Deliverable #2:

1. **Name:** Quinlin McNatt
2. **Project Tracker:** Fitness Tracker
3. **Project Summary:** The goal of my semester project is to create a fitness tracker app that I can use to track statistics on workouts, health, and nutrition.

---

4. **Project Requirements:** Note: (I tweaked my requirements after doing more brainstorming, I don't really need admins since the application is for me…)

    1. User can sign up
    2. User can sign in
    3. User input stats
    4. Users can view history
    5. Users can set goals
    6. Users can add nutrition, workouts
    7. Application provides visuals to help user
    8. Users can search food, workouts
    9. User can log workouts/nutrition
    10. Users can browse foods, and workouts

Functionality ID #	**Requirement ID #**	**User Requirement:**
1	1A	User can create account
2	2A	User can login to account
3	3A	User can input information
4	4A	User can view logging history
5	5A	User can set goals
5	5B	User can change goals
6	6A	User can add nutrition and workouts
6	6B	User can change nutrition and workouts
7	7A	User sees visual analytics on log
8	8A	User can search created workouts
8	8A	User can search created nutrition
9	9A	User can log nutrition
9	9B	User can log workout
10	10A	User can browse workouts
10	10B	User can browse nutririon

5. **UI Mockups:** My stretch goal is to create a GUI to make the app better, but for now I'm going to go with command line since the main challenge of the project will be analytics already. If I can complete that I will build a GUI.

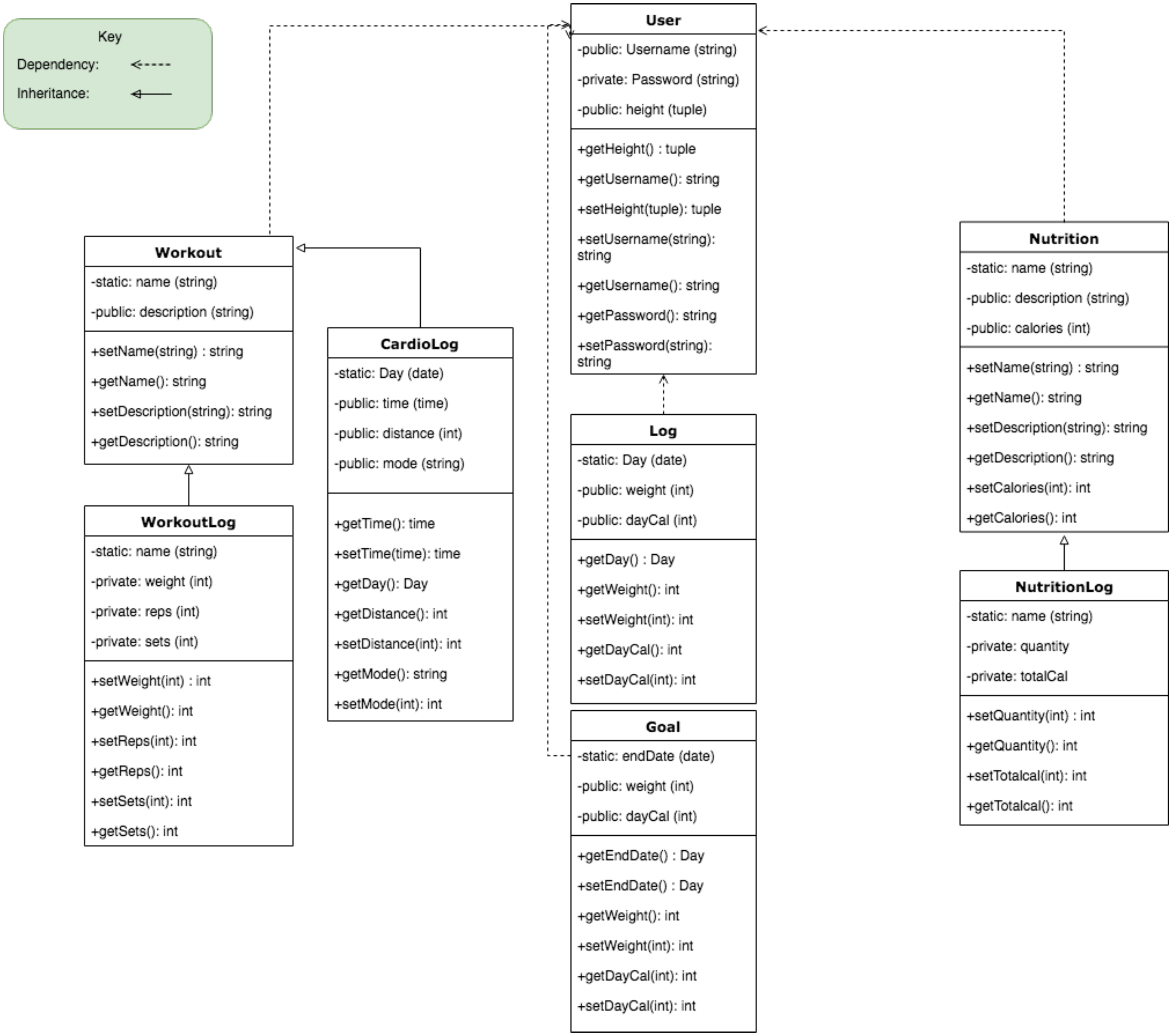
The command line will look like this:

```
Welcome to Fitness Tracker Q's edition! First type in your username,
if you haven't signed up create a profile by typing "new":
{if "new"} enter your new username:
"input"
enter your new password:
"input"
verify your new password:
"input"
Account created! -> skip to Welcome line
Else{}
"input"
Password:
"input"
Welcome "username" you can now log data!
First enter your weight for today, type skip to skip:
"input"
Height:
"input"
Quick Cal count if you do not want to add foods:
"input"
Now what would you like to see or enter?
   1. Log food and workout data
   2. Set goals
   3. Add/Edit/Create nutrition food
   4. Add/Edit/Create workout
   5. View data
   6. Quit

   6. You have selected log food and workout data.  Which would you
      like to log?
      "input"
      You have selected {thing} {prints out list} select item to log:
      {if food} Quantity:
      "input"
      {if workout} weight:
      reps:
```

```
6. You have selected set goals, what goal would you like to set?
   “lists current data on workouts and weight that have settable
   goals”
   “input”
   {Prints out current goal, tells you if you have reached it} If
   you would like to set a new goal, type a number, if not type
   quit!
   “input”
   Goal Set {if new goal was set}
   {Goes back to main menu}

6. You have selected Add/Edit food, would you like to add, edit, or
   create?
   “input”
   {if edit lists all current foods}
   Search a food item you would like to edit:
   Edit food:
   {else if types add}
   Add food, list the name:
   calories:
   {shows total for the day}
   {else if types create}
   Which food would you like to add to the database?
   “input”
   calories:


6. {same as 3, with different inputs}

6. View data: What data would you like to see? {lists data that is
   viewable}

   “input”
   {shows analytics on data including goals and all logs made by the
   user.}

6. Quit {saves data and exits program}
```

Overall, this is a rough skeleton of my UI. I’m sure as I delve into coding there will be some small tweaks, and maybe adding things here and there. Data will either be displayed from a java library, or I’m thinking about exporting data into a csv then loading it into an excel file that will produce analytics on the numbers! Haven’t decided yet. The user will navigate between screens as long as I have a few defined program rules, like typing quit will always quit out.

6. **Class Diagram:** Here's a class diagram for my current application, I'm sure that some methods will need to be added, however most of the important ones are outlined here:


Key
Dependency:
Inheritance:
User
-public: Username (string)
-private: Password (string)
-public: height (tuple)
+getHeight() : tuple
+getUsername(): string
+setHeight(tuple): tuple
+setUsername(string): string
+getUsername(): string
+getPassword(): string
+setPassword(string): string
Workout
-static: name (string)
-public: description (string)
+setName(string) : string
+getName(): string
+setDescription(string): string
+getDescription(): string
CardioLog
-static: Day (date)
-public: time (time)
-public: distance (int)
-public: mode (string)
+getTime(): time
+setTime(time): time
+getDay(): Day
+getDistance(): int
+setDistance(int): int
+getMode(): string
+setMode(int): int
WorkoutLog
-static: name (string)
-private: weight (int)
-private: reps (int)
-private: sets (int)
+setWeight(int) : int
+getWeight(): int
+setReps(int): int
+getReps(): int
+setSets(int): int
+getSets(): int
Log
-static: Day (date)
-public: weight (int)
-public: dayCal (int)
+getDay() : Day
+getWeight(): int
+setWeight(int): int
+getDayCal(): int
+setDayCal(int): int
Goal
-static: endDate (date)
-public: weight (int)
-public: dayCal (int)
+getEndDate() : Day
+setEndDate() : Day
+getWeight(): int
+setWeight(int): int
+getDayCal(int): int
+setDayCal(int): int
Nutrition
-static: name (string)
-public: description (string)
-public: calories (int)
+setName(string) : string
+getName(): string
+setDescription(string): string
+getDescription(): string
+setCalories(int): int
+getCalories(): int
NutritionLog
-static: name (string)
-private: quantity
-private: totalCal
+setQuantity(int) : int
+getQuantity(): int
+setTotalcal(int): int
+getTotalcal(): int