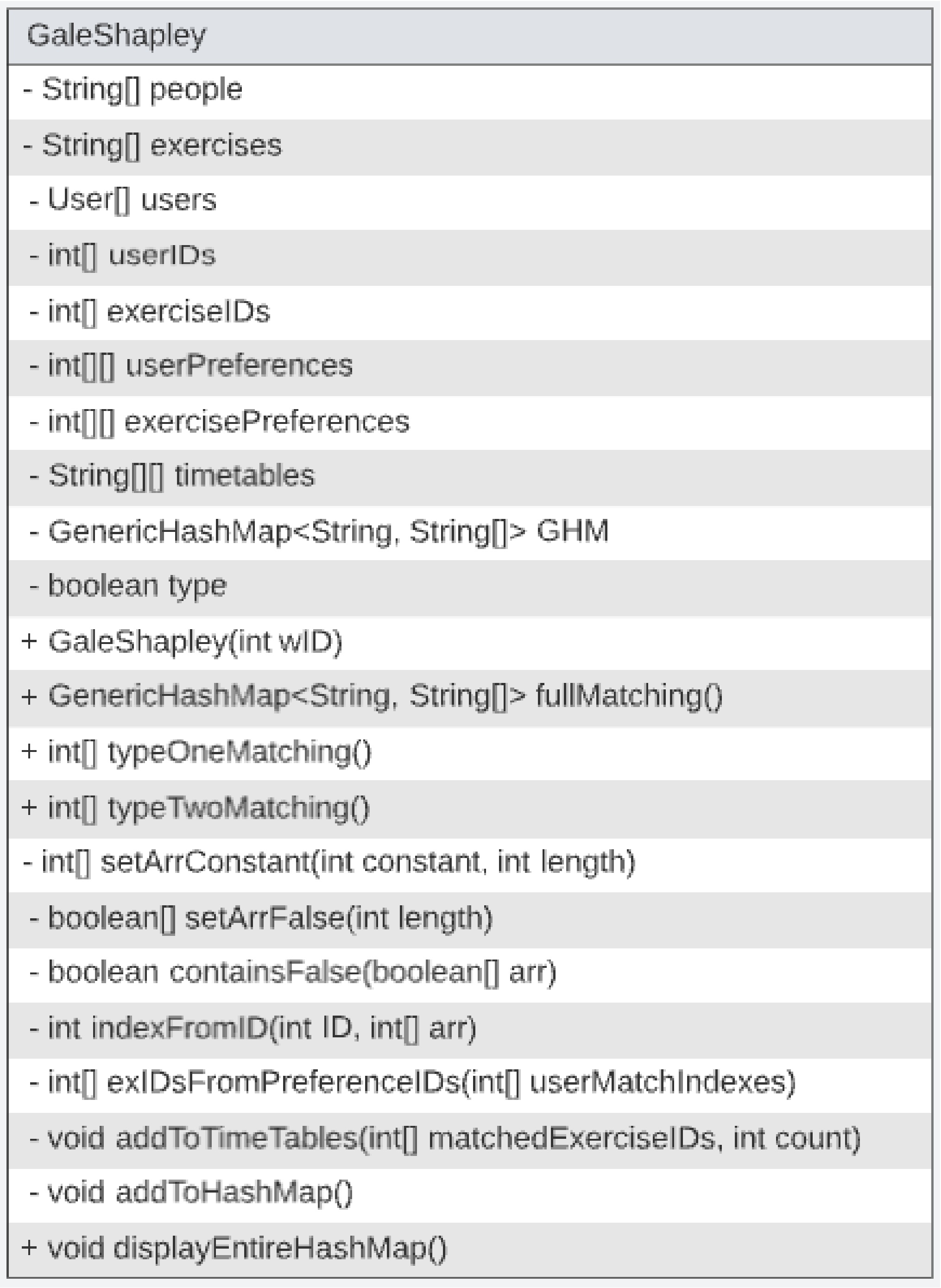


DatabaseConnection
+ DatabseConnection()
+ void close()
+ String[] getUserNames()
+ User[] getUsers()
+ User getUserFromID(int uID)
+ User getUserFromName(String firstName)
+ int getNoOfUsers()
+ boolean zeroUsers()
+ void addUser(String firstName, String surname)
+ void deleteUser(int uID)
+ void modifyUser(int uID, String newFirst, String newSurname)
+ boolean userExists(int uID)
+ void displayUsers()
+ boolean userHasTime(int uID, int exID)
+ void deleteUserTimesAndPreferences(int deleteID)
+ String getExerciseFromID(int exID)
+ String[] getExerciseFromIDs(int[] exIDs)
+ void displayExercises()
+ boolean zeroWorkouts()
+ boolean workoutExists(int wID)
+ boolean workoutHasExercises()
+ int getWorkoutIDFromName(String WorkoutName)
+ addWorkout(String workoutName)
+ deleteWorkout(int wID)
+ addExToWorkout(int exID, int wID, int sets)
+ deleteExerciseFromWorkout(int exID, int wID)
+ boolean exerciseInWorkout(int exID, int wID)
+ void changeSetsOfExercise(int wID, int exID, int newSets)
+ int getComponentID(int exID, int wID)
+ void displayWorkouts()



DatabaseConnection
+ void deleteWorkoutTimesAndPreferences(int wID)
+ void displayExercisesInWorkout(int wID)
+ int[] getExerciseIDsFromWorkout(int wID)
+ String[] getExercisesInWorkout(int wID)
+ int getNoOfExercisesInWorkout(int wID)
+ void addTime(int uID, int exID, int time)
+ boolean userHasTime(int uID, int exID)
+ void addPreference(int uID, int exID, int wID, int preference)
+ void addPreference(int uID, int compID, int preference)
+ int[][] getPreferences(int wID)
+ int[][] getTimes(int wID)
+ int[] getRequiredTimes()
+ boolean moreUsers
+ int getNoOfComponents()
+ WorkoutComponent[] getWorkoutComponents()

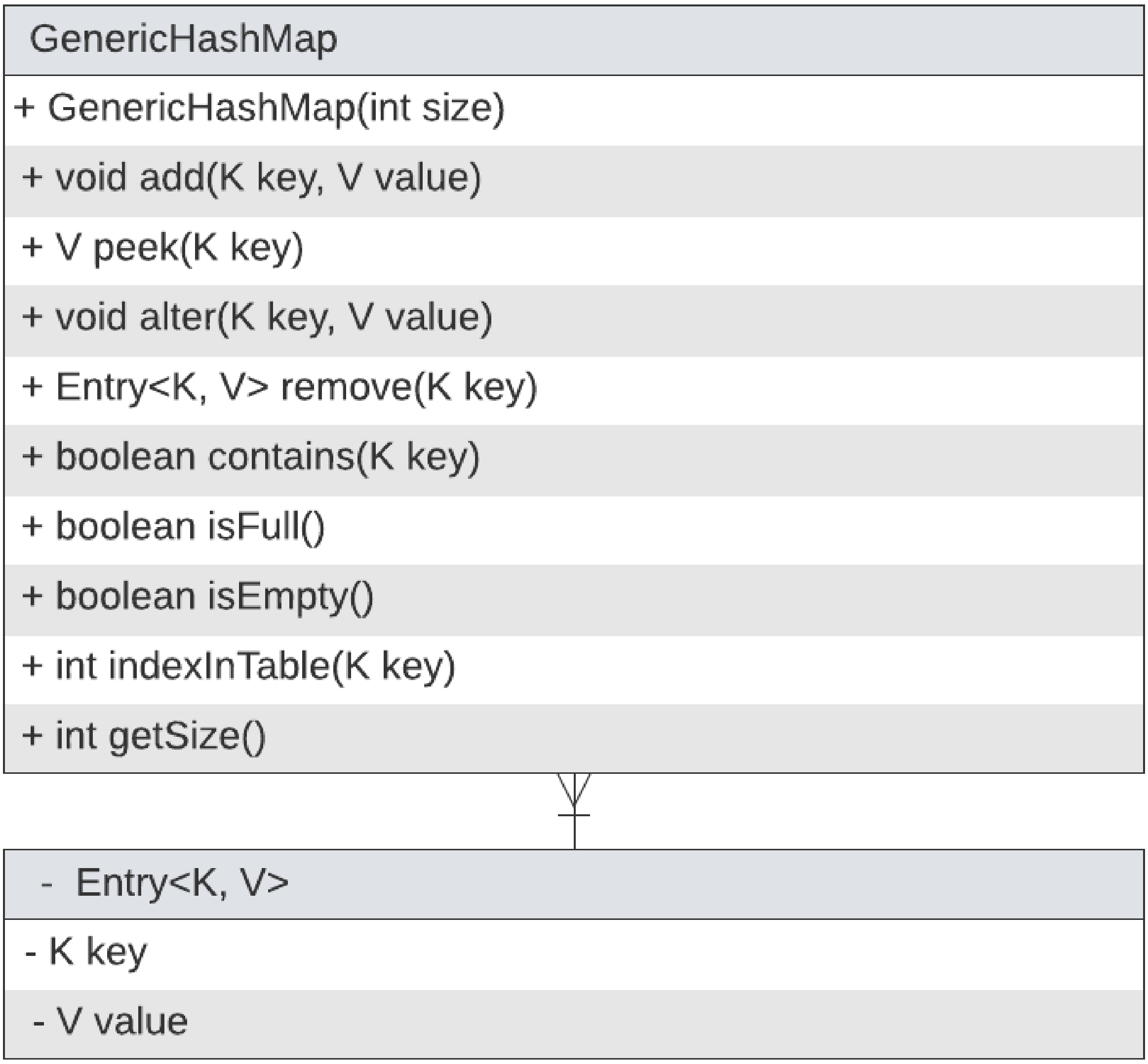
GaleShapley



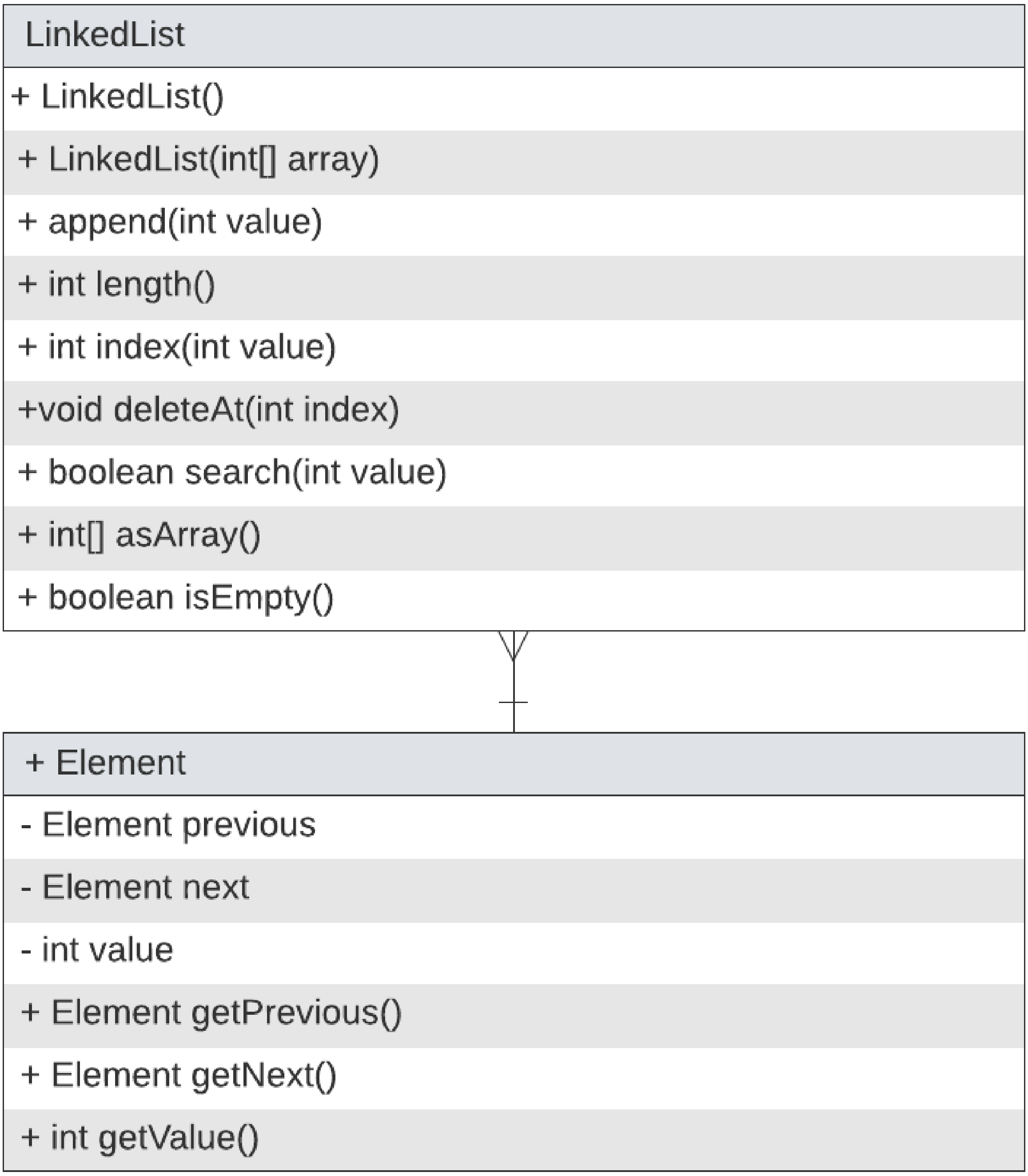
GaleShapley

- String[] people
- String[] exercises
- User[] users
- int[] userIDs
- int[] exerciseIDs
- int[][] userPreferences
- int[][] exercisePreferences
- String[][] timetables
- GenericHashMap<String, String[]> GHM
- boolean type
+ GaleShapley(int wID)
+ GenericHashMap<String, String[]> fullMatching()
+ int[] typeOneMatching()
+ int[] typeTwoMatching()
- int[] setArrConstant(int constant, int length)
- boolean[] setArrFalse(int length)
- boolean containsFalse(boolean[] arr)
- int indexFromID(int ID, int[] arr)
- int[] exIDsFromPreferenceIDs(int[] userMatchIndexes)
- void addToTimeTables(int[] matchedExerciseIDs, int count)
- void addToHashMap()
+ void displayEntireHashMap()

GenericHashMap


GenericHashMap
+ GenericHashMap(int size)
+ void add(K key, V value)
+ V peek(K key)
+ void alter(K key, V value)
+ Entry<K, V> remove(K key)
+ boolean contains(K key)
+ boolean isFull()
+ boolean isEmpty()
+ int indexInTable(K key)
+ int getSize()
- Entry<K, V>
- K key
- V value

LinkedList



LinkedList

+ LinkedList()

+ LinkedList(int[] array)

+ append(int value)

+ int length()

+ int index(int value)

+void deleteAt(int index)

+ boolean search(int value)

+ int[] asArray()

+ boolean isEmpty()

+ Element

- Element previous

- Element next

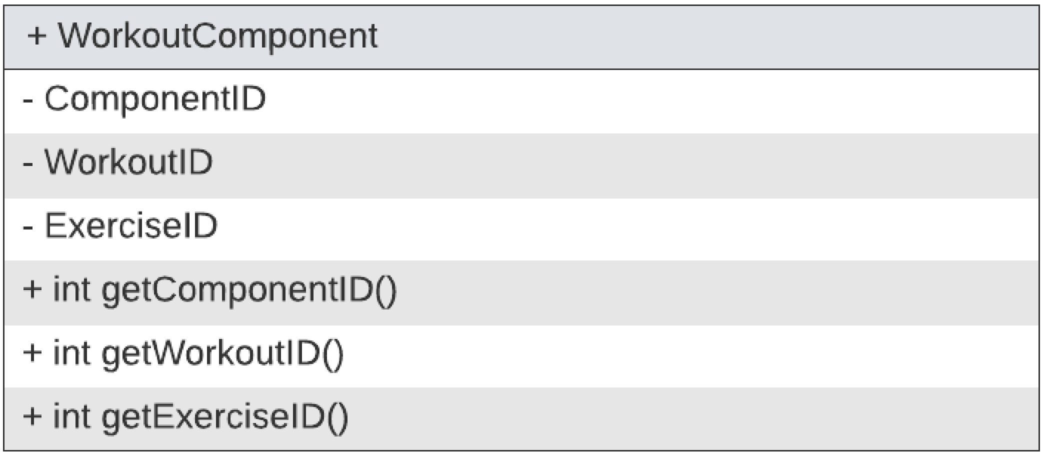
- int value

+ Element getPrevious()

+ Element getNext()

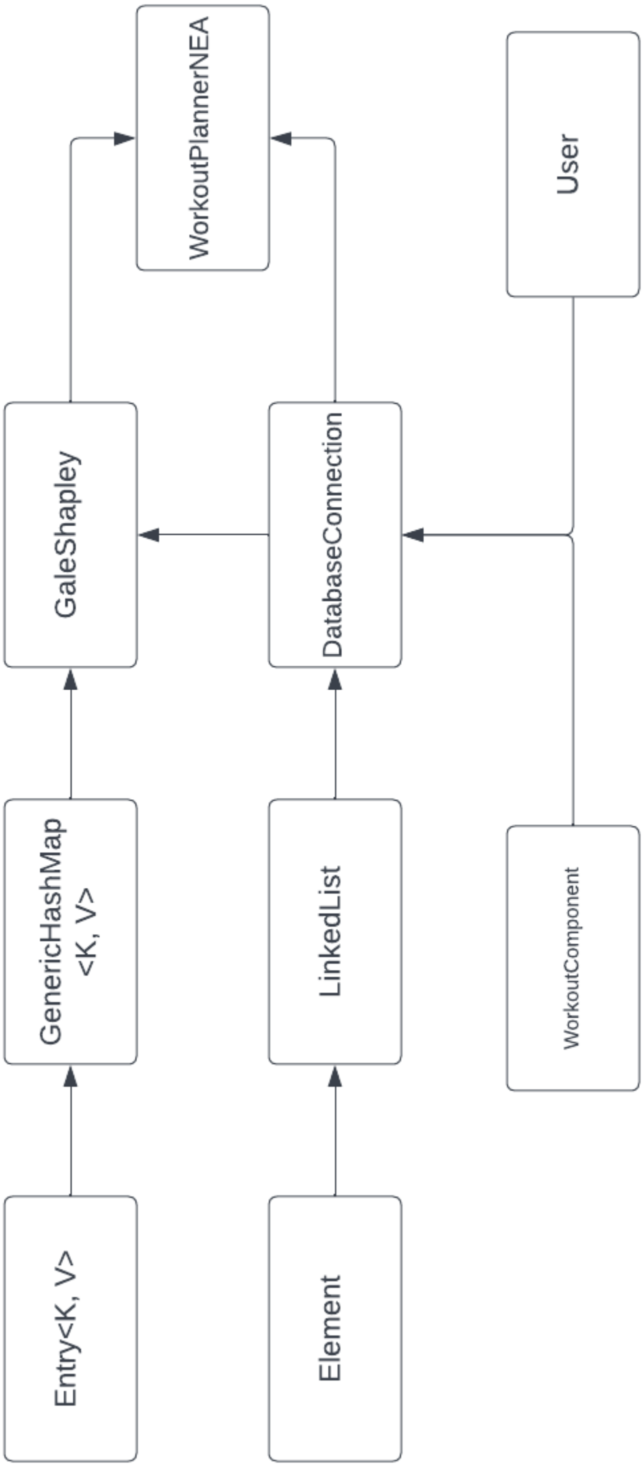
+ int getValue()

WorkoutComponent

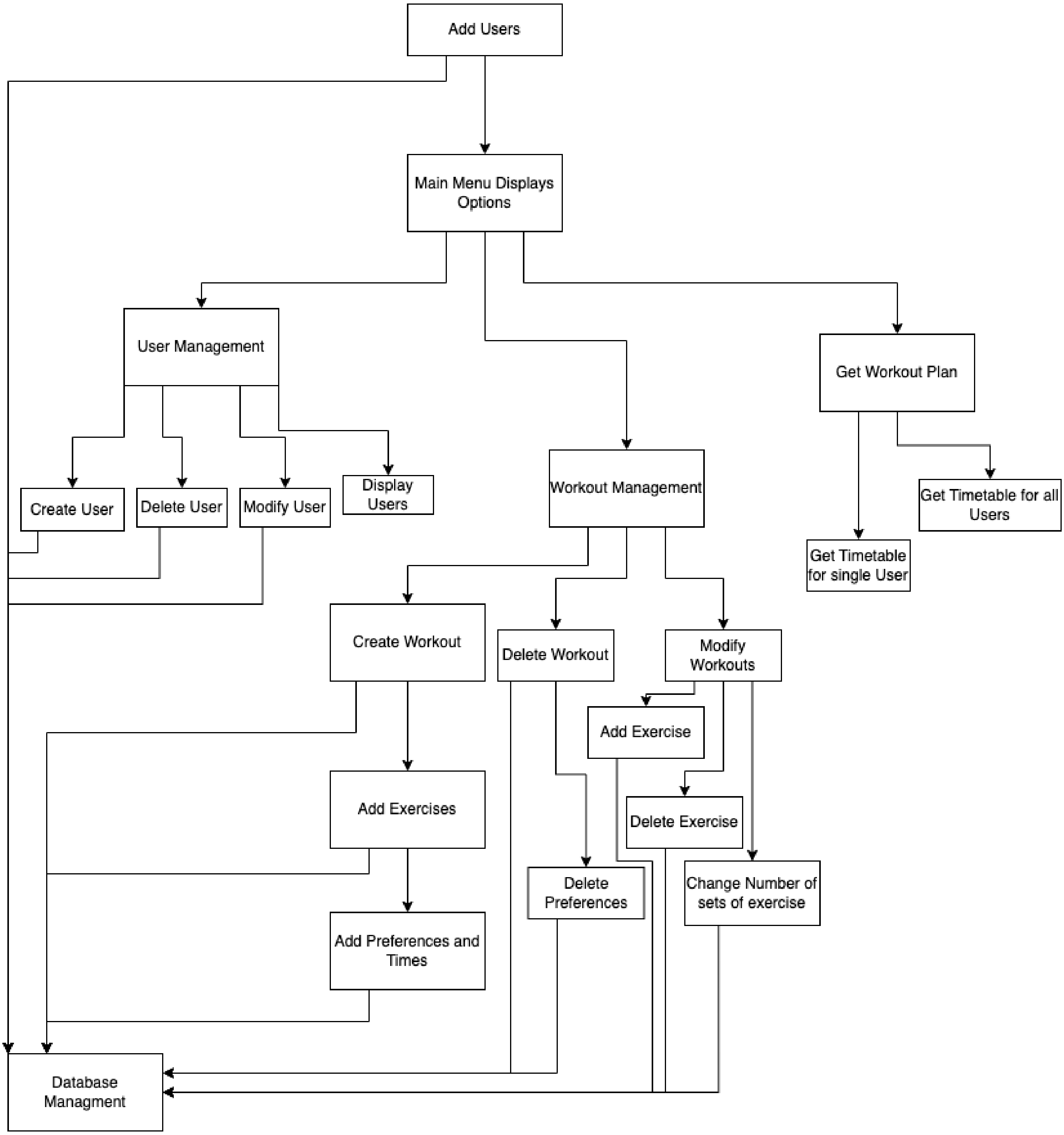


+ WorkoutComponent
- ComponentID
- WorkoutID
- ExerciseID
+ int getComponentID()
+ int getWorkoutID()
+ int getExerciseID()

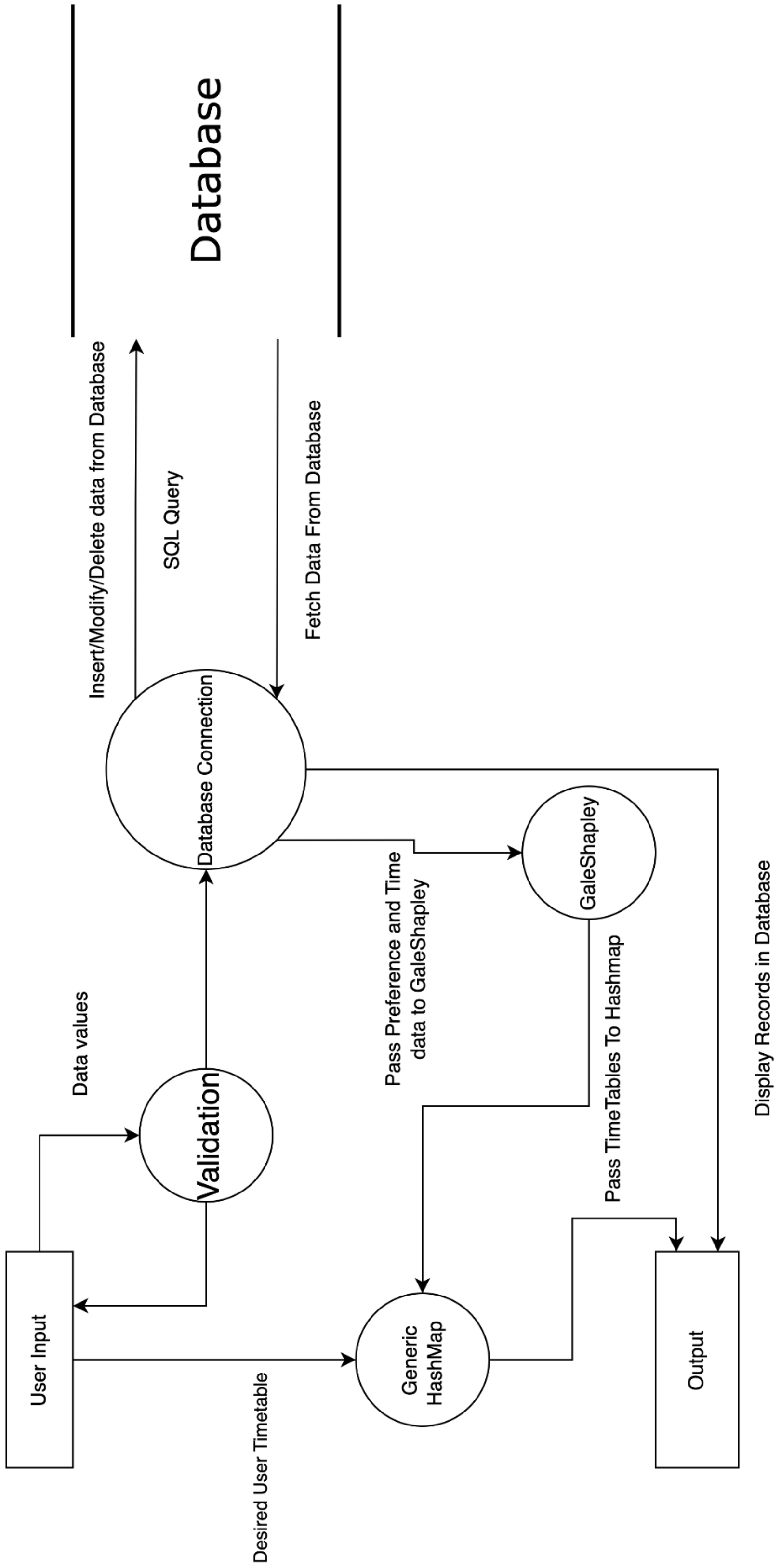
Class Relationships Diagram



System Structure Diagram (Main Menu)



Data Flow Diagram



3 - Technical Solution

WorkoutPlannerNEA

```
package workoutplannernea;

import java.util.InputMismatchException;
import java.util.Scanner;

public class WorkoutPlannerNEA {
    public static void main(String[] args) {
        DatabaseConnection dbconn = new DatabaseConnection();
        int exitKey = 0;
        boolean typeTestPassed;
        Scanner sc = new Scanner(System.in);

        System.out.println("Hello new user and welcome to this
workout planner. To get started \nwe are going to need you to
enter the names of everyone in your gym group.");
        System.out.println("To get started please enter how many
people are in your group \n(If you have already entered your group
in then please enter 0)");
        int noOfUsers = -1;
        typeTestPassed = false;
        while(!typeTestPassed || noOfUsers < 0){
            try{
                typeTestPassed = false;
                System.out.println("Please enter the number of
users you want to add: ");
                noOfUsers = sc.nextInt();
                typeTestPassed = true;
            }
            catch(InputMismatchException e){
                System.out.println("You must enter a number");
                sc.next();
            }
        }

        for (int i = 0; i < noOfUsers; i++) {
            System.out.println("Please enter User " + (i + 1) + "'s
first name");
            String first = sc.next().trim();
            System.out.println("Please enter User " + (i + 1) + "'s 
surname");
            String surname = sc.next().trim();
            dbconn.addUser(first, surname);
            if (!dbconn.zeroWorkouts()){
                int[] exerciseIDs = dbconn.getRequiredTimes();
                String[] requiredTimes = 
dbconn.getExerciseFromIDs(exerciseIDs);
                for(int j = 0; j < exerciseIDs.length; j++){
                    System.out.println("Please enter how long it 
takes you to do " + requiredTimes[j] + ": ");
                    int time = -1;
                    typeTestPassed = false;
                    while(!typeTestPassed || time < 1){
                        try{
                            typeTestPassed = false;
                            time = sc.nextInt();
                            typeTestPassed = true;
                        }
                        catch(InputMismatchException e){
                            System.out.println("Please enter a 
valid time: ");
                            sc.next();
                        }
                    }

dbconn.addTime(dbconn.getUserFromName(first).getID(), 
exerciseIDs[j], time);
                }

                WorkoutComponent[] requiredPreferences = 
dbconn.getWorkoutComponents();
                int uID = dbconn.getUserFromName(first).getID();
                for(WorkoutComponent wc : requiredPreferences){
                    System.out.println("Please enter your 
preference for " + dbconn.getExerciseFromID(wc.getExerciseID()) + 
" in workout " + wc.getWorkoutID());
                    int preference = -1;
                    typeTestPassed = false;
                    while(!typeTestPassed || preference < 1 || 
preference > 10){
                        try{
                            typeTestPassed = false;
                            preference = sc.nextInt();
                            typeTestPassed = true;
                        }
                        catch(InputMismatchException e){
                            System.out.println("Please enter a
valid time: ");
                            sc.next();
                        }
                    }
                    dbconn.addPreference(uID, wc.getComponentID(),
preference);
                }
            }
        }
        dbconn.displayUsers();

        System.out.println("Thank you for entering you users in.
Now what do you want to do?");
        while(exitKey == 0){
            System.out.println("Press 1 for user management");
            System.out.println("Press 2 for workout management");
            System.out.println("Press 3 to get schedules for your
group");
            typeTestPassed = false;
            int firstChoice = -1;
            while(!typeTestPassed || firstChoice < 1 || firstChoice
> 3){
                try{
                    typeTestPassed = false;
                    System.out.println("Please enter you choice:
");
                    firstChoice = sc.nextInt();
                    typeTestPassed = true;
                }
                catch(InputMismatchException e){
                    System.out.println("You must enter a valid
number");
                    sc.next();
                }
            }
            int secondChoice = -1;
            switch (firstChoice) {
                case 1:
                    System.out.println("Press 1 to add a user");
                    System.out.println("Press 2 to delete a user");
                    System.out.println("Press 3 to modify a user");
                    System.out.println("Press 4 to display users");

                    typeTestPassed = false;
                    while(!typeTestPassed || secondChoice < 1 ||
 secondChoice > 4){
                        try{
                            typeTestPassed = false;
                            System.out.println("Please enter your
 choice");

                            secondChoice = sc.nextInt();
                            typeTestPassed = true;
                        }
                        catch(InputMismatchException e){
                            System.out.println("You must enter a
 valid number");

                            sc.next();
                        }
                    }

                    switch (secondChoice) {
                        case 1:
                            System.out.println("Please enter the
 users first name: ");

                            String first = sc.next().trim();
                            System.out.println("Please enter the
 users last name: ");

                            String last = sc.next().trim();
                            dbconn.addUser(first, last);

                            if (!dbconn.zeroWorkouts()){
                                int[] exerciseIDs =
 dbconn.getRequiredTimes();

                                String[] requiredTimes =
 dbconn.getExerciseFromIDs(exerciseIDs);
                                for(int i = 0; i <
 exerciseIDs.length; i++){
```

```
                                   System.out.println("Please 
enter how long it takes you to do " + requiredTimes[i] + ": ");
                                   int time = -1;
                                   typeTestPassed = false;
                                   while(!typeTestPassed || time < 
1){
                                       try{
                                           typeTestPassed = false;

System.out.println("Please enter your time");
                                           time = sc.nextInt();
                                           typeTestPassed = true;
                                       }

catch(InputMismatchException e){
                                           System.out.println("You 

must enter a valid time");

                                           sc.next();
                                       }
                                   }

dbconn.addTime(dbconn.getUserFromName(first).getID(), 
exerciseIDs[i], time);
                               }

                               WorkoutComponent[] 
requiredPreferences = dbconn.getWorkoutComponents();
                               int uID = 
dbconn.getUserFromName(first).getID();
                               for(WorkoutComponent wc : 
requiredPreferences){
                                   System.out.println("Please 
enter your preference for " + 
dbconn.getExerciseFromID(wc.getExerciseID()) + " in workout " + 
wc.getWorkoutID());
                                   int preference = -1;
                                   typeTestPassed = false;
                                   while(!typeTestPassed || 
preference < 1 || preference > 10){
                                       try{
                                           typeTestPassed = false;
System.out.println("Please enter your preference: ");
                                           preference =
sc.nextInt();

                                           typeTestPassed = true;
                                       }

catch(InputMismatchException e){
                                           System.out.println("You
mist enter a valid preference");
                                           sc.next();
                                       }
                                   }
                                   dbconn.addPreference(uID,
wc.getComponentID(), preference);
                               }
                           }
                           break;

                       case 2:
                           if(dbconn.zeroUsers()){
                               System.out.println("There are no
users to delete");
                               break;
                           }
                           dbconn.displayUsers();

                           System.out.println("Please enter the
UserID of the user you want to delete: ");
                           int deleteID = -1;
                           typeTestPassed = false;
                           while(!typeTestPassed ||
!dbconn.userExists(deleteID)){
                               try{
                                   typeTestPassed = false;
                                   System.out.println("Please
enter a UserID: ");
                                   deleteID = sc.nextInt();
                                   typeTestPassed = true;
                               }
                               catch(InputMismatchException e){
                                   System.out.println("You must
enter a valid UserID");
```

```
                                   sc.next();
                               }
                           }

                           dbconn.deleteUser(deleteID);

dbconn.deleteUserTimesAndPreferences(deleteID);
                           break;

                       case 3:
                           if(dbconn.zeroUsers()){
                               System.out.println("There are no
users to modify");
                               break;
                           }
                           dbconn.displayUsers();

                           System.out.println("Please enter the
UserID of the user you want to modify");
                           int modifyID = -1;
                           typeTestPassed = false;
                           while(!typeTestPassed ||
!dbconn.userExists(modifyID)){
                               try{
                                   typeTestPassed = false;
                                   System.out.println("Please
enter a UserID: ");

                                   modifyID = sc.nextInt();
                                   typeTestPassed = true;
                               }
                               catch(InputMismatchException e){
                                   System.out.println("You must
enter a valid UserID");

                                   sc.next();
                               }
                           }
                           System.out.println("Please enter the
new first name: ");

                           String newFirst = sc.next();
                           System.out.println("Please enter the
new last name: ");

                           String newLast = sc.next();
```

```
                        dbconn.modifyUser(modifyID, newFirst,
newLast);

                    case 4:
                        if(dbconn.zeroUsers())
System.out.println("There are no users in your group.");
                        else dbconn.displayUsers();
                        break;
                }
                break;
            case 2:
                System.out.println("Press 1 to create a
workout");
                System.out.println("Press 2 to delete a
workout");
                System.out.println("Press 3 to modify a
workout");
                System.out.println("Press 4 to display
workouts");
                System.out.println("Press 5 to see the
exercises in a workout");
                typeTestPassed = false;
                while(!typeTestPassed || secondChoice < 1 ||
secondChoice > 5){
                    try{
                        typeTestPassed = false;
                        System.out.println("Please enter your
choice: ");

                        secondChoice = sc.nextInt();
                        typeTestPassed = true;
                    }
                    catch(InputMismatchException e){
                        System.out.println("You must enter a
valid number");

                        sc.next();
                    }
                }

                switch (secondChoice) {
                    case 1:
                        System.out.println("Please enter the
name of the new workout");
                            String workoutName = sc.next();
                            dbconn.addWorkout(workoutName);

                            int workoutID = 
dbconn.getWorkoutIDFromName(workoutName);
                            System.out.println("How many exercises 
are going to be in this workout? ");
                            int noOfExercises = -1;
                            typeTestPassed = false;
                            while(!typeTestPassed || noOfExercises 
< 1){
                                try{
                                    typeTestPassed = false;
                                    System.out.println("Please 
enter a number: ");

                                    noOfExercises = sc.nextInt();
                                    typeTestPassed = true;
                                }
                                catch(InputMismatchException e){
                                    System.out.println("You must 
enter a valid number");

                                    sc.next();
                                }
                            }

                            dbconn.displayExercises();

                            for (int i = 0; i < noOfExercises; i++)
{
                                System.out.println("Please enter 
the ExerciseID of the (next) exercise you want to add: ");
                                int exID = -1;
                                while(!typeTestPassed || 
!dbconn.exerciseExists(exID) || dbconn.exerciseInWorkout(exID, 
workoutID)){
                                    try{
                                        typeTestPassed = false;
                                        System.out.println("Please 
enter an ExerciseID: ");

                                        exID = sc.nextInt();
                                        typeTestPassed = true;
                                    }
                                    catch(InputMismatchException
e){
                                        System.out.println("You
must enter a valid exerciseID");
                                        sc.next();
                                    }
                                }
                                System.out.println("And how many
sets would you like to do? ");
                                int sets = -1;
                                typeTestPassed = false;
                                while(!typeTestPassed || sets < 1){
                                    try{
                                        typeTestPassed = false;
                                        System.out.println("Please
enter the number of sets: ");
                                        sets = sc.nextInt();
                                        typeTestPassed = true;
                                    }
                                    catch(InputMismatchException
e){
                                        System.out.println("You
must enter a valid number");
                                        sc.next();
                                    }
                                }
                                dbconn.addExToWorkout(exID,
workoutID, sets);
                                if(i == 0){
                                    System.out.println("An overview
on the preference system: The more you want an exercise towards
the beginning, the closer you enter a number to 1."
                                            + " If you want an
exercise towards the end of the workout then your number should be
closer to 10");
                                }
                                User[] users = dbconn.getUsers();
                                String exercise =
dbconn.getExerciseFromID(exID);
                                for(int j = 0; j < users.length;
j++){
```

```
System.out.println(users[j].getFirst() + ", please enter your 
preference for " + exercise + " : ");
                                   int preference = -1;
                                   typeTestPassed = false;
                                   while(!typeTestPassed || 
preference < 1 || preference > 10){
                                       try{
                                           typeTestPassed = false;

System.out.println("Please enter your preference: ");
                                           preference = 
sc.nextInt();

                                           typeTestPassed = true;
                                       }

catch(InputMismatchException e){
                                           System.out.println("You 
must enter a valid preference");

                                           sc.next();
                                       }
                                   }


if(!dbconn.userHasTime(users[j].getID(), exID)){
                                       System.out.println("Please 
enter how long it takes you to do " + sets + " sets of " + 
exercise + ": ");
                                       int time = -1;
                                       typeTestPassed = false;
                                       while(!typeTestPassed || 
time < 1){
                                           try{
                                               typeTestPassed = 
false;

System.out.println("Please enter a time: ");
                                               time = 
sc.nextInt();

                                               typeTestPassed = 
true;
```

```
                                           }

catch(InputMismatchException e){

System.out.println("You must enter a valid time");
                                               sc.next();
                                           }
                                       }

dbconn.addTime(users[j].getID(), exID, time);
                                   }

dbconn.addPreference(users[j].getID(), dbconn.getComponentID(exID,
workoutID), preference);
                               }


                           }
                           System.out.println("You have now added
" + workoutName + " to your workout list");
                           break;

                       case 2:
                           if(dbconn.zeroWorkouts()){
                               System.out.println("There are no
workouts to delete.");
                               break;
                           }
                           dbconn.displayWorkouts();

                           System.out.println("Please enter the
Workout ID of the workout you want to delete: ");
                           int deleteID = -1;
                           while(!typeTestPassed ||
!dbconn.workoutExists(deleteID)){
                               try{
                                   typeTestPassed = false;
                                   System.out.println("Please
enter a WorkoutID");
                                   deleteID = sc.nextInt();
                                   typeTestPassed = true;
                               }
                               catch(InputMismatchException e){
                                   System.out.println("You must 
enter a valid WorkoutID");

                                   sc.next();
                               }
                           }

dbconn.deleteWorkoutPreferences(deleteID);
                           dbconn.deleteComponents(deleteID);
                           dbconn.deleteWorkout(deleteID);

                           System.out.println("The workout with 
WorkoutID " + deleteID + " has been deleted");
                           break;

                       case 3:
                           if(dbconn.zeroWorkouts()){
                               System.out.println("There are no 
workouts to modify.");

                               break;
                           }
                           dbconn.displayWorkouts();

                           System.out.println("Please enter the 
Workout ID of the workout you want to modify: ");
                           int modifyID = -1;
                           while(!typeTestPassed || 
!dbconn.workoutExists(modifyID)){
                               try{
                                   typeTestPassed = false;
                                   System.out.println("Please 
enter a WorkoutID: ");

                                   modifyID = sc.nextInt();
                                   typeTestPassed = true;
                               }
                               catch(InputMismatchException e){
                                   System.out.println("You must 
enter a valid WorkoutID");

                                   sc.next();
                               }
                           }
                           System.out.println("Press 1 to add an 
exercise to this workout");
                           System.out.println("Press 2 to delete 
an exercise in this workout");
                           System.out.println("Press 3 to change 
the number of sets of a workout in this exercise");
                           typeTestPassed = false;
                           int thirdChoice = -1;
                           while(!typeTestPassed || thirdChoice < 
1 || thirdChoice > 3){
                               try{
                                   typeTestPassed = false;
                                   System.out.println("Please 
enter your choice: ");
                                   
                                   thirdChoice = sc.nextInt();
                                   typeTestPassed = true;
                               }
                               catch(InputMismatchException e){
                                   System.out.println("You must 
enter a valid choice");
                                   
                                   sc.next();
                               }
                           }
                           
                           switch (thirdChoice) {
                               case 1:
                                   dbconn.displayExercises();
                                   System.out.println("Please 
enter the ExerciseID of the exercise you want to add: ");
                                   int exID = -1;
                                   typeTestPassed = false;
                                   while(!typeTestPassed || 
!dbconn.exerciseExists(exID) || dbconn.exerciseInWorkout(exID, 
modifyID)){
                                       try{
                                           typeTestPassed = false;
                                           
System.out.println("Please enter an ExerciseID: ");
                                           exID = sc.nextInt();
                                           typeTestPassed = true;
                                       }
```

```
catch(InputMismatchException e){
                                           System.out.println("You
must enter a valid ExerciseID: ");
                                           sc.next();
                                       }
                                   }
                                   System.out.println("And how
many sets would you like to do? ");
                                   int sets = -1;
                                   typeTestPassed = false;
                                   while(!typeTestPassed || sets <
1){
                                       try{
                                           typeTestPassed = false;

System.out.println("Please enter how many sets you want to do: ");
                                           sets = sc.nextInt();
                                           typeTestPassed = true;
                                       }

catch(InputMismatchException e){
                                           System.out.println("You
must enter a valid number of sets");
                                           sc.next();
                                       }
                                   }
                                   dbconn.addExToWorkout(exID,
modifyID, sets);

                                   int compID =
dbconn.getComponentID(exID, modifyID);

                                   User[] users =
dbconn.getUsers();
                                   String newExercise =
dbconn.getExerciseFromID(exID);

                                   for(User u : users){

System.out.println(u.getFirst() + ", please enter your preference
for " + newExercise + ": ");
                                       int preference = -1;
                                       typeTestPassed = false;
                                       while(!typeTestPassed ||
preference < 1){
                                           try{
                                               typeTestPassed =
false;

System.out.println("Please enter your preference: ");
                                               preference =
sc.nextInt();
                                               typeTestPassed =
true;
                                           }

catch(InputMismatchException e){

System.out.println("You must enter a valid preference");
                                               sc.next();
                                           }
                                       }

dbconn.addPreference(u.getID(), compID, preference);
                                   }

                                   for(User u : users){

if(!dbconn.userHasTime(u.getID(), exID)){

System.out.println(u.getFirst() + ", please enter how long it
takes you to do " + sets + " sets of " + newExercise + ": ");
                                           int time = -1;
                                           typeTestPassed = false;
                                           while(!typeTestPassed
|| time < 1){
                                               try{
                                                   typeTestPassed
= false;

System.out.println("Please enter your time: ");
                                                   time =
sc.nextInt();
                                                   typeTestPassed
= true;
                                               }

catch(InputMismatchException e){

System.out.println("You must enter a valid time");
                                                   sc.next();
                                               }
                                           }

dbconn.addTime(u.getID(), exID, time);
                                       }
                                   }

                                   break;

                               case 2:

if(!dbconn.workoutHasExercises(modifyID)){
                                       System.out.println("This
workout has no exercises to delete");
                                       break;
                                   }

dbconn.displayExercisesInWorkout(modifyID);
                                   System.out.println("Enter the
Exercise ID of the exercise you want to delete: ");
                                   int delID = -1;
                                   typeTestPassed = false;
                                   while(!typeTestPassed ||
!dbconn.exerciseInWorkout(delID, modifyID)){
                                       try{
                                           typeTestPassed = false;

System.out.println("Please enter the ExerciseID: ");
                                           exID = sc.nextInt();
                                           typeTestPassed = true;
                                       }

catch(InputMismatchException e){
                                           System.out.println("You
must enter a valid exerciseID");
```

```
                                           sc.next();
                                       }
                                   }

dbconn.deleteExerciseFromWorkout(delID, modifyID);
                                   break;

                               case 3:

if(!dbconn.workoutHasExercises(modifyID)){
                                       System.out.println("This
workout has no exercises to modify");
                                       break;
                                   }

dbconn.displayExercisesInWorkout(modifyID);
                                   System.out.println("Please
enter the Exercise ID of the exercise you want to change: ");
                                   exID = -1;
                                   typeTestPassed = false;
                                   while(!typeTestPassed ||
!dbconn.exerciseInWorkout(exID, modifyID)){
                                       try{
                                           typeTestPassed = false;

System.out.println("Please enter a valid ExerciseID: ");
                                           exID = sc.nextInt();
                                           typeTestPassed = true;
                                       }

catch(InputMismatchException e){
                                           System.out.println("You

must enter a valid exerciseID");

                                           sc.next();
                                       }
                                   }
                                   System.out.println("What is the

new number of sets? ");

                                   int newSets = -1;
                                   typeTestPassed = false;
                                   while(!typeTestPassed ||

newSets < 1){
```

```
                                       try{
                                           typeTestPassed = false;

System.out.println("Please enter how many sets you want to do: ");
                                           newSets = sc.nextInt();
                                           typeTestPassed = true;
                                       }

catch(InputMismatchException e){
                                           System.out.println("You
must enter a valid number of sets");
                                           sc.next();
                                       }
                                   }

dbconn.changeSetsOfExercise(modifyID, exID, newSets);
                                   break;

                           }
                           break;

                       case 4:
                           if(dbconn.zeroWorkouts()){
                               System.out.println("There are no
workouts to display.");
                               break;
                           }
                           dbconn.displayWorkouts();
                           break;

                       case 5:
                           if(dbconn.zeroWorkouts()){
                               System.out.println("There are no
workouts to view.");
                               break;
                           }
                           dbconn.displayWorkouts();
                           System.out.println("Please enter the
Workout ID of the workout you want to display: ");
                           int displayID = -1;
                           typeTestPassed = false;
                           while(!typeTestPassed ||
!dbconn.workoutExists(displayID)){
                               try{
                                   typeTestPassed = false;
                                   System.out.println("Please
enter a WorkoutID: ");

                                   displayID = sc.nextInt();
                                   typeTestPassed = true;
                               }
                               catch(InputMismatchException e){
                                   System.out.println("You must
enter a valid WorkoutID");

                                   sc.next();
                               }
                           }

dbconn.displayExercisesInWorkout(displayID);
                           break;
                   }
                   break;
               case 3:
                   if(dbconn.zeroWorkouts()){
                       System.out.println("There are no workouts
to get schedules for.");
                       break;
                   }
                   User[] users = dbconn.getUsers();
                   dbconn.displayWorkouts();
                   System.out.println("Please enter the workout ID
of the workout you want schedules for: ");
                   int wID = -1;
                   typeTestPassed = false;
                   while(!typeTestPassed ||
!dbconn.workoutExists(wID)){
                       try{
                           typeTestPassed = false;
                           System.out.println("Please enter a
WorkoutID: ");

                           wID = sc.nextInt();
                           typeTestPassed = true;
                       }
                       catch(InputMismatchException e){
                           System.out.println("You must enter a
valid WorkoutID");
                           sc.next();
                       }
                   }
                   GaleShapley gs = new GaleShapley(wID);
                   GenericHashMap<String, String[]> timetables = 
gs.fullMatching();
                   dbconn.displayUsers();
                   System.out.println("Please enter the UserID of 
the timetable you want to see. Enter -1 to see all timetables: ");
                   int viewID = 0;
                   typeTestPassed = false;
                   while(!typeTestPassed || 
(!dbconn.userExists(viewID) && viewID != -1)){
                       try{
                           typeTestPassed = false;
                           System.out.println("Please enter your 
choice: ");

                           viewID = sc.nextInt();
                           typeTestPassed = true;
                       }
                       catch(InputMismatchException e){
                           System.out.println("You must enter a 
valid choice");

                           sc.next();
                       }
                   }
                   if(viewID == -1) gs.displayEntireHashMap();
                   else{
                       String firstName = 
dbconn.getUserFromID(viewID).getFirst();
                       String[] selectedTimetable = 
timetables.peek(firstName);
                       System.out.println(firstName + ": ");
                       for(String s : selectedTimetable){
                           System.out.println(s);
                       }
                   }
                   break;
           }
           System.out.println("Please enter 0 to go back to the 
main menu. To exit enter any other number: ");
```

```
            exitKey = sc.nextInt();
        }
        dbconn.close();
        System.out.println("Goodbye and Thank You");
    }
}
```

DatabaseConnection

```
package workoutplannernea;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.util.logging.Level;
import java.util.logging.Logger;

@SuppressWarnings("BooleanMethodIsAlwaysInverted")
public class DatabaseConnection {
   private Connection conn = null;

   public DatabaseConnection() {
       try
       {
           conn = 
DriverManager.getConnection("jdbc:sqlite:WorkoutPlanner.db");//Sp
ecify the database, since relative in the main project folder
           conn.setAutoCommit(false);
       }
       catch (Exception e)
       {
           System.err.println(e.getClass().getName() + ": " + 
e.getMessage());
           System.exit(0);
       }
   }
   public void close(){
       try
       {
           conn.close();
       }
       catch (SQLException ex)
       {
Logger.getLogger(DatabaseConnection.class.getName()).log(Level.SE
VERE, null, ex);
       }
   }

   public String[] getUserNames(){
       String[] users = new String[getNoOfUsers()];
       Statement stmt;
       ResultSet rs;
       int count = 0;

       try{
           stmt = conn.createStatement();
           rs = stmt.executeQuery("SELECT FirstName, Surname FROM
Users ORDER BY UserID ASC");
           while(rs.next()){
               String FirstName = rs.getString("FirstName");
               String Surname = rs.getString("Surname");
               String fullName = FirstName + " " + Surname;
               users[count] = fullName;
               count++;
           }
           stmt.close();
           rs.close();
       }
       catch(SQLException e){
           System.err.println(e.getClass().getName() + ": " +
e.getMessage());
       }
       return users;
   }

   public User[] getUsers(){
       User[] users = new User[getNoOfUsers()];
       Statement stmt;
       ResultSet rs;
       int count = 0;

       try{
           stmt = conn.createStatement();
           rs = stmt.executeQuery("SELECT UserID, FirstName, 
Surname FROM Users ORDER BY UserID ASC");
           while(rs.next()){
               int UserID = rs.getInt("UserID");
               String FirstName = rs.getString("FirstName");
               String Surname = rs.getString("Surname");
               User user = new User(UserID, FirstName, Surname);
               users[count] = user;
               count++;
           }
           stmt.close();
           rs.close();
       }
       catch(SQLException e){
           System.err.println(e.getClass().getName() + ": " + 
e.getMessage());
       }
       return users;
   }

   public User getUserFromID(int uID){
       Statement stmt;
       ResultSet rs;
       User user = null;
       try{
           stmt = conn.createStatement();
           rs = stmt.executeQuery("SELECT UserID, FirstName, 
Surname FROM Users WHERE UserID = " + uID);
           while(rs.next()){
               int UserID = rs.getInt("UserID");
               String FirstName = rs.getString("FirstName");
               String Surname = rs.getString("Surname");
               user = new User(UserID, FirstName, Surname);
           }
           stmt.close();
           rs.close();
       }
       catch(SQLException e){
           System.err.println(e.getClass().getName() + ": " +
e.getMessage());
       }

       return user;

   }

   public User getUserFromName(String firstname){
       Statement stmt;
       ResultSet rs;
       User user = null;
       try{
           stmt = conn.createStatement();
           rs = stmt.executeQuery("SELECT UserID, FirstName,
Surname FROM Users WHERE FirstName = '" + firstname + "'");
           String FirstName = rs.getString("FirstName");
           int UserID = rs.getInt("UserID");
           String Surname = rs.getString("Surname");
           user = new User(UserID, FirstName, Surname);
           stmt.close();
           rs.close();
       }
       catch(SQLException e){
           System.err.println(e.getClass().getName() + ":" +
e.getMessage());
       }
       return user;
   }

   public int getNoOfUsers(){
       Statement stmt;
       ResultSet rs;
       int number = 5;

       try{
           stmt = conn.createStatement();
           rs = stmt.executeQuery("SELECT COUNT(UserID) FROM
Users");
```

```
            number = rs.getInt("COUNT(UserID)");
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ":" + 
e.getMessage());
        }
        return number;
    }

    public void addUser(String FirstName, String Surname){
        Statement stmt;
        try{
            stmt = conn.createStatement();
            stmt.execute("INSERT INTO Users(FirstName, Surname) 
VALUES('" + FirstName + "', '" + Surname + "')");
            stmt.close();
            conn.commit();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ":" + 
e.getMessage());
        }
    }

    public void deleteUser(int UserID){
        Statement stmt;
        try{
            stmt = conn.createStatement();
            stmt.execute("DELETE FROM Users WHERE UserID = " + 
UserID);
            stmt.close();
            conn.commit();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ":" + 
e.getMessage());
        }
    }

    public void modifyUser(int uID, String firstName, String 
 surname){
        Statement stmt;
        String sql = "UPDATE Users SET FirstName = '" + firstName
 + "', Surname = '" + surname + "' WHERE UserID =" + uID;
        try{
            stmt = conn.createStatement();
            stmt.executeUpdate(sql);
            stmt.close();
            conn.commit();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ":" +
 e.getMessage());
        }
    }

    public void deleteUserTimesAndPreferences(int deleteID) {
 //This method is for when a user is deleted
        Statement stmt;
        try{
            stmt = conn.createStatement();
            stmt.execute("DELETE FROM Times WHERE UserID = " +
 deleteID);
            stmt.execute("DELETE FROM Preferences WHERE UserID = "
 + deleteID);
            stmt.close();
            conn.commit();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ":" +
 e.getMessage());
        }
    }

    public void deleteWorkoutPreferences(int deleteID){
        Statement stmt;
        try{
            stmt = conn.createStatement();
            stmt.execute("DELETE FROM Preferences WHERE
 ComponentID = (SELECT ComponentID FROM WorkoutComponent WHERE
 WorkoutID = " + deleteID + ")");
            stmt.close();
```

```
            conn.commit();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ":" +
e.getMessage());
        }
    }

    public void deleteComponents(int deleteID){
        Statement stmt;
        try{
            stmt = conn.createStatement();
            stmt.execute("DELETE FROM WorkoutComponent WHERE
WorkoutID = " + deleteID);
            stmt.close();
            conn.commit();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ":" +
e.getMessage());
        }
    }
    public boolean userExists(int uID){
        Statement stmt;
        ResultSet rs;
        int number = 0;
        try{
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT COUNT(UserID) FROM
Users WHERE UserID = " + uID);
            number = rs.getInt("Count(UserID)");
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ":" +
e.getMessage());
        }
        return 0 < number;
    }

    public boolean zeroUsers(){
        Statement stmt;
        ResultSet rs;
        int count = 1;

        try{
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT COUNT(UserID) FROM
Users");
            count = rs.getInt("COUNT(UserID)");
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
e.getMessage());
        }

        return count == 0;
    }

    public boolean userHasTime(int uID, int exID){
        Statement stmt;
        ResultSet rs;
        int number = 0;
        try{
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT COUNT(TimeTaken) FROM
Times WHERE UserID = " + uID + " AND ExerciseID = " + exID);
            number = rs.getInt("COUNT(TimeTaken)");
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ":" +
e.getMessage());
        }
        return number != 0;
    }

    public void displayUsers(){
        Statement stmt;
        ResultSet rs;
```

```
       try{
           stmt = conn.createStatement();
           rs = stmt.executeQuery("SELECT UserID, FirstName, 
Surname FROM USERS");
           while(rs.next()){
               System.out.println("UserID: " + 
rs.getInt("UserID") + ", First Name : " + 
rs.getString("FirstName") + ", LastName: " + 
rs.getString("Surname"));
           }
           stmt.close();
           rs.close();
       }
       catch(SQLException e){
           System.err.println(e.getClass().getName() + ":" + 
e.getMessage());
       }
   }

   public String getExerciseFromID(int exID){
       Statement stmt;
       ResultSet rs;
       String Name = "";
      try{
           stmt = conn.createStatement();
           rs = stmt.executeQuery("SELECT ExerciseName FROM 
Exercises WHERE ExerciseID = " + exID);
           Name = rs.getString("ExerciseName");
           stmt.close();
           rs.close();
       }
       catch(SQLException e){
           System.err.println(e.getClass().getName() + ": " + 
e.getMessage());
       }
      return Name;
   }

   public String[] getExerciseFromIDs(int[] IDs){
       String[] exercises = new String[IDs.length];
       for(int i = 0; i < IDs.length; i++){
           exercises[i] = getExerciseFromID(IDs[i]);
       }
       return exercises;
   }

   public int getComponentID(int exID, int wID){
       Statement stmt;
       ResultSet rs;
       int compID = -1;
       try{
           stmt = conn.createStatement();
           rs = stmt.executeQuery("SELECT ComponentID FROM 
WorkoutComponent WHERE WorkoutID = " + wID + " AND ExerciseID = " 
+ exID);
           compID = rs.getInt("ComponentID");
       }
       catch(SQLException e){
           System.err.println(e.getClass().getName() + ": " + 
e.getMessage());
       }
       return compID;
   }

   public boolean exerciseExists(int exID){
       Statement stmt;
       ResultSet rs;
       int number = 0;

       try{
           stmt = conn.createStatement();
           rs = stmt.executeQuery("SELECT COUNT(ExerciseID) FROM 
Exercises WHERE ExerciseID = " + exID);
           number = rs.getInt("COUNT(ExerciseID)");
           stmt.close();
           rs.close();
       }
       catch(SQLException e){
           System.err.println(e.getClass().getName() + ": " + 
e.getMessage());
       }
       return number != 0;
   }
```

```
    public void displayExercises(){
        Statement stmt;
        ResultSet rs;

        try{
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT ExerciseID,
ExerciseName FROM Exercises");
            while(rs.next()){
                int exID = rs.getInt("ExerciseID");
                String exName = rs.getString("ExerciseName");
                System.out.println("Exercise ID: " + exID + ",
Exercise Name: " + exName);
            }
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
e.getMessage());
        }
    }

    public void addWorkout(String workoutName){
        Statement stmt;

        try{
            stmt = conn.createStatement();
            stmt.execute("INSERT INTO WORKOUT(WorkoutName)
VALUES('" + workoutName + "')");
            stmt.close();
            conn.commit();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
e.getMessage());
        }
    }

    public void deleteWorkout(int wID){
        Statement stmt;
        try{
            stmt = conn.createStatement();
            stmt.execute("DELETE FROM Workout WHERE WorkoutID = "
+ wID);
            stmt.execute("DELETE FROM WorkoutComponent WHERE
WorkoutID = " + wID);
            stmt.close();
            conn.commit();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
e.getMessage());
        }
    }

    public void addExToWorkout(int exID, int wID, int sets){
        Statement stmt;

        try{
            stmt = conn.createStatement();
            stmt.executeUpdate("INSERT INTO
WorkoutComponent(ExerciseID, WorkoutID, Sets) VALUES(" + exID +
", " + wID + ", " + sets + ")");
            stmt.close();
            conn.commit();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
e.getMessage());
        }
    }

    public boolean exerciseInWorkout(int exID, int wID){
        Statement stmt;
        ResultSet rs;
        int number = 0;

        try{
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT COUNT(ComponentID) FROM
WorkoutComponent WHERE ExerciseID = " + exID + " AND WorkoutID =
 " + wID);
            number = rs.getInt("COUNT(ComponentID)");
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
 e.getMessage());
        }
        return number != 0;
    }

    public void deleteExerciseFromWorkout(int exID, int wID){
        Statement stmt;

        try{
            stmt = conn.createStatement();
            stmt.execute("DELETE FROM WorkoutComponent WHERE
 ExerciseID = " + exID + " AND WorkoutID = " + wID);
            stmt.close();
            conn.commit();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
 e.getMessage());
        }
    }

    public int getWorkoutIDFromName(String name) {
        Statement stmt;
        ResultSet rs;
        int number = -1;

        try{
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT WorkoutID FROM WORKOUT
 WHERE WORKOUTNAME = '" + name + "'");
            number = rs.getInt("WorkoutID");
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
```

```
           System.err.println(e.getClass().getName() + ": " +
e.getMessage());
       }

       return number;
   }

   public boolean workoutHasExercises(int wID){
       Statement stmt;
       ResultSet rs;
       int number = 0;

       try{
           stmt = conn.createStatement();
           rs = stmt.executeQuery("SELECT COUNT(ComponentID) FROM
WorkoutComponent WHERE WorkoutID = " + wID);
           number = rs.getInt("COUNT(ComponentID)");
           stmt.close();
           rs.close();
       }
       catch(SQLException e){
           System.err.println(e.getClass().getName() + ": " +
e.getMessage());
       }
       return number != 0;
   }

   public boolean workoutExists(int wID){
       Statement stmt;
       ResultSet rs;
       int number = 0;

       try{
           stmt = conn.createStatement();
           rs = stmt.executeQuery("SELECT COUNT(WorkoutID) FROM
Workout WHERE WorkoutID = " + wID);
           number = rs.getInt("COUNT(WorkoutID)");
           stmt.close();
           rs.close();
       }
       catch(SQLException e){
           System.err.println(e.getClass().getName() + ": " +
e.getMessage());
       }
       return number != 0;
   }

   public boolean zeroWorkouts(){
       Statement stmt;
       ResultSet rs;
       int count = 1;

       try{
           stmt = conn.createStatement();
           rs = stmt.executeQuery("SELECT COUNT(WorkoutID) FROM
Workout");
           count = rs.getInt("COUNT(WorkoutID)");
           stmt.close();
           rs.close();
       }
       catch(SQLException e){
           System.err.println(e.getClass().getName() + ": " +
e.getMessage());
       }

       return count == 0;
   }

   public void displayWorkouts(){
       Statement stmt;
       ResultSet rs;

       try{
           stmt = conn.createStatement();
           rs = stmt.executeQuery("SELECT WorkoutID, WorkoutName
FROM Workout");
           while(rs.next()){
               int wID = rs.getInt("WorkoutID");
               String wName = rs.getString("WorkoutName");
               System.out.println("Workout ID: " + wID + ",
Workout Name: " + wName);
           }
           stmt.close();
           rs.close();
```

```
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
e.getMessage());
        }
    }

    public void displayExercisesInWorkout(int wID){
        Statement stmt;
        ResultSet rs;

        try{
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT Exercises.ExerciseName,
WorkoutComponent.Sets, WorkoutComponent.ExerciseID FROM
Exercises, WorkoutComponent "
                    + "WHERE Exercises.ExerciseID =
WorkoutComponent.ExerciseID AND WorkoutComponent.WorkoutID = " +
wID);

            while(rs.next()){
                String eName = rs.getString("ExerciseName");
//ename = Exercise Name
                int exID = rs.getInt("ExerciseID");
                int sets = rs.getInt("Sets");
                System.out.println("Exercise ID : " + exID + ", "
+ sets + " sets of " + eName);
            }
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
e.getMessage());
        }
    }

    public void changeSetsOfExercise(int wID, int exID, int
newSets) {
        Statement stmt;

        try{
            stmt = conn.createStatement();
            stmt.executeUpdate("UPDATE WorkoutComponent SET Sets = 
" + newSets + " WHERE ExerciseID = " + exID + " AND WorkoutID = " 
+ wID);
            stmt.close();
            conn.commit();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " + 
e.getMessage());
        }
    }

    public int[] getExerciseIDsFromWorkout(int wID){
        Statement stmt;
        ResultSet rs;
        LinkedList linky = new LinkedList();

        try{
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT ExerciseID FROM 
WorkoutComponent WHERE WorkoutID = " + wID);

            while(rs.next()){
                int exID = rs.getInt("ExerciseID");
                linky.append(exID);
            }
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " + 
e.getMessage());
        }

        return linky.asArray();
    }

    public int noOfExercisesInWorkout(int wID){
        Statement stmt;
        ResultSet rs;
        int number = 0;
        try{
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT COUNT(ExerciseID) FROM
WorkoutComponent WHERE WorkoutID = " + wID);
            number = rs.getInt("COUNT(ExerciseID)");
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
e.getMessage());
        }
        return number;
    }

    public String[] getExercisesFromWorkout(int wID){
        String[] exercises = new
String[noOfExercisesInWorkout(wID)];
        Statement stmt;
        ResultSet rs;
        int count = 0;

        try{
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT Exercises.ExerciseName
FROM Exercises INNER JOIN WorkoutComponent ON
WorkoutComponent.ExerciseID = Exercises.ExerciseID AND
WorkoutComponent.WorkoutID = " + wID + ";");

            while(rs.next()){
                String eName = rs.getString("ExerciseName");
//ename = Exercise Name
                exercises[count] = eName;
                count++;
            }
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
e.getMessage());
        }
        return exercises;
    }

    public void addTime(int uID, int exID, int time){
        Statement stmt;

        try{
            stmt = conn.createStatement();
            stmt.executeUpdate("INSERT INTO Times(UserID, 
ExerciseID, TimeTaken) VALUES("+ uID + ", " + exID + ", " + time 
+ ")");
            stmt.close();
            conn.commit();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " + 
e.getMessage());
        }
    }

    public void addPreference(int uID, int compID, int 
preference){
        Statement stmt;
        try{
            stmt = conn.createStatement();
            stmt.executeUpdate("INSERT INTO Preferences(UserID, 
ComponentID, Preference) VALUES(" + uID + ", " + compID + ", " + 
preference + ")");
            stmt.close();
            conn.commit();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " + 
e.getMessage());
        }
    }

    public int[][] getPreferences(int wID){
        int noOfUsers = getNoOfUsers();
        int exercises = noOfExercisesInWorkout(wID);
        int[][] preferences = new int[noOfUsers][exercises];
        User[] users = getUsers();
        int count = 0;
        Statement stmt;
        ResultSet rs = null;
        try{
            stmt = conn.createStatement();

            for(int x = 0; x < noOfUsers; x++){
                rs = stmt.executeQuery("SELECT ExerciseID FROM 
WorkoutComponent, Preferences WHERE Preferences.UserID = " + 
users[x].getID()
                        + " AND Preferences.ComponentID = 
WorkoutComponent.ComponentID AND WorkoutComponent.WorkoutID = " + 
wID + " ORDER BY Preference ASC");
                while(rs.next()){
                    preferences[x][count] = rs.getInt(1);
                    count++;
                }
                count = 0;
            }
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " + 
e.getMessage());
        }

        return preferences;
    }

    public WorkoutComponent[] getWorkoutComponents(){
        WorkoutComponent[] components = new 
WorkoutComponent[getNoOfComponents()];
        int count = 0;
        Statement stmt;
        ResultSet rs;
        try{
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT ComponentID, WorkoutID, 
ExerciseID FROM WorkoutComponent");
            while(rs.next()){
                components[count] = new
 WorkoutComponent(rs.getInt("ComponentID"),
 rs.getInt("WorkoutID"), rs.getInt("ExerciseID"));
                count++;
            }
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
 e.getMessage());
        }
        return components;
    }

    public int getNoOfComponents(){
        Statement stmt;
        ResultSet rs;
        int number = -1;

        try{
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT COUNT(ComponentID) FROM
 WorkoutComponent");
            number = rs.getInt("COUNT(ComponentID)");
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
 e.getMessage());
        }
        return number;
    }

    public int[][] getTimes(int wID){ //This set of data will
 create the machines preferences. We need to get each machine and
 then
        int noOfUsers = getNoOfUsers();
        int noOfExercises = noOfExercisesInWorkout(wID);
        int[][] times = new int[noOfExercises][noOfUsers];
        int[] exerciseIDs = getExerciseIDsFromWorkout(wID);
        int count = 0;
        Statement stmt;
        ResultSet rs = null;
```

```
        try{
            stmt = conn.createStatement();
            for(int x = 0; x < noOfExercises; x++){
                int currentExID = exerciseIDs[x];
                rs = stmt.executeQuery("SELECT UserID FROM Times
WHERE ExerciseID = " + currentExID + " ORDER BY TimeTaken ASC");
                while(rs.next()){
                    times[x][count] = rs.getInt("UserID");
                    count++;
                }
                count = 0;
            }
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
e.getMessage());
        }
        return times;
    }

    public int[] getRequiredTimes(){
        Statement stmt;
        ResultSet rs;
        LinkedList IDs = new LinkedList();
        try{
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT DISTINCT(ExerciseID)
FROM WorkoutComponent");
            while(rs.next()){
                IDs.append(rs.getInt("DISTINCT(ExerciseID)"));
            }
            stmt.close();
            rs.close();
        }
        catch(SQLException e){
            System.err.println(e.getClass().getName() + ": " +
e.getMessage());
        }
        return IDs.asArray();
    }
    public boolean moreUsers(int wID){
        return getNoOfUsers() > noOfExercisesInWorkout(wID);
    }


}

//wID = WorkoutID
//exID = ExerciseID
//compID = ComponentID
```

GaleShapley

```
package workoutplannernea;

public class GaleShapley {
    final private String[] people;
    final private String[] exercises;
    final private User[] users;
    private int[] userIDs;
    final private int[] exerciseIDs;
    final private int[][] userPreferences;
    final private int[][] exercisePreferences;
    final private String[][] timetables;
    GenericHashMap<String, String[]> GHM;
    boolean type;


    public GaleShapley(int wID){
        DatabaseConnection dbconn = new DatabaseConnection();
        int noOfUsers = dbconn.getNoOfUsers();
        people = dbconn.getUserNames();
        userIDs = new int[noOfUsers];
        exercises = dbconn.getExercisesFromWorkout(wID);
        users = dbconn.getUsers();
        GHM = new GenericHashMap<>(users.length);
        String[] arr;
        if(dbconn.moreUsers(wID)) arr = new String[users.length];
        else arr = new String[exercises.length];
        for(int i = 0; i < users.length; i++){
            userIDs[i] = users[i].getID();
            String username = users[i].getFirst();
            GHM.add(username, arr);
        }
        exerciseIDs = dbconn.getExerciseIDsFromWorkout(wID);
        userPreferences = dbconn.getPreferences(wID);
        exercisePreferences = dbconn.getTimes(wID);
        type = dbconn.moreUsers(wID);
        dbconn.close();
        if(type) timetables = new
String[people.length][people.length];
        else timetables = new
String[people.length][exercises.length];


    }


    public GenericHashMap<String, String[]> fullMatching(){
        if(type){
            for(int i = 0; i < people.length; i++){
                int[] matches = typeTwoMatch();
                addToTimeTables(matches, i);
            }
        }
        else{
            for(int i = 0; i < exercises.length; i++){
                int[] matches = typeOneMatch();
                addToTimeTables(matches, i);
            }
            addToHashMap();
        }
        return GHM;
    }

    public int[] typeOneMatch(){ // We use this matching
algorithm if there are more machines than users
        int userIndex; //This is in reference to the user's
position in the machines preferences
        int exerciseIndex; // And this is for the machines
position in the users preference

        int[] userMatchIndexes = setArrConstant(-1,
people.length); // Index of matched Exercise in own preferences
        int[] exerciseMatchIndexes = setArrConstant(-1,
exercises.length); //Index of matched User in own preferences

        int[] userNextProposal = setArrConstant(0,
people.length);

        boolean[] matchedUsers = setArrFalse(people.length);
        boolean[] matchedExercises =
setArrFalse(exercises.length);

        while(containsFalse(matchedUsers)){
            for(int i = 0; i < people.length; i++){
                if(!matchedUsers[i]){ //user will 'propose' if
not matched
                    exerciseIndex =
indexFromID(userPreferences[i][userNextProposal[i]],
exerciseIDs);
                    if(!matchedExercises[exerciseIndex]){//
Automatic matching if both user and exercise are unmatched
                        matchedExercises[exerciseIndex] = true;
                        matchedUsers[i] = true;
                        userMatchIndexes[i] =
indexFromID(exerciseIDs[exerciseIndex], userPreferences[i]);
                        exerciseMatchIndexes[exerciseIndex] =
indexFromID(userIDs[i], exercisePreferences[exerciseIndex]);
                    }
                    else{
                        userIndex = indexFromID(userIDs[i],
exercisePreferences[exerciseIndex]);
                        if(exerciseMatchIndexes[exerciseIndex] >
userIndex){ //in this case the previously matched user needs to
be unmatched and replaced by the current proposer
                            int rejectedUserIndex =
indexFromID(exercisePreferences[exerciseIndex][exerciseMatchIndex
es[exerciseIndex]], userIDs);
                            userMatchIndexes[rejectedUserIndex] =
-1;
                            matchedUsers[rejectedUserIndex] =
false;
                            exerciseMatchIndexes[exerciseIndex] =
userIndex;
```

```
                            userMatchIndexes[i] =
indexFromID(exerciseIDs[exerciseIndex], userPreferences[i]);
                            matchedExercises[exerciseIndex] =
true;
                            matchedUsers[i] = true;
                        }
                    }
                    userNextProposal[i]++;
                }
            }
        }
        int[] matchedExerciseIDs =
exIDsFromPreferenceIDs(userMatchIndexes);
        alterUserAndExercisePreferences(userMatchIndexes,
exerciseMatchIndexes);
        return matchedExerciseIDs;
    }

    public int[] typeTwoMatch(){// We use this matching method if
there are more users than machines
        int exerciseIndex;
        int userIndex;

        int[] exerciseMatchIndexes = setArrConstant(-1,
exercises.length);
        int[] userMatchIndexes = setArrConstant(-1,
people.length);

        int[] exerciseNextProposal = setArrConstant(0,
exercises.length);

        boolean[] matchedExercises =
setArrFalse(exercises.length);
        boolean[] matchedUsers = setArrFalse(people.length);

        while(containsFalse(matchedExercises)){
            for(int i = 0; i < exercises.length; i++){
                if(!matchedExercises[i]){
                    userIndex =
indexFromID(exercisePreferences[i][exerciseNextProposal[i]],
userIDs);
                    if(!matchedUsers[userIndex]){
                        matchedUsers[userIndex] = true;
                        matchedExercises[i] = true;
                        exerciseMatchIndexes[i] =
                        indexFromID(userIDs[userIndex],
                        exercisePreferences[i]);
                        userMatchIndexes[userIndex] =
indexFromID(exerciseIDs[i], userPreferences[userIndex]);
                    }
                    else{
                        exerciseIndex =
indexFromID(exerciseIDs[i], userPreferences[userIndex]);
                        if(userMatchIndexes[userIndex] >
exerciseIndex){
                            int rejectedExercisesIndex =
indexFromID(userPreferences[userIndex][userMatchIndexes[userIndex
]], exerciseIDs);

exerciseMatchIndexes[rejectedExercisesIndex] = -1;

matchedExercises[rejectedExercisesIndex] = false;
                            userMatchIndexes[userIndex] =
exerciseIndex;
                            exerciseMatchIndexes[i] =
indexFromID(userIDs[userIndex], exercisePreferences[i]);
                            matchedUsers[userIndex] = true;
                            matchedExercises[i] = true;
                        }
                    }
                    exerciseNextProposal[i]++;
                }
            }
        }
        int[] matchedExerciseIDs =
exIDsFromPreferenceIDs(userMatchIndexes);
        alterUserAndExercisePreferences(userMatchIndexes,
exerciseMatchIndexes);
        return matchedExerciseIDs;
    }

    private int[] setArrConstant(int constant, int length){
        int[] arr = new int[length];
        for(int i = 0; i < length; i++){
            arr[i] = constant;
        }
        return arr;
    }

    private boolean[] setArrFalse(int length){
        boolean[] arr = new boolean[length];
        for(int i = 0; i < arr.length; i++){
            arr[i] = false;
        }
        return arr;
    }

    private static boolean containsFalse(boolean[] arr){
        for(boolean b : arr){
            if(!b) return true;
        }
        return false;
    }

    private int indexFromID(int ID, int[] arr){
        for(int i = 0; i < arr.length; i++){
            if(arr[i] == ID) return i;
        }
        return -1;
    }

    private int[] exIDsFromPreferenceIDs(int[] userMatchIndexes){
        int[] exIDs = new int[userMatchIndexes.length];
        for(int i = 0; i < exIDs.length; i++){
            if(userMatchIndexes[i] != -1){
                exIDs[i] =
userPreferences[i][userMatchIndexes[i]];
            }
            else exIDs[i] = -1;
        }
        return exIDs;
    }

    private void addToTimeTables(int[] matchedExerciseIDs, int
count){
        DatabaseConnection dbconn = new DatabaseConnection();
        for(int i = 0; i < people.length; i++){
            if(matchedExerciseIDs[i] == -1){
                timetables[i][count] = "Rest";
            }
            else{
                timetables[i][count] =
dbconn.getExerciseFromID(matchedExerciseIDs[i]);
            }
        }
        dbconn.close();
    }



    private void alterUserAndExercisePreferences(int[]
userMatchIndexes, int[] exerciseMatchIndexes){ //Surprisingly
this is simple. I add each array to a linked list. Delete the
matched value and at it to the end to move each ID
        for(int i = 0; i < people.length; i++){
            if(userMatchIndexes[i] != -1){
                LinkedList linky = new
LinkedList(userPreferences[i]);
                linky.deleteAt(userMatchIndexes[i]);

linky.append(userPreferences[i][userMatchIndexes[i]]);
                userPreferences[i] = linky.asArray();
            }

        }
        for(int j = 0; j < exercises.length; j++){
            if(exerciseMatchIndexes[j] != -1){
                LinkedList linky = new
LinkedList(exercisePreferences[j]);
                linky.deleteAt(exerciseMatchIndexes[j]);

linky.append(exercisePreferences[j][exerciseMatchIndexes[j]]);
                exercisePreferences[j] = linky.asArray();
            }
        }
    }

    public void addToHashMap(){
```

```
    for(int i = 0; i < users.length; i++){
      GHM.alter(users[i].getFirst(), timetables[i]);
    }
  }

  public void displayEntireHashMap(){
    for(int i = 0; i < users.length; i++){
      String[] singleTimeTable = 
GHM.peek(users[i].getFirst());
      System.out.println(users[i].getFirst() + ": ");
      for(String s : singleTimeTable){
        System.out.println(s);
      }
      System.out.println("");
    }

  }
}
```

GenericHashMap

```
package workoutplannernea;

public class GenericHashMap<K, V> {

  private class Entry<K, V>{
    private K key;
    private V value;

    public Entry(K k, V v) {
      this.key = k;
      this.value = v;
    }
  }

  public GenericHashMap(int size){
   this.SIZE = size;
   table = new Entry[SIZE];
  }
  private int elements = 0;
  private final int SIZE;
  private Entry<K, V> table[];
```

```
public void add(K key, V value){
    if(isFull() || contains(key)){
        throw new IllegalArgumentException();
    }

    Entry<K, V> newEntry = new Entry<>(key, value);
    int index = Math.abs(key.hashCode()) % SIZE;
    if(table[index] == null){
        table[index] = newEntry;
        elements++;
        return;
    }
    else{
        int count = 0;
        int checkingIndex = (index + 1) % SIZE;

        while(count < SIZE - 1){
            if(table[checkingIndex] == null){
                table[checkingIndex] = newEntry;
                elements++;
                return;
            }
            checkingIndex = (checkingIndex + 1) % SIZE;
            count++;
        }
    }
}

public V peek(K key){

    if(isEmpty()){
        return null;
    }

    int hash = Math.abs(key.hashCode()) % SIZE;

    int count = 0;
    while(count < SIZE){
        try{
            if(table[hash].key == key){
                return table[hash].value;
            }
            count++;
            hash = (hash + 1) % SIZE;
        }
        catch(NullPointerException e){
            count++;
            hash = (hash + 1) % SIZE;
        }
    }

    return null;
}

public void alter(K key, V newValue){
    if(isEmpty() || !contains(key)){
        throw new IllegalArgumentException();
    }

    int index = indexInTable(key);
    table[index].value = newValue;
}

public boolean contains(K key){
    return peek(key) != null;
}

public boolean isFull(){
    return elements == SIZE;
}

public boolean isEmpty(){
    return elements == 0;
}


public int indexInTable(K key){
    int currentHash = Math.abs(key.hashCode()) % SIZE;
    int count = 0;
    while(count < SIZE){
        try{
                if(table[currentHash].key == key){
                    return currentHash;
                }
                count++;
                currentHash = (currentHash + 1) % SIZE;
            }
            catch(NullPointerException e){
                count++;
                currentHash = (currentHash + 1) % SIZE;
            }
        }
        return -1;
    }

    public int getSize(){
        return SIZE;
    }
}
```

LinkedList

```
package workoutplannernea;

public class LinkedList{
   Element front;
   private int elements = 0;

   public boolean isEmpty(){
       return elements == 0;
   }

   public LinkedList(){

   }

   public LinkedList(int[] arr){
       for(int i : arr){
           append(i);
       }
   }

   public void append(int value){
       Element element = new Element();
       element.value = value;
       element.next = null;

       if(front==null){
           front = element;
       }
       else{
           Element e = front;
           while(e.next != null){
               e = e.next;
           }
           e.next = element;
           element.previous = e;
       }
       elements++;
    }

   public int length(){
       return elements;
   }

   public int index(int value){
       int index = -1;
       int currentIndex = 0;
       Element e = front;

       while (e != null)
       {
           if (e.value == value)
           {
               index = currentIndex;
               break;
           }
           e = e.next;
           currentIndex++;
       }
       return index;
   }

   public void deleteAt(int index){
       if(index >= elements || isEmpty()){
           throw new IllegalArgumentException("Error in
DeleteAt");
       }
       else{
           Element element = front;
           for(int i = 0; i < index; i++){
               element = element.next;
           }
           if(index == 0){
               front = element.next;
               element.next.previous = null;
           }
           else{
               element.previous.next = element.next;
               element.next.previous = element.previous;
           }
           elements--;
       }
   }

   public boolean search(int value){
       return index(value) != -1;
   }

   public int[] asArray(){
       int[] arr = new int[elements];
       Element e = front;
       for(int i = 0; i < elements; i++){
           arr[i] = e.value;
           e = e.next;
       }
       return arr;
   }
}
```

Element

```
package workoutplannernea;
```

```
public class Element {

 private int value;
 private Element previous;
 private Element next;

 public int getValue(){
        return value;
 }
 public Element Previous(){
       return previous;
 }
 public Element Next(){
       return next;
 }
}
```

<u>User</u>

```
package workoutplannernea;

public class User{
    private int UserID;
    private String FirstName, Surname;

    public User(int userid, String firstname, String surname){
        this.UserID = userid;
        this.FirstName = firstname;
        this.Surname = surname;
    }

    public int getID(){
        return UserID;
    }

    public String getFirst(){
        return FirstName;
    }

    public String getSurname(){
        return Surname;
    }

}
```

WorkoutComponent

```
package workoutplannernea;

public class WorkoutComponent {
    private int ComponentID, WorkoutID, ExerciseID;

    public WorkoutComponent(int compID, int wID, int exID){
        ComponentID = compID;
        WorkoutID = wID;
        ExerciseID = exID;
    }

    public int getComponentID(){ return ComponentID;}

    public int getWorkoutID(){ return WorkoutID;}

    public int getExerciseID(){ return ExerciseID;}

}
```

4 - Testing

Test Cases

P = Pass
F = Fail

User Testing

Title	Purpose	Input Data	Expected Result	Result (P/F)
Validation when trying to modify/delete users when no	Make sure that users can't attempt to modify or delete	N/A	"No Users exist"	P

users exist	users when none exist. Furthermore, the code should tell the user that no users exist when they try to display users			
Add User	Tests functionality of adding a user	Ilya Osipov	When display users is selected Ilya Osipov should also be displayed along with the other Users	P
Modify User	Tests functionality of modifying a user	User: Ilya Osipov NewFirst: Nikola NewSurname: Kovac	When Display Users is called Ilya Osipov should be replaced by Nikola Kovac	P
Wrong UserID entered for modification or deletion	If the user enters a non-existent UserID then the code should tell the user that no such user exists.	UserID: 20	Should be prompted to keep entering UserIDs until a valid one is entered	P
Delete Users	Tests functionality of deleting a user	Delete User (Nikola Kovac)	When display users is called Nikola Kovac should not come up	P
Display Users	Tests functionality of the display function	N/A (Only selecting display Users from the main menu)	Roshan Ganithi Bryan Abikaram Mazin Shaikh Hassan Mehdi	P

Workout Testing

Title	Purpose	Input Data	Expected Result	Result (P/F)
Validation when trying to modify/delete	Makes sure that user can't attempt to	N/A	"No workouts exist"	

workouts when no workouts exist	modify or delete workouts when none exist. Furthermore, the code should tell the user that no workouts exist when they try to display workouts			
Create Workout	Tests functionality of adding a workout	Name: Pull, 4 sets lat pulls, 3 seated rows, 3 sets rear delts, 3 sets bicep curls, 3 sets of preacher curls	Pull Workout added to database	
Modify Workout (Add exercise)	Tests functionality for adding exercise to existing workout	3 sets of push ups	Exercise Added to workout component table in database	
Modify Workout (Add existing workout)	Tests code to see if user is allowed to enter the same exercise twice to the same workout	Rear delts	Users are prompted to enter a new exercise until they do.	
Modify Workout (Delete exercise)	Tests functionality of removing exercise from workout	Delete push ups	Exercise removed from workout component table of database	
Modify Workout (Change sets)	Tests functionality of changing sets of an exercise in a workout	Change 4 to 3 sets of bicep curls	WorkoutComponent record modified in database	
Display Workouts	Tests functionality of display workout function	N/A	Pull	
Delete Workout	Tests functionality of deleting a workout	Pull	Pull has been deleted from database	

Full Timetable Testing

Table 1 - Preferences
Barb rows = BP, RD = TriC, PU = PF, LP = SP, BC = LR

	Lat Pulldowns	Seated Rows	Rear Delts	Bicep Curls	Preacher Curls
Alhassan	4	2	3	1	5
Bogdan	4	3	2	5	1
Cai	3	1	5	4	2
Daniel	5	2	4	3	1
Edgar	2	3	4	1	5

Table 2 - Times

	Alhassan	Bogdan	Cai	Daniel	Edgar
Lat Pulldowns	5	2	4	1	3
Seated Rows	2	1	4	3	5
Rear Delts	1	5	2	4	3
Bicep Curls	2	4	3	1	5
Preacher Curls	3	1	4	5	2

Table 3 - Preferences

	Bench Press	Incline Press	Shoulder Press	Pec Fly	Tricep Dips	Tricep Extension
Roshan	1	2	3	4	5	6
Bryan	1	3	2	6	5	4
Mazin	1	2	4	3	6	5
Hassan	6	5	1	2	4	3
Bogdan	2	6	1	3	4	5

Table 4 - Times

	Roshan	Bryan	Mazin	Hassan	Bogdan
Bench Press	8	3	10	6	7
Incline Press	5	5	4	7	5
Shoulder Press	10	3	8	5	7
Pec Fly	6	5	2	3	3
Tricep Dips	2	9	9	8	4
Tricep Extension	9	6	4	3	1

Table 5 - Preferences

	Chest Flies	Barbell Bench Press	Decline Press
Roshan	1	8	8
Bryan	6	3	8
Mazin	7	6	2
Hassan	10	1	5

Table 6 - Times

	Roshan	Bryan	Mazin	Hassan
Chest Flies	1	10	7	3
Barbell Bench Press	1	10	10	1
Decline Press	7	2	9	6

Title	Purpose	Input	Expected Result	Result (P/F)
Matching Test 1	Tests the functionality of the matching algorithm	People: [Alhassan, Bogdan, Cai, Daniel, Edgar] Exercises: [Barbell Rows, Rear Delts, Pull Ups, Lat Pulldowns, Bicep Curls] Preferences and Times: Table 1 & Table 2 Displaying all Timetables	Alhassan: BicepCurl, Rear Delt, Seated Row, Lat Pulldowns, Preacher Curl Bogdan: Preacher Curl, Seated Row, Lat Pulldowns, Bicep Curl, Rear Delt Exercise Cai: Rear Delt Exercise, Lat Pulldowns, Preacher Curl, Seated Row, Bicep Curl Daniel: Seated Row, Bicep Curl, Rear Delt Exercise, Preacher Curl, Lat Pulldowns Edgar: Lat Pulldowns, Preacher Curl, Bicep Curl, Rear Delt Exercise, Seated Row	P
Matching Test 2		People: [Roshan, Bryan, Mazin, Hassan, Bogdan] Exercises: [Bench press, Incline press, Shoulder press, Pec fly, Tricep Dip, Tricep Extensions] Preferences and	Roshan: Tricep Rope Extensions, Incline Press, Barbell Bench Press, Shoulder Press, Incline Press, Chest Flies Bryan: Barbell Bench Press, Shoulder Press, Incline Press,	P

		Times: Table 3 and Table 4. Fetch Roshan's timetable	Tricep Dips, Chest Flies, Tricep Rope Extensions Mazin: Incline Press, Chest Flies, Tricep Dips, Barbell Bench Press, Shoulder Press, Tricep Dips Hassan: Shoulder Press, Tricep Dips, Chest Flies, Incline Press, Tricep Rope Extensions, Barbell Bench Press Bogdan: Chest Flies, Barbell Bench Press, Shoulder Press, Tricep Rope Extensions, Tricep Dips, Incline Press	
Matching Test 3	Tests how matching works when all preferences and times are set to 1	People and Exercises are kept the same as above. Preferences and Times are all set to 1. Retrieve all timetables		P
Matching Test 4	Tests matching when there are more people than exercises	Table 5 & Table 6	Roshan: Chest Flies, Barbell Bench, Press, Decline Press, Rest Bryan: Decline Press, Rest, Barbell Bench Press, Chest Flies Mazin: Rest, Chest Flies, Rest, Decline Press Hassan: Barbell Bench Press,	P

			Decline Press, Chest Flies, Barbell Bench Press	

Validation Testing

Title	Purpose	Input	Expected Result	Result(P/F)
Number of users to add	To validate user Input (Purpose of all following tests)	String, Number that isn't an existing ID or a negative number (Input for all following tests)	Prompted to enter a valid number/ID (Expected Result of all following tests)	P
Preference Entered				P
Time entered				P
Main menu first choice				P
Delete User. ID validation				P
Modify User. ID Validation				P
Main Menu second choice				P
Number of exercises in workout				P
Exercise ID to Add to Workout				P
Sets of Exercise				P
Preferences for exercise				P
Time taken for exercise				P
WorkoutID for deletion				P
WorkoutID for				P

modification				
Main Menu third choice				P
Exercise to delete from Workout				P
ExerciseID of exercise that is having sets changed				P
The new number of sets				P
WorkoutID of workout that user wants to view exercises of				P
WorkoutID that user is trying to get schedules for				P
UserID of desired timetable				P

Youtube Link

https://youtu.be/3B_VrQ9K90g

5 - Evaluation

Objectives Summary

1 - User Need: The user should be able to manage all users of the system

1.1 - The User should be able to add users to the database

This objective has been met. The user is capable of adding users to the database

1.2 - The User should be able to delete users from the database

This objective has been met. The user is capable of deleting users from the database

1.3 - The User should be able to modify users in the database

This objective has been met. The user is capable of modifying users in the database

1.4 - The User should be able to view all users in the database

This objective has been met. The user is capable of viewing all users stored in the database

1.5 - Whenever a User is created they should be made to enter their preference and time taken for each exercise that is currently in a workout

This objective has been met. The user is forced to enter their preferences and times whenever a user is created

2 - User Need: The user should be able to manage the workouts in the system

2.1 - The User should be able to add workouts to the database

This objective has been met. The user is capable of creating/adding workouts to the database

2.2 - The User should be able to delete workouts from the database

This objective has been met. The user is capable of deleting workouts from the database

2.3 - The user should be able modify the exercises in the workout in the database

2.3.1 - The user should be able to add an exercise to an existing workout

This objective has been met. The user is capable of adding an exercise to an existing workout

2.3.2 - The user should be able to delete an exercise from an existing workout

This objective has been met. The user is capable of deleting an exercise from an existing workout

2.3.3 - The user should be able to modify the number of sets of an exercise in an existing workout

This objective has been met. The user is capable of modifying the number of sets of an exercise in an existing workout

2.4 - The User should be able to view all workouts in the database

This objective has been met. The user is capable of viewing all workouts in the database

2.5 - The user should be able to view all exercises and how many sets there are in an existing workout.

This objective has been met. The user is capable of viewing all exercises and the respective number of sets in any existing workout

3 - User Need: The user should be able to get timetables for their workouts

3.1 - The Gale-Shapley algorithm should be used to create the routines.

This objective has been met. The user is capable of creating routines using the Gale-Shapley

3.2 - The User should be able to get routines for every person in the group

This objective has been met. The user is capable of retrieving routines for every member of the group.

3.3 - The User should be able to enter a specific user's ID and get their routines for a workout

This objective has been met. The user is capable of entering an ID to get a specific user's routine for a workout

User Feedback Dialogue with Aden Grandcourt

Roshan G. Ganithi - **Q**, Aden Grandcourt - **A**

Q - Alright Aden so have you attempted to use this system for you and your friends at the gym

A - I have, actually, and I have some feedback that I'd like to share.

Q - However first could I ask if the system has addressed your problems and made your gym experience better.

A - It has been such a time saver. We use the system every time we want to try a different workout and also when we want to make small changes to our existing workouts.

Q - That is good to hear. Are there any ways that the system could change or features that could be added to make it even better?

A - So I feel it would be more usable if there was a graphical interface that was used instead of the command line. For example if we could enter values into a table for our preferences and times.

Q - That makes sense. I believe that I could implement that in a future version. Is there anything else?

A - As we spend more time at the gym we like to change our rest times between sets to change the effectiveness of our workouts. It would be useful to be able to change our preferences and times for workouts and exercises. Adding on, it can get quite tedious to enter each user's preferences and times. Also, when we make some timetables there are some members of the group that get to do the same workout twice and some people that don't do it at all. I understand that it might not be the optimal match but maybe if we could have an option to choose between prioritising people having a chance to try every exercise rather than prioritising the optional match that would be good.

Q - That makes a lot of sense. I can understand the reasoning for changing the times and preferences and also the matching options. I will be working on those in the future. Alright, thank you very much Aden.

A - No problem.

Feedback Analysis

It is clear to see, from Aden's responses, that the system is successful and completed all the objectives it was set to fulfil. It solved all the problems that Aden discussed in our previous interview and from his most recent interview, his group is finding the system helpful. The system allows the user to create workouts and plan them with greater efficiency.

Aden's comments on the use of command line inputs is more than fair because it is the norm for a system to make use of a graphical user interface due to the greater freedom the user is given when adding their data.

Furthermore, Aden mentioned the possibility of the user deciding between prioritising the optimal matches and the user having the opportunity to complete all exercises. This is fair because it is important for people who exercise to train all parts of their body instead of prioritising certain parts.

Finally, the remarks on the subject of data entry were more than understandable. I feel that either a graphical user interface or the option to enter times and preferences from an excel sheet would solve this problem.

Possible Extensions

When I revisit this project my main priority will be to add a graphical user interface to make data entry easier and to make the system more user friendly/aesthetically pleasing. For example the user will be able to click on their choices such as 'Add User' instead of entering a numerical value into the command line.

Furthermore, I will address the issue where the users can't modify their preferences and times once they are entered for the first time. This will be added as one of the main menu options and will give the user more freedom.
Lastly, I would like to give the user the option to prioritise the completion of all exercises in the workout. I intend to implement this as a simple option that the user can alternate between before they retrieve their routines.

Project Conclusion

It is fair to say that the intended user has found the system beneficial to their workout experience in both saving time and making their workouts more effective.

On the other hand, there are changes that could be made to improve the user experience. While these changes might be large they will not change the main and intended purpose of the system.