**Food**

**Food(Name, Pic, FoodBenefits, ServingSize, CaloriesPerServing, FoodCategory, Nutrients[]): Food**

Create Food Object

Assign Name to name

Assign Pic to pic

Assign FoodBenefits to foodBenefits

Assign ServingSize to servingSize

Assign CaloriesPerServing to caloriesPerServing

Assign FoodCategory to foodCategory

Assign Nutrients[] to nutrients[]

Return Food

**setFoodCategory(String)**

Assign String value to foodCategory

**getFoodCategory(): float**

Return foodCategory

**setCaloriesPerServing(float)**

Assign float value to caloriesPerServing

**getCaloriesPerServing(): float**

Return caloriesPerServing

**getServingSize(): float**

Return servingSize

**getNutrients(): float[]**

Return float[]

**setNutrients(float[])**

Assign float[] array to nutrients[]

**setServingSize(float)**

Assign float value to servingSize

**setFoodBenefits(String)**

Assign String value to foodBenefits

**getFoodBenefits(): String**

Return foodBenefits

**getFoodPic(): image**

Return pic

**setFoodPic(image)**

Assign image value to Pic

**setFavoriteFood(!isFavorite)**

Assign isFavorite to its opposite boolean value

**Delete()**

Delete Food Object

**Exercise**

**setName(String)**

Assign String value to exerciseName

**getName() : String**

Return exerciseName

**Delete()**

Delete Exercise object

**Pedometer**

**automaticSteps()**

If(isActive = true)

Get steps recorded by the phone’s default step counter

Assigns the value of steps counted to currentSteps

**manuallySetSteps(int)**

Assigns int input value to currentSteps

**getSteps() : int**

Returns currentSteps

**calCalories(currentSteps, pace): float**

Calculate calories burned

Returns the calories burned from walking

**pausePedometer(): false**

Set isActive to false

**resumePedometer(): true**

Set isActive to true

**reset() : currentSteps = 0**

Assign currentSteps to a value of 0

**uploadWalkingLog(CurrentSteps, Pace) : ExerciseLog**

Create ExerciseLog object with the following parameters: today’s date, Walking

Assign today’s date to date

Assign “Walking” to exceriseName

Assign the new Exercise Log’s pace attribute from Pace

Call calCalories() method (parameters: CurrentSteps, Pace) and assign it’s return value to the new ExceriseLog’s caloriesBurned attribute

**Stopwatch**

**Start(isRunning)**

If (isRunning = false)

isRunning = true

Start increasing time

Else (isRunning = true)

isRunning = false

Stop time from increasing

**Reset()**

Set time = 00:00

**UploadTime(time)**

Current time is saved

Send the time to its parent ExerciseLog

**Clock**

**getSleepTime(): time**

Return sleepTime

**setSleepTime(time)**

Assign time value to sleepTime

**setSleepGoal(time)**

//Call the User’s GetAge() in each if statement

If(GetAge() >= 3 AND GetAge() <= 5)

If(time >= 10 hours AND time time <= 13 hours)

Assign time value to sleepGoal

Else

Display a Pop Up Window saying “This isn’t a healthy amount of sleep”

Assign time value to sleepGoal

Else If(GetAge() >= 6 AND GetAge() <= 12)

If(time >= 9 hours AND time time <= 12 hours)

Assign time value to sleepGoal

Else

Display a Pop Up Window saying “This isn’t a healthy amount of sleep”

Assign time value to sleepGoal

Else If(GetAge() >= 13 AND GetAge() <= 17)

If(time >= 8 hours AND time time <= 10 hours)

Assign time value to sleepGoal

Else

Display a Pop Up Window saying “This isn’t a healthy amount of sleep”

Assign time value to sleepGoal

Else If(GetAge() >= 18)

If(time >= 7 hours AND time time <= 9 hours)

Assign time value to sleepGoal

Else

Display a Pop Up Window saying “This isn’t a healthy amount of sleep”

Assign time value to sleepGoal

**getSleepGoal(): time**

Return sleepGoal

**evaluateSleepTime(sleepTime, sleepGoal): tips, feedback**

If(sleepTime < sleepGoal)

Calculate the difference between the sleepGoal and sleepTime

Assign feeback the String “Need more Sleep”

Assign tips the String “Refrain from eating big meals before sleeping, Refrain from sugary foods a couple hours before going to bed, If you’re going to use your phone: turn on Night Shift Mode, Ideally refrain from using your phone an hour before going to bed”

Return tips and feedback

Else if(sleepTime = sleepGoal)

Assign feeback the String “You Slept Enough”

Assign tips the String “Getting enough sleep will help you perform better throughout the day. Good sleep helps you focus more, Strengthens your immune system, Gives you more energy for daily activities, and improves your mental state”

Return tips and feedback

Else if(sleepTime > sleepGoal)

Calculate the difference between the sleepGoal and sleepTime

Assign feeback the String “You Overslept”

Assign tips the String “Getting enough sleep will help you perform better throughout the day. Good sleep helps you focus more, Strengthens your immune system, Gives you more energy for daily activities, and improves your mental state”

Return tips and feedback

**uploadSleepTime(SleepTime, SleepGoal): SleepLog**

Create SleepLog object

Assign SleepTime to sleepTime

Assign SleepGoal to sleepGoal

Assign today’s date to date

Return SleepLog

**FoodLog**

**FoodLog(FoodDish, Date, ServingsConsumed, CalorieLimit, TimeToEat): FoodLog**

Create FoodLog object

Assign FoodDish to Food

Assign Date to date

Assign ServingsConsumed to servingsConsumed

Assign CalorieLimit to calorieLimit

Assign TimeToEat to timeToEat

Return FoodLog

**uploadFoodLog(HealthLog)**

Call HeathLog’s addFoodLog method // HeathLog’s addFoodLog(this.FoodLog)

**getServingsConsumed(): float**

Return servingsConsumed

**setServingsConsumed(float)**

Assign float value to servingsConsumed

**getCaloriesConsumed(): float**

Return caloriesConsumed

**setCaloriesConsumed(float)**

Assign float value to caloriesConsumed

**getCalorieLimit(): float**

Return calorieLimit

**setCalorieLimit(user.GetCalorieLimit())**

Assign User’s calorieLimit value to calorieLimit

**getTimeToEat(): time**

Return timeToEat

**setTimeToEat(time)**

Assign time value to timeToEat

**getMealType(): String**

Return mealType

**setMealType(String)**

Assign String value to MealType

**getDate() : date**

Return date

**delete()**

Delete FoodLog Object

**ExerciseLog**

**ExerciseLog(Date, ExerciseType)**

Create ExerciseLog object

Assign Date to date

Assign ExerciseType to Exercise

Return ExerciseLog

**CalcCalories() : float**

If ((Call getExerciseType() method) = “Running”)

Return calculations from running formula

If ((Call getExerciseType() method) = “Swimming”)

Return calculations from Swimming formula

If ((Call getExerciseType() method) = “Walking”)

Return calculations from Walking formula

If ((Call getExerciseType() method) = “Push Ups”)

Return calculations from Push Ups formula

If ((Call getExerciseType() method) = “Deadlifts”)

Return calculations from Deadlifts formula

If ((Call getExerciseType() method) = “Weights”)

Return calculations from Weights formula

**getExerciseType() : Exercise**

Return exerciseType

**SetExerciseType(Exercise)**

Assign the Exercise Object to exerciseType

**setDate(date)**

Assign date value to date

**getDate() : date**

Return date

**setTime(time)**

Assign time value to time

**getTime() : time**

Return time

**setDistance(float)**

Assign float value to distance

**getDistance() : float**

Return Distance

**setBPM(float)**

Assign float value to BPM

**getBPM() : float**

Return BPM

**setPace(float)**

Assign float value to pace

**getPace() : float**

Return pace

**setWeights(float)**

Assign float value to weights

**getWeights() : float**

Return weights

**setReps(int)**

Assign int value to reps

**getReps() : int**

Return reps

**getSets() : int**

Return sets

**setSets(int)**

Assign int value to sets

**getDate() : date**

Return date

**delete()**

Delete ExerciseLog Object

**SleepLog**

**SleepLog(SleepTime, SleepGoal, Date): SleepLog**

Create SleepLog object

Assign SleepTime to sleepTime

Assign SleepGoal to sleepGoal

Assign Date to date

Return SleepLog

**getSleepTime(): time**

Return sleepTime

**getSleepGoal(): time**

Return sleepGoal

**uploadSleepLog(HealthLog)**

Call HeathLog’s addSleepLog(this.SleepLog) method //Saves SleepLog within HealthLog

**getDate() : date**

Return date

**delete()**

Delete SleepLog Object

**Graph**

**Graph(SleepPoints, SleepTimePoints, CalorieConsumedPoints, CalorieConsumedTimePoints, CaloriesBunredTime, CaloriesBurnedTimePoints) : Graph**

Create Graph Object

//Store Plot Points into ArrayLists (x,y)/(amount of sleep/calories, time sleep started/time calories consumed or burned)

Assign SleepPoints to sleepPoints

Assign SleepTimePoints to sleepTimePoints

Assign CalorieConsumedPoints to calorieConsumedPoints

Assign CalorieConsumedTimePoints to calorieConsumedTimePoints

Assign CaloriesBunredTime to caloriesBurnedTime

Assign CaloriesBurnedTimePoints to caloriesBurnedTimePoints

//Plot points and Assign each ArrayList to a pair of coordinates

Return Graph

**Delete()**

Delete Graph Object

**HealthLog**

**HealthLog(Date) : HealthLog**

Create HealthLog object

Assign Date to date

Return HealthLog

**addToUserProfile(this.HealthLog) : ArrayList(HealthLogs)**

Add this.HealthLog into HealthLogs ArrayList

**foodSuggestion(FoodLog) : ArrayList(Food)**

Declare an int called totalCal

Declare a float array called totalNutrients[]

Declare an int called caloriesLeft

Declare Food called currentFood

Loop through all elements of dailyFoodLogs

Add up each FoodLog’s caloriesConsumed attribute to totalCal

Add up each Food’s (nutrients[] \* servingSize) (Convert from String to Int) to totalNutrients[]

Assign caloriesLeft the difference between User’s calorieLimit and totalCal

If(calorieLimit <= totalCalorie)

Display Pop-Up saying “You’ve Reached your Limit”

Return an empty ArrayList(Food)

Else

Loop through the entire Array of Foods //Will store the Foods below the remaining calorie limit

If(Food.caloriesPerServing <= calorieLimit )

Add Food to ArrayList(Food)

Loop through ArrayList(Food)

Loop through totalNutrients[] elements

If(totalNutrients[i] + Food.nutrients[i] > recommenedNutrients[i])

Remove element from ArrayList

Return ArrayList(Food)

**getExerciseLogs : ArrayList(ExerciseLog)**

Return ExerciseLogs

**addExerciseLog(ExerciseLog)**

Add ExerciseLog into dailyExerciseLogs (ArrayList)

**getFoodLogs() : ArrayList(FoodLog)**

Return FoodLogs

**addFoodLog(FoodLog)**

Add FoodLog into dailyFoodLogs (ArrayList)

**getSleepLog() : SleepLog**

Return dailySleepLog

**getHealthLog() : HealthLog**

Return HealthLog object

**addSleepLog(SleepLog)**

Assign SleepLog to dailySleepLog

**createGraph(SleepPoints, SleepTimePoints, CalorieConsumedPoints, CalorieConsumedTimePoints, CaloriesBunredTime, CaloriesBurnedTimePoints) : Graph**

Create Graph Object with the parameter this.HealthLog

//Store Plot Points into ArrayLists (x,y)/(amount of sleep/calories, time sleep started/time calories consumed or burned)

Assign SleepPoints to sleepPoints

Assign SleepTimePoints to sleepTimePoints

Assign CalorieConsumedPoints to calorieConsumedPoints

Assign CalorieConsumedTimePoints to calorieConsumedTimePoints

Assign CaloriesBunredTime to caloriesBurnedTime

Assign CaloriesBurnedTimePoints to caloriesBurnedTimePoints

//Plot points and Assign each ArrayList to a pair of coordinates

Return Graph

**totalCaloriesConsumed() : float**

Declare a float variable called total

Loop through each element in dailyFoodLog

Add the value of each element’s caloriesConsumed attribute to total

grossCaloriesConsumed = total

**totalCaloriesBurned() : float**

Declare a float variable called total

Loop through each element in dailyExceriseLog

Add the value of each element’s caloriesBurned attribute to total

grossCaloriesBurned = total

**calcNetCalories**

**(grossCaloriesConsumed, grossCaloriesBurned) : float**

Return the difference of grossCaloriesConsumed and grossCaloriesBurned

**getDate() : date**

Return date

**delete()**

Delete SleepLog Object

**User**

**User(Name, Age, Email, Weight, Height, ProfilePic, Password, PhoneNumber) : User**

Create User Object

Assign Name to name

Assign Age to age

Assign Email to email

Assign Weight to weight

Assign Height to height

Assign ProfilePic to profilePic

Assign Password to password

Assign PhoneNumber to phoneNumber

Return User

**setPassword(String)**

Assign String input value to password

**getName() : String**

Return name

**setName(String)**

Assign String input to name

**getEmail() : String**

Return email

**setEmail(String)**

Assign String input to email

**setAge(int)**

Assign int input to age

**getAge() : int**

Return age

**setSex(String)**

Assign String input to sex

**getSex() : String**

Return sex

**setWeight(String)**

Assign String input to weight

**getWeight() : String**

return weight

**setHeight(String)**

Assign String input to height

**getHeight() : String**

Return height

**setPic(picture)**

Assign picture input to picture

**getPic() : picture**

Return picture

**setPhoneNumber(int)**

Assign int input to phoneNumber

**getPhoneNumber() : int**

Return phoneNumber

**getHealthLog(date) : HealthLog**

Search User’s HealthLog ArrayList attribute and search each HealthLog’s date attribute

If(HealthLog.getDate() = date) with the same date value as the paramater

Return HealthLog object

**getHealthLogs() : HealthLogs[]**

Return the user’s array of HealthLogs

**setCurrentCalorieLimit(float)**

Assign float input to currentCalorieLimit

**getCurrentCalorieLimit() : float**

Return CurrentCalorieLimit

**deleteAccount()**

Delete User Object and All User created data