Olivia Smith 505584075

1. One notable obstacle that I overcame was trying to understand how .h and the various .cpp files interacted with each other and how to elicit the response that I wanted from each file. In addition, how to call other functions within different .cpp files and what the importance of the .h files were.

// Start of Stahl's sample test code

Gym gym("SportsClub", "1800 Sepulveda Boulevard", PREMIER);

gym.setFeatures(true, false, true, false);

GymMember me("Howard", "123456", EXECUTIVE);

assert(me.workoutsThisMonth() == 0);

me.startNewMonth();

me.newWorkout();

assert(me.workoutsThisMonth() == 1);

if (gym.canWorkoutHere(me)) gym.checkin(me, true, false, true, false);

assert(me.workoutsThisMonth() == 2);

// End of Stahl's sample test code

//CODE BOARD TESTING

//GymMember Getters

GymMember m("Pixie", "12345", OTHER);

assert(m.getName() == "Pixie");

assert(m.getAccountNumber() == "12345");

assert(m.getKindAsString() == "Other"||"OTHER");

//Workout Defaults

assert(m.workoutsThisMonth() == 0);

m.startNewMonth();

assert(m.workoutsThisMonth() == 0);

//New workout

assert(m.workoutsThisMonth() == 0);

m.startNewMonth();

m.newWorkout();

m.newWorkout();

m.newWorkout();

assert(m.workoutsThisMonth() == 3);

m.startNewMonth();

m.newWorkout();

assert(m.workoutsThisMonth() == 1);

//Gym getters and defaults

Gym exec("ExecGym", "Los Angeles", EXECUTIVE);

assert(exec.getName() == "ExecGym");

assert(exec.getLocation() == "Los Angeles");

assert(exec.getKindAsString() == "Executive"||"EXECUTIVE");

assert(exec.hasCardioFeatures() == false);

assert(exec.hasWeightsFeatures() == false);

assert(exec.hasPoolFeatures() == false);

assert(exec.hasTrackFeatures() == false);

//Gym cardio features

exec.setFeatures(true, false, false, false);

assert(exec.hasCardioFeatures() == true);

assert(exec.hasWeightsFeatures() == false);

assert(exec.hasPoolFeatures() == false);

assert(exec.hasTrackFeatures() == false);

//Gym weights feature

exec.setFeatures(false, true, false, false);

assert(exec.hasCardioFeatures() == false);

assert(exec.hasWeightsFeatures() == true);

assert(exec.hasPoolFeatures() == false);

assert(exec.hasTrackFeatures() == false);

//gym pool functions

exec.setFeatures(false, false, true, false);

assert(exec.hasCardioFeatures() == false);

assert(exec.hasWeightsFeatures() == false);

assert(exec.hasPoolFeatures() == true);

assert(exec.hasTrackFeatures() == false);

//gym track feature

exec.setFeatures(false, false, false, true);

assert(exec.hasCardioFeatures() == false);

assert(exec.hasWeightsFeatures() == false);

assert(exec.hasPoolFeatures() == false);

assert(exec.hasTrackFeatures() == true);

//executive checkin

GymMember executive("You", "67890", EXECUTIVE);

GymMember premier("Me", "11111", PREMIER);

GymMember regular("Someone Else", "22222", REGULAR);

assert(exec.canWorkoutHere(executive) == true);

assert(exec.canWorkoutHere(premier) == false);

assert(exec.canWorkoutHere(regular) == false);

//premier checkin

Gym prem("PremGym", "Sacramento", PREMIER);

assert(prem.canWorkoutHere(executive) == true);

assert(prem.canWorkoutHere(premier) == true);

assert(prem.canWorkoutHere(regular) == false);

//regular checkin

Gym regu("ReguGym", "San Diego", REGULAR);

assert(regu.canWorkoutHere(executive) == true);

assert(regu.canWorkoutHere(premier) == true);

assert(regu.canWorkoutHere(regular) == true);

//regular checkin wanting nothing

assert(regu.checkin(regular, false, false, false, false) == true);

//premier wants pool

premier.startNewMonth();

regu.setFeatures(true, true, true, true); assert(regu.checkin(premier, false, false, true, false) == true);

assert(premier.workoutsThisMonth() == 1);

//exectuive checkin wants cardio

regu.setFeatures(true, false, false, false);

executive.startNewMonth();

assert(regu.checkin(executive, true, false, false, false) == true);

assert(regu.checkin(executive, true, false, false, false) == true);

assert(executive.workoutsThisMonth() == 2);

//executive wants pool but no pool at gym

regu.setFeatures(false, false, false, false);

executive.startNewMonth();

assert(regu.checkin(executive, false, true, false, true) == false);

assert(regu.checkin(executive, true, false, false, true) == false);

assert(executive.workoutsThisMonth() == 0);

//Own test case data

GymMember member\_other("Paul", "1", OTHER);

GymMember member\_regular("Pual", "12", REGULAR);

GymMember member\_premier("Plua", "123", PREMIER);

GymMember member\_exec("Plau", "69", EXECUTIVE);

Gym gym\_other("BFit", "UCLA", OTHER);

Gym gym\_regular("BFitFit", "UCLA", REGULAR);

Gym gym\_premier("BFitFitFit", "UCLA", PREMIER);

Gym gym\_exec("Fit", "UCLA", EXECUTIVE);

/\* GymMember methods \*/

// workoutsThisMonth()

for (int i = 0; i < 100; i++) {

member\_other.newWorkout();

}

assert(member\_other.workoutsThisMonth() == 100);

// startNewMonth()

member\_other.startNewMonth();

assert(member\_other.workoutsThisMonth() == 0);

// newWorkout()

member\_regular.newWorkout();

assert(member\_regular.workoutsThisMonth() == 1);

// getKind()

assert(member\_other.getKind() == OTHER);

// getKindAsString()

assert(member\_other.getKindAsString() == "OTHER");

assert(member\_regular.getKindAsString() == "REGULAR");

assert(member\_premier.getKindAsString() == "PREMIER");

assert(member\_exec.getKindAsString() == "EXECUTIVE");

// getName()

assert(member\_other.getName() == "Paul");

// getAccountNumber()

assert(member\_exec.getAccountNumber() == "69");

/\* Gym methods \*/

// setFeatures()

// hasCardioFeatures()

// hasWeightsFeatures()

// hasPoolFeatures()

// hasTrackFeatures()

gym\_other.setFeatures(true, false, false, false);

gym\_regular.setFeatures(false, true, false, false);

gym\_premier.setFeatures(false, false, true, false);

gym\_exec.setFeatures(false, false, false, true);

assert(gym\_other.hasCardioFeatures());

assert(!gym\_regular.hasCardioFeatures());

assert(gym\_regular.hasWeightsFeatures());

assert(!gym\_premier.hasWeightsFeatures());

assert(gym\_premier.hasPoolFeatures());

assert(!gym\_exec.hasPoolFeatures());

assert(gym\_exec.hasTrackFeatures());

assert(!gym\_other.hasTrackFeatures());

// canWorkoutHere()

//members working out at other gyms. any member can workout at an OTHER gym

//no OTHER member can workout at Regular, premier, or executive gyms, but can workout at an other gym

assert(gym\_other.canWorkoutHere(member\_other));

assert(gym\_other.canWorkoutHere(member\_regular));

assert(gym\_other.canWorkoutHere(member\_premier));

assert(gym\_other.canWorkoutHere(member\_exec));

assert(!gym\_regular.canWorkoutHere(member\_other));

assert(gym\_regular.canWorkoutHere(member\_regular));

assert(gym\_regular.canWorkoutHere(member\_premier));

assert(gym\_regular.canWorkoutHere(member\_exec));

assert(!gym\_premier.canWorkoutHere(member\_other));

assert(!gym\_premier.canWorkoutHere(member\_regular));

assert(gym\_premier.canWorkoutHere(member\_premier));

assert(gym\_premier.canWorkoutHere(member\_exec));

assert(!gym\_exec.canWorkoutHere(member\_other));

assert(!gym\_exec.canWorkoutHere(member\_regular));

assert(!gym\_exec.canWorkoutHere(member\_premier));

assert(gym\_exec.canWorkoutHere(member\_exec));

// checkin()

member\_other.startNewMonth();

member\_regular.startNewMonth();

member\_premier.startNewMonth();

member\_exec.startNewMonth();

assert(gym\_other.checkin(member\_other, true, false, false, false));

assert(!gym\_other.checkin(member\_other, true, true, true, true));

assert(member\_other.workoutsThisMonth() == 1);

assert(gym\_regular.checkin(member\_regular, false, true, false, false));

assert(!gym\_regular.checkin(member\_regular, true, true, true, true));

assert(member\_regular.workoutsThisMonth() == 1);

assert(gym\_premier.checkin(member\_premier, false, false, true, false));

assert(!gym\_premier.checkin(member\_premier, true, true, true, true));

assert(member\_premier.workoutsThisMonth() == 1);

assert(gym\_exec.checkin(member\_exec, false, false, false, true));

assert(!gym\_exec.checkin(member\_exec, true, true, true, true));

assert(member\_exec.workoutsThisMonth() == 1);

// getKind()

assert(gym\_other.getKind() == OTHER);

// getKindAsString()

assert(gym\_other.getKindAsString() == "OTHER");

assert(gym\_regular.getKindAsString() == "REGULAR");

assert(gym\_premier.getKindAsString() == "PREMIER");

assert(gym\_exec.getKindAsString() == "EXECUTIVE");

// getLocation()

assert(gym\_other.getLocation() == "UCLA");

// getName()

1. assert(gym\_exec.getName() == "Fit");