Appendices

Interaction with Advisor

J: Is my design good?

S: Yes, but I think you should follow the model, view, and controller structure in Java.

J: What is that?

S: You should divide your program into three different sections. The model is the data structure, the view is what the user sees, and the controller controls the follow.

J: Thanks, I'll try to do that.

Next Interaction:

S: Your view and model is not bad, but you don't use your controller class to control the flow.

J: You're right, I should do that in the future, I don't have time now.

References

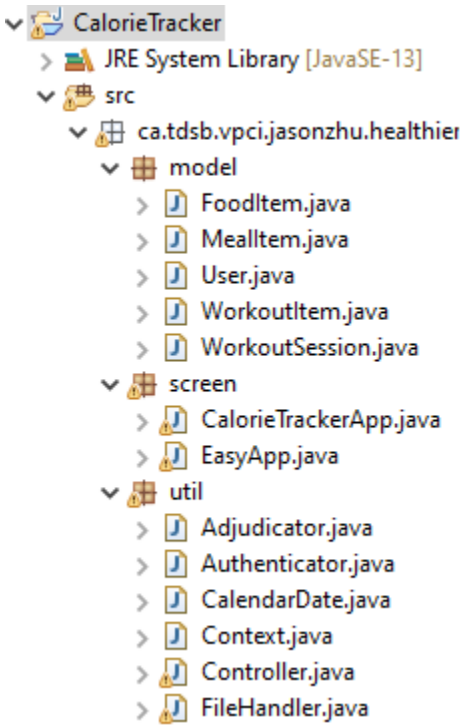
[1] Mulkey, D. 2006, *EasyApp – Simplified AWT Controls*. Visited 22 Dec. 2020. http://ibcomp.fis.edu/Java/EasyApp.html

[2] Wikipedia, Model–view–controller. Visited 22 Dec. 2020. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller#:~:text=Model%E2%80%93view%E2%80%93controller%20(usually,logic%20into%20three%20interconnected%20elements.

[3] Test Plan Template. Visited 24 Dec. 2020. https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2007/07/sample-test-plan-template.pdf

[4] Marcin, A. (2018, September 29). How Many Calories Do You Burn While You're Asleep? Visited January 29, 2021. https://www.healthline.com/health/calories-burned-sleeping

Source Code



Util Package

```
package ca.tdsb.vpci.jasonzhu.healthier.util;

import java.awt.*;

import ca.tdsb.vpci.jasonzhu.healthier.screen.EasyApp;
import ca.tdsb.vpci.jasonzhu.healthier.screen.CalorieTrackerApp;

public class Controller extends EasyApp {

	public static void main(String[] args) {
		new CalorieTrackerApp();
   }

}
```

```java
package ca.tdsb.vpci.jasonzhu.healthier.util;

import java.util.ArrayList;
import ca.tdsb.vpci.jasonzhu.healthier.model.MealItem;
import ca.tdsb.vpci.jasonzhu.healthier.model.User;
import ca.tdsb.vpci.jasonzhu.healthier.model.WorkoutSession;

public class Adjudicator {

	private FileHandler fileHandler = new FileHandler();

	public double targetCalories(User user) {

		double weight = user.getWeight();
		double height = user.getHeight();
		double age = user.getAge();
		double targetCalories = 0;
		String gender = user.getGender();

		if (gender.equalsIgnoreCase("Male")) {
			targetCalories = 66 + (6.2 * weight) + (12.7 * height) - (6.76 *
age);
		} else if (gender.equalsIgnoreCase("Female")) {
			targetCalories = 655.1 + (4.35 * weight) + (4.7 * height) - (4.7
* age);
		}

		return Math.round(targetCalories * 10) / 10.0;

	}

	public double calculateDailyCalories(ArrayList<MealItem> mealItems) {

		double calories = 0;

		for (MealItem m : mealItems) {
			if (m != null) {
				calories += m.getCalories() * m.getPortions();
			}
		}

		return calories;
	}

	public double calculateDailyCaloriesBurned(ArrayList<WorkoutSession>
workoutSessions) {

		double caloriesBurned = 0;

		for (WorkoutSession w : workoutSessions) {
			if (w != null) {
				caloriesBurned += w.getCaloriesBurned() *
w.getSessionTime();
			}
		}
```

```
            return caloriesBurned;

      }

      public double netCalories() {

            return Math.round((calculateDailyCalories(fileHandler.getMealItems()) -
calculateDailyCaloriesBurned(fileHandler.getWorkoutSessions()) -
Context.getUser().getTargetCalories()) * 10.0) / 10.0;

      }

      public String healthMessage(double netCalories) {

            if (netCalories < 100 && netCalories > 0) {
                  return "You have gained a small amount of calories today, but
this barely affects your weight.|To maintain your current weight, continue with this
calorie count.|To lose weight, burn more than 100 calories a day.|To gain weight,
take in more than 100 calories a day.";
            } else if (netCalories >= 100) {
                  return "You gained a noticeable amount of calories today.|If you
continue with this calorie count, you will gain weight.|To maintain your current
weight, try to take in or burn less than 100 calories a day.|To lose weight, burn
more than 100 calories each day.";
            } else if (netCalories > -100 && netCalories < 0) {
                  return "You have lost small amount of calories today, but this
barely affects your weight.|To maintain your current weight, continue with this
calorie count.|To lose weight, burn more than 100 calories a day.|To gain weight,
gain 100 or more calories a day.";
            } else if (netCalories <= -100) {
                  return "You have lost a noticeable amount of calories today.|If
you continue with this calorie count, you will lose weight.|To maintain your current
weight, try to take in or burn less than 100 calories.|To gain weight, gain more than
100 calories each day.";
            } else {
                  return "You have no gain or loss of calories today.|To maintain
weight, continue this calorie count.|To lose weight, burn more than 100 calories each
day.|To gain weight, gain more than 100 calories each day.";
            }

      }

}
```

```java
package ca.tdsb.vpci.jasonzhu.healthier.util;

import java.util.ArrayList;

import ca.tdsb.vpci.jasonzhu.healthier.model.FoodItem;
import ca.tdsb.vpci.jasonzhu.healthier.model.User;
import ca.tdsb.vpci.jasonzhu.healthier.model.WorkoutItem;

public class Authenticator {

	private ArrayList<User> users = new ArrayList<User>();
	private ArrayList<FoodItem> foods = new ArrayList<FoodItem>();
	private ArrayList<WorkoutItem> workouts = new ArrayList<WorkoutItem>();
	private FileHandler fileHandler = new FileHandler();

	public String register(User user) {

		String username = user.getUsername();

		if (!user.getPassword().trim().equals(user.getConfirmPassword().trim()))
{
			return "Unsuccessful register, passwords do not match.";
		}

		if (user.getUsername().equals("") || user.getPassword().equals("") ||
user.getConfirmPassword().equals("") || user.getGender().equals("Choose Gender")) {
			return "Unsuccessful register, empty fields.";
		}

		users = fileHandler.getUsers();

		if (users != null && users.size() > 0) {
			for (User u : users) {
				if (u != null) {
					if (u.getUsername().trim().equals(username.trim()))
{
						return "Unsuccessful register, username is
already in system.";
					}
				}
			}
		}
		fileHandler.register(user);
		return "success";

	}

	public boolean login(String username, String password) {

		users = fileHandler.getUsers();
		for (User u : users) {
			if (u != null) {
				if (u.getUsername().equals(username) &&
u.getPassword().equals(password)) {
					Context.setUser(u);
```

```
                        fileHandler.createFiles();
                        return true;
                    }
                }
            }

            return false;

        }

        public boolean addFoodItem(FoodItem foodItem) {

            String food = foodItem.getName();

            foods = fileHandler.getFoodItems();
            if (foods != null) {
                for (FoodItem f : foods) {
                    if (f != null) {
                        if (f.getName().equals(food)) {
                            return false;
                        }
                    }
                }
            }
            fileHandler.addFood(foodItem);
            return true;

        }

        public boolean addWorkoutItem(WorkoutItem workoutItem) {

            String workout = workoutItem.getName();

            workouts = fileHandler.getWorkoutItems();
            if (workouts != null) {
                for (WorkoutItem w : workouts) {
                    if (w != null) {
                        if (w.getName().equals(workout)) {
                            return false;
                        }
                    }
                }
            }
            fileHandler.addWorkout(workoutItem);
            return true;
        }

}
```

```java
package ca.tdsb.vpci.jasonzhu.healthier.util;

import java.util.Calendar;
import java.util.Date;

public class CalendarDate {

	private String month = "Month";

	Calendar cal = Calendar.getInstance();
	Date date = cal.getTime();

	public int getDays() {

		return cal.getActualMaximum(Calendar.DAY_OF_MONTH);
	}

	public String getMonth() {

		switch (cal.get(Calendar.MONTH)) {
		case 0:
			month = "January";
			break;
		case 1:
			month = "February";
			break;
		case 2:
			month = "March";
			break;
		case 3:
			month = "April";
			break;
		case 4:
			month = "May";
			break;
		case 5:
			month = "June";
			break;
		case 6:
			month = "July";
			break;
		case 7:
			month = "August";
			break;
		case 8:
			month = "September";
			break;
		case 9:
			month = "October";
			break;
		case 10:
			month = "November";
			break;
		case 11:
			month = "December";
```

```
                        break;
                }

                return month;

        }

        public int getYear() {
                return cal.get(Calendar.YEAR);
        }

        public int getTodaysDate() {
                cal.setTime(date);
                return cal.get(Calendar.DAY_OF_MONTH);
        }

        public int getWeekdayOfFirstDay() {
                cal.set(Calendar.DAY_OF_MONTH, 1);
                date = cal.getTime();
                return cal.get(Calendar.DAY_OF_WEEK);
        }

        public void setToPrevMonth() {
                cal.add(Calendar.MONTH, -1);
                date = cal.getTime();
        }

        public void setToNextMonth() {
                cal.add(Calendar.MONTH, 1);
                date = cal.getTime();
        }

}
```

```java
package ca.tdsb.vpci.jasonzhu.healthier.util;

import ca.tdsb.vpci.jasonzhu.healthier.model.User;

public class Context {

	private static User user;
	private static int dayOfMonth;
	private static String month;
	private static int year;
	private static String spaces = "          ";

	public static String getSpaces() {
		return spaces;
	}

	public static void setSpaces(String spaces) {
		Context.spaces = spaces;
	}

	public static String getWeekdays() {
		return weekdays;
	}

	public static void setWeekdays(String weekdays) {
		Context.weekdays = weekdays;
	}

	static String weekdays = "Sunday" + spaces + "Monday" + spaces + "Tuesday" +
spaces + "Wednesday" + spaces + "Thursday" + spaces + "Friday" + spaces + "Saturday";

	public static String getMonth() {
		return month;
	}

	public static void setMonth(String month) {
		Context.month = month;
	}

	public static int getDayOfMonth() {
		return dayOfMonth;
	}

	public static void setDayOfMonth(int dayOfMonth) {
		Context.dayOfMonth = dayOfMonth;
	}

	public static User getUser() {
		return user;
	}

	public static void setUser(User user) {
		Context.user = user;
	}
```

```
	public static void setYear(int year) {
		Context.year = year;
	}

	public static int getYear() {
		return Context.year;
	}
}
```

```java
package ca.tdsb.vpci.jasonzhu.healthier.util;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.ArrayList;
import java.awt.*;
import ca.tdsb.vpci.jasonzhu.healthier.model.FoodItem;
import ca.tdsb.vpci.jasonzhu.healthier.model.MealItem;
import ca.tdsb.vpci.jasonzhu.healthier.model.User;
import ca.tdsb.vpci.jasonzhu.healthier.model.WorkoutItem;
import ca.tdsb.vpci.jasonzhu.healthier.model.WorkoutSession;

public class FileHandler {

	private RandomAccessFile userFile;
	private RandomAccessFile foodItemFile;
	private RandomAccessFile workoutItemFile;
	private RandomAccessFile foodDateFile;
	private RandomAccessFile workoutDateFile;

	public FileHandler() {
		try {
			userFile = new RandomAccessFile("users.txt", "rw");
		}catch(Exception e) {
			e.printStackTrace();
		}
	}

	public void register(User user) {
		try {
			userFile.writeBytes(user.toString());
			userFile.writeBytes(System.lineSeparator());
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

	public void createFiles() {
		try {
			foodItemFile = new RandomAccessFile("food" +
Context.getUser().getUsername() + ".txt", "rw");
			workoutItemFile = new RandomAccessFile("workout" +
Context.getUser().getUsername() + ".txt", "rw");
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

	public void addFood(FoodItem foodItem) {
		try {
			foodItemFile = new RandomAccessFile("food" +
Context.getUser().getUsername() + ".txt", "rw");
			foodItemFile.seek(foodItemFile.length());
			foodItemFile.writeBytes(foodItem.toString());
			foodItemFile.writeBytes(System.lineSeparator());
```

```
			} catch (IOException e) {
				e.printStackTrace();
			}
		}

		public void addWorkout(WorkoutItem workoutItem) {
			try {
				workoutItemFile = new RandomAccessFile("workout" +
Context.getUser().getUsername() + ".txt", "rw");
				workoutItemFile.seek(workoutItemFile.length());
				workoutItemFile.writeBytes(workoutItem.toString());
				workoutItemFile.writeBytes(System.lineSeparator());

			} catch (IOException e) {
				e.printStackTrace();
			}
		}

		public void createDateFile() {
			try {
				foodDateFile = new
RandomAccessFile(Context.getUser().getUsername() + Context.getMonth() +
Context.getDayOfMonth() + Context.getYear() + "food" + ".txt", "rw");
				workoutDateFile = new
RandomAccessFile(Context.getUser().getUsername() + Context.getMonth() +
Context.getDayOfMonth() + Context.getYear() + "workout" + ".txt", "rw");
			} catch (IOException e) {
				e.printStackTrace();
			}
		}

		public void chooseFood(MealItem mealItem) {
			try {
				foodDateFile = new
RandomAccessFile(Context.getUser().getUsername() + Context.getMonth() +
Context.getDayOfMonth() + Context.getYear() + "food" + ".txt", "rw");
				foodDateFile.seek(foodDateFile.length());
				foodDateFile.writeBytes(mealItem.toString());
				foodDateFile.writeBytes(System.lineSeparator());

			} catch (IOException e) {
				e.printStackTrace();
			}
		}

		public void chooseWorkout(WorkoutItem workoutItem) {
			try {
				workoutDateFile = new
RandomAccessFile(Context.getUser().getUsername() + Context.getMonth() +
Context.getDayOfMonth() + Context.getYear() + "workout" + ".txt", "rw");
				workoutDateFile.seek(workoutDateFile.length());
				workoutDateFile.writeBytes(workoutItem.toString());
				workoutDateFile.writeBytes(System.lineSeparator());

			} catch (IOException e) {
```

```
                        e.printStackTrace();
                }
        }

        public void printMealItems(List list) {
                ArrayList<MealItem> mealItems = getMealItems();
                for (MealItem m : mealItems) {
                        if (m != null)
                                list.add(m.getMeal() + ": " + m.getName() + " (" +
m.getCalories() + " Calories / portion)" + " x " + m.getPortions());
                }
        }

        public void printWorkoutSessions(List list) {
                ArrayList<WorkoutSession> workoutSessions = getWorkoutSessions();
                for (WorkoutSession w : workoutSessions) {
                        if (w != null)
                                list.add(w.getName() + " (" + w.getCaloriesBurned() + "
Calories Burned / hour) " + w.getSessionTime() + " hours");
                }
        }

        public ArrayList<User> getUsers() {
                String[] strings = new String[9];
                ArrayList<User> users = new ArrayList<User>();
                String user = null;
                try {
                        userFile.seek(0);
                        while((user = userFile.readLine()) != null) {
                                if (!user.isEmpty()) {
                                        strings = user.split(", ");
                                        User newUser = new User();
                                        newUser.setUsername(strings[0].trim());
                                        newUser.setPassword(strings[1]);
                                        newUser.setConfirmPassword(strings[2]);
                                        newUser.setAge(Double.parseDouble(strings[3]));
                                        newUser.setWeight(Double.parseDouble(strings[4]));
                                        newUser.setHeight(Double.parseDouble(strings[5]));
                                        newUser.setGender(strings[6]);

        newUser.setTargetCalories(Double.parseDouble(strings[7]));
                                        users.add(newUser);
                                }
                        }
                }catch (Exception e) {
                                e.printStackTrace();
                        e.getMessage();
                }
                return users;
        }

        public ArrayList<FoodItem> getFoodItems() {
                String[] strings = new String[4];
                ArrayList<FoodItem> foods = new ArrayList<FoodItem>();
                String food;
```

```
		try {
			RandomAccessFile main = new RandomAccessFile("food" +
Context.getUser().getUsername() + ".txt", "r");
			do {
				food = main.readLine();
				if (food != null) {
					strings = food.split(", ");
					FoodItem foodItem = new FoodItem();
					foodItem.setName(strings[0]);

	foodItem.setCalories(Double.parseDouble(strings[1]));
					foods.add(foodItem);
				}
			} while (food != null);
		} catch (IOException e) {
			e.printStackTrace();
			e.getMessage();
		}
		return foods;
	}

	public ArrayList<WorkoutItem> getWorkoutItems() {
		String[] strings = new String[2];
		ArrayList<WorkoutItem> workouts = new ArrayList<WorkoutItem>();
		String workout;
		try {
			RandomAccessFile main = new RandomAccessFile("workout" +
Context.getUser().getUsername() + ".txt", "r");
			do {
				workout = main.readLine();
				if (workout != null) {
					strings = workout.split(", ");
					WorkoutItem workoutItem = new WorkoutItem();
					workoutItem.setName(strings[0]);

	workoutItem.setCaloriesBurned(Double.parseDouble(strings[1]));
					workouts.add(workoutItem);
				}
			} while (workout != null);
		} catch (IOException e) {
			e.printStackTrace();
			e.getMessage();
		}
		return workouts;
	}

	public ArrayList<MealItem> getMealItems() {
		String[] strings = new String[4];
		ArrayList<MealItem> mealItems = new ArrayList<MealItem>();
		String mealItem;
		try {
			RandomAccessFile main = new
RandomAccessFile(Context.getUser().getUsername() + Context.getMonth() +
Context.getDayOfMonth() + Context.getYear() + "food" + ".txt", "r");
			do {
```

```
                        mealItem = main.readLine();
                        if (mealItem != null) {
                                strings = mealItem.split(", ");
                                MealItem meal = new MealItem();
                                meal.setName(strings[0]);
                                meal.setCalories(Double.parseDouble(strings[1]));
                                meal.setMeal(strings[2]);
                                meal.setPortions(Double.parseDouble(strings[3]));
                                mealItems.add(meal);
                        }
                } while (mealItem != null);
        } catch (IOException e) {
                e.printStackTrace();
                e.getMessage();
        }
        return mealItems;
    }

    public ArrayList<WorkoutSession> getWorkoutSessions() {
        String[] strings = new String[3];
        ArrayList<WorkoutSession> workoutSessions = new
ArrayList<WorkoutSession>();
        String workoutSession;
        try {
                RandomAccessFile main = new
RandomAccessFile(Context.getUser().getUsername() + Context.getMonth() +
Context.getDayOfMonth() + Context.getYear() + "workout" + ".txt", "r");
                do {
                        workoutSession = main.readLine();
                        if (workoutSession != null) {
                                strings = workoutSession.split(", ");
                                WorkoutSession session = new WorkoutSession();
                                session.setName(strings[0]);

    session.setCaloriesBurned(Double.parseDouble(strings[1]));

    session.setSessionTime(Double.parseDouble(strings[2]));
                                workoutSessions.add(session);
                        }
                } while (workoutSession != null);
        } catch (IOException e) {
                e.printStackTrace();
                e.getMessage();
        }
        return workoutSessions;
    }

}
```

Screen Package

```
package ca.tdsb.vpci.jasonzhu.healthier.screen;

import java.util.ArrayList;
import java.awt.*;

import ca.tdsb.vpci.jasonzhu.healthier.util.Adjudicator;
import ca.tdsb.vpci.jasonzhu.healthier.util.Authenticator;
import ca.tdsb.vpci.jasonzhu.healthier.util.Context;
import ca.tdsb.vpci.jasonzhu.healthier.model.FoodItem;
import ca.tdsb.vpci.jasonzhu.healthier.model.MealItem;
import ca.tdsb.vpci.jasonzhu.healthier.model.User;
import ca.tdsb.vpci.jasonzhu.healthier.model.WorkoutItem;
import ca.tdsb.vpci.jasonzhu.healthier.model.WorkoutSession;
import ca.tdsb.vpci.jasonzhu.healthier.util.CalendarDate;
import ca.tdsb.vpci.jasonzhu.healthier.util.FileHandler;

public class CalorieTrackerApp extends EasyApp {

	private CalendarDate calendarDate = new CalendarDate();
	private Authenticator a = new Authenticator();
	private Adjudicator adjudicator = new Adjudicator();
	private FileHandler fileHandler = new FileHandler();

	private boolean fromRegister = false;
	private boolean fromAddFood = false;
	private boolean fromAddWorkout = false;
	private boolean fromChooseFood = false;
	private boolean fromChooseWorkout = false;
	private boolean fromCalendar = false;
	private boolean calendarFromLogin = false;
	private boolean calendarFromDiet = false;

	//HOME
	private Label title_welcome = addLabel("Welcome to your Calorie
Tracker!",50,50,200,50,this);
	private Button newUser = addButton("New User",50,100,100,50,this);
	private Button returningUser = addButton("Returning
User",250,100,100,50,this);
	//REGISTER
	private Label title_register;
	private Button register_confirm;
	private Label username;
	private Label password;
	private Label confirmPassword;
	private Label weight;
	private Label height;
	private Label age;
	private Label gender;
	private TextField usernameBox;
	private TextField passwordBox;
	private TextField confirmPasswordBox;
	private TextField weightBox;
	private TextField heightBox;
```

```
private TextField ageBox;
private Choice genderBox;
//LOGIN
private Button login_confirm;
private Label login_username;
private Label login_password;
private TextField usernameLoginBox;
private TextField passwordLoginBox;
private Label title_login;
//DIET SCREEN
private Label title_diet;
private Label title_diet_workout;
private Label title_diet_health;
private Label diet_date;
private Button calendarButton;
private Button addFood;
private Button chooseFood;
private Button chooseWorkout;
private List dailyFoods;
private List dailyWorkouts;
private List caloriesFromFood;
private List caloriesBurnedFromWorkout;
private List healthReport;
//ADD FOOD
private Label title_addFood;
private Label foodName;
private Label calories;
private Button addFoodConfirm;
private Button foodBack;
private TextField foodNameBox;
private TextField caloriesBox;
//CALENDAR
private ArrayList<Button> days = new ArrayList<Button>();
private Button prevMonth;
private Button nextMonth;
private Label title_calendar;
private int dateFromButton;
private Label weekdays;
//ADD WORKOUT
private Button addWorkoutItem;
private Button addWorkoutConfirm;
private Button workoutBack;
private Label title_addWorkout;
private Label caloriesBurned;
private Label workoutName;
private TextField caloriesBurnedBox;
private TextField workoutNameBox;
//CHOOSE FOOD
private Label title_chooseFood;
private Button chooseFoodConfirm;
private Button chooseFoodBack;
private Choice choiceOfFood;
private Choice meal;
private Label portionAmount;
private TextField portionAmountBox;
```

```
	private ArrayList<FoodItem> foods = new ArrayList<FoodItem>();
	private String foodNames;
	//CHOOSE WORKOUT
	private Label title_chooseWorkout;
	private Button chooseWorkoutConfirm;
	private Button chooseWorkoutBack;
	private Choice choiceOfWorkout;
	private Label workoutDuration;
	private TextField workoutDurationBox;
	private ArrayList<WorkoutItem> workouts = new ArrayList<WorkoutItem>();
	private String workoutNames;

	public CalorieTrackerApp() {
		setTitle("Calorie Tracker");
	    setSize(400,200);
	}

	public void actions(Object source,String command) {
		if (source == newUser) {
			changeToRegisterScreen();
		}
		if (source == returningUser) {
			changeToLoginScreen();
		}
		if (source == register_confirm) {
			boolean isEmpty = false;
			User user = new User();
			user.setUsername(usernameBox.getText());
			user.setPassword(passwordBox.getText());
			user.setConfirmPassword(confirmPasswordBox.getText());
			try {
				user.setWeight(Double.parseDouble(weightBox.getText()));
				user.setHeight(Double.parseDouble(heightBox.getText()));
				user.setAge(Double.parseDouble(ageBox.getText()));
			} catch (NumberFormatException e) {
				output("Unsuccessful register, enter a number for weight,
height, and age.");
				isEmpty = true;
			}
			try {
				user.setGender(genderBox.getSelectedItem());
			} catch (NullPointerException e) {
				output("Unsuccessful register, empty fields");
				isEmpty = true;
			}
			user.setTargetCalories(adjudicator.targetCalories(user));
			if (isEmpty) {}
			else {
				String register = a.register(user);
				if (register.equals("success")) {
					changeToLoginScreen();
				} else {
					output(register);
				}
			}
```

```
		}
		if (source == login_confirm) {
			if (a.login(usernameLoginBox.getText(),
passwordLoginBox.getText())) {
				calendarFromLogin = true;
				changeToCalendarScreen();
			} else {
				output("Unsuccessful log-in, username or password
incorrect.");
			}
		}
		if (source == addFood) {
			changeToAddFoodScreen();
		}
		if (source == addFoodConfirm) {
			boolean errorMessage = false;
			FoodItem foodItem = new FoodItem();
			if (!foodNameBox.getText().equals(""))
				foodItem.setName(foodNameBox.getText());
			else {
				output("Please enter a name for the food.");
				errorMessage = true;
			}
			if (!errorMessage) {
				try {

	foodItem.setCalories(Integer.parseInt(caloriesBox.getText()));
				} catch (NumberFormatException e) {
					output("Please enter a number for calories.");
					errorMessage = true;
				}
			}
			if (!errorMessage) {
				if (!a.addFoodItem(foodItem)) {
					output("Food is already in system. Use a different
name.");
				} else {
					fromAddFood = true;
					changeToDietScreen();
				}
			}
		}
		if (source == addWorkoutItem) {
			changeToAddWorkoutItemScreen();
		}
		if (source == addWorkoutConfirm) {
			boolean errorMessage = false;
			WorkoutItem workoutItem = new WorkoutItem();
			if (!workoutNameBox.getText().equals(""))
				workoutItem.setName(workoutNameBox.getText());
			else {
				output("Please enter a name for the workout.");
				errorMessage = true;
			}
			if (!errorMessage) {
```

```
                        try {

      workoutItem.setCaloriesBurned(Integer.parseInt(caloriesBurnedBox.getText()));
                        } catch (NumberFormatException e) {
                              output("Please enter a number for calories
burned.");
                              errorMessage = true;
                        }
                  }
                  if (!errorMessage) {
                        if (!a.addWorkoutItem(workoutItem)) {
                              output("Workout is already in system. Use a
different name.");
                        } else {
                              fromAddWorkout = true;
                              changeToDietScreen();
                        }
                  }
            }
            if (source == calendarButton) {
                  calendarFromDiet = true;
                  changeToCalendarScreen();
            }
            if (source == foodBack) {
                  fromAddFood = true;
                  changeToDietScreen();
            }
            if (source == workoutBack) {
                  fromAddWorkout = true;
                  changeToDietScreen();
            }
            if (source == days) {
                  changeToDietScreen();
            }
            if (source == chooseFood) {
                  changeToChooseFoodScreen();
            }
            if (source == chooseWorkout) {
                  changeToChooseWorkoutScreen();
            }
            if (source == chooseFoodBack) {
                  fromChooseFood = true;
                  changeToDietScreen();
            }
            if (source == chooseWorkoutBack) {
                  fromChooseWorkout = true;
                  changeToDietScreen();
            }
            for (int i = 0; i < days.size(); i++) {
                  if (source == days.get(i)) {
                        dateFromButton = i + 1;
                        Context.setDayOfMonth(dateFromButton);
                        Context.setMonth(calendarDate.getMonth());
                        Context.setYear(calendarDate.getYear());
                        fileHandler.createDateFile();
```

```
                        fromCalendar = true;
                        changeToDietScreen();
                    }
                }
                if (source == chooseFoodConfirm) {
                    for (FoodItem f : foods) {
                        if (f != null) {
                            boolean errorMessage = false;
                            if
(choiceOfFood.getSelectedItem().equals(f.getName())) {
                                MealItem mealItem = new MealItem(f);
                                if (!meal.getSelectedItem().equals("Choose
Meal"))

        mealItem.setMeal(meal.getSelectedItem());
                                else {
                                    output("Please select a meal.");
                                    errorMessage = true;
                                }
                                if (!errorMessage) {
                                    try {

        mealItem.setPortions(Double.parseDouble(portionAmountBox.getText()));
                                    } catch (NumberFormatException e) {
                                        output("Please enter a number
for portions.");
                                        errorMessage = true;
                                    }
                                }
                                if (!errorMessage) {
                                    fileHandler.chooseFood(mealItem);
                                    fromChooseFood = true;
                                    changeToDietScreen();
                                }
                            } else if
(choiceOfFood.getSelectedItem().equals("Choose Food")) {
                                output("Please choose a food.");
                                break;
                            }
                        }
                    }
                }
                if (source == chooseWorkoutConfirm) {
                    for (WorkoutItem w : workouts) {
                        if (w != null) {
                            boolean errorMessage = false;
                            if
(choiceOfWorkout.getSelectedItem().equals(w.getName())) {
                                WorkoutSession workoutSession = new
WorkoutSession(w);
                                try {

        workoutSession.setSessionTime(Double.parseDouble(workoutDurationBox.getText())
);
                                } catch (NumberFormatException e) {
```

```
                                                output("Please enter a session duration
(hours).");
                                                errorMessage = true;
                                          }
                                          if (!errorMessage) {

      fileHandler.chooseWorkout(workoutSession);
                                                fromChooseWorkout = true;
                                                changeToDietScreen();
                                          }
                                    } else if
(choiceOfWorkout.getSelectedItem().equals("Choose Workout")) {
                                          output("Please choose a workout.");
                                          break;
                                    }
                              }
                        }
                  }
                  if (source == prevMonth) {
                        calendarDate.setToPrevMonth();
                        changeToCalendarScreen();
                  }

                  if (source == nextMonth) {
                        calendarDate.setToNextMonth();
                        changeToCalendarScreen();
                  }
      }

      public void changeToRegisterScreen() {

            title_register = addLabel("Register Section", 50, 50, 200, 50, this);
            username = addLabel("Username: ", 50, 100, 100, 50, this);
            usernameBox  = addTextField("", 175, 100, 175, 40, this);

            password = addLabel("Password: ", 50, 150, 100, 50, this);
            passwordBox  = addTextField("", 175, 150, 175, 40, this);

            confirmPassword = addLabel("Confirm Password: ", 50, 200, 120, 50,
this);
            confirmPasswordBox  = addTextField("", 175, 200, 175, 40, this);

            weight = addLabel("Weight (lbs): ", 50, 250, 120, 50, this);
            weightBox  = addTextField("", 175, 250, 175, 40, this);

            height = addLabel("Height (inches): ", 50, 300, 120, 50, this);
            heightBox  = addTextField("", 175, 300, 175, 40, this);

            age = addLabel("Age: ", 50, 350, 120, 50, this);
            ageBox  = addTextField("", 175, 350, 175, 40, this);

            gender = addLabel("Gender: ", 50, 400, 120, 50, this);
            genderBox  = addChoice("Choose Gender|Male|Female", 175, 415, 175, 40,
this);
```

```
        register_confirm = addButton("Confirm", 300, 450, 50, 40, this);

        setTitle("Register");
    setSize(400,525);
    usernameBox.setFont(new Font("Arial",0,20) );
    passwordBox.setFont(new Font("Arial",0,20) );
    confirmPasswordBox.setFont(new Font("Arial",0,20) );
    weightBox.setFont(new Font("Arial",0,20) );
    heightBox.setFont(new Font("Arial",0,20) );
    ageBox.setFont(new Font("Arial",0,20) );
    genderBox.setFont(new Font("Arial",0,20) );
    title_register.setFont(new Font("Arial",0,20) );

    title_welcome.removeNotify();
    newUser.removeNotify();
    returningUser.removeNotify();

    fromRegister = true;

}

public void changeToLoginScreen() {

       if (fromRegister) {
              title_login = addLabel("Log-In Section", 50, 50, 200, 50, this);
              login_confirm = addButton("Confirm", 300, 200, 50, 40, this);
              login_username = addLabel("Username: ", 50, 100, 100, 50, this);
              usernameLoginBox = addTextField("", 175, 100, 175, 40, this);
              login_password = addLabel("Password: ", 50, 150, 100, 50, this);
              passwordLoginBox  = addTextField("", 175, 150, 175, 40, this);

              setTitle("Login");
            setSize(400,275);
            title_login.setFont(new Font("Arial",0,20) );
            usernameLoginBox.setFont(new Font("Arial",0,20) );
            passwordLoginBox.setFont(new Font("Arial",0,20) );

       title_welcome.removeNotify();
              title_register.removeNotify();
              username.removeNotify();
              password.removeNotify();
              usernameBox.removeNotify();
              passwordBox.removeNotify();
              confirmPasswordBox.removeNotify();
              weightBox.removeNotify();
              heightBox.removeNotify();
              confirmPassword.removeNotify();
              weight.removeNotify();
              height.removeNotify();
              age.removeNotify();
              gender.removeNotify();
              ageBox.removeNotify();
              genderBox.removeNotify();
              register_confirm.removeNotify();
       } else {
```

```
                title_login = addLabel("Log-In Section", 50, 50, 200, 50, this);
                login_confirm = addButton("Confirm", 300, 200, 50, 40, this);
                login_username = addLabel("Username: ", 50, 100, 100, 50, this);
                usernameLoginBox = addTextField("", 175, 100, 175, 40, this);
                login_password = addLabel("Password: ", 50, 150, 100, 50, this);
                passwordLoginBox = addTextField("", 175, 150, 175, 40, this);

                setTitle("Login");
            setSize(400,275);
            title_login.setFont(new Font("Arial",0,20) );
            usernameLoginBox.setFont(new Font("Arial",0,20) );
            passwordLoginBox.setFont(new Font("Arial",0,20) );

            title_welcome.removeNotify();
            newUser.removeNotify();
            returningUser.removeNotify();
        }

    }

    public void changeToCalendarScreen() {
        if(prevMonth != null) {
            prevMonth.removeNotify();
        }
        if(nextMonth != null) {
            nextMonth.removeNotify();
        }
        prevMonth = addButton("<", 10, 30, 25, 25, this);
        nextMonth = addButton(">", 665, 30, 25, 25, this);

        if (title_calendar != null) {
            title_calendar.removeNotify();
        }
        if (weekdays != null) {
            weekdays.removeNotify();
        }
        title_calendar = addLabel(calendarDate.getMonth() + " " +
calendarDate.getYear(), 290, 30, 200, 25, this);
        weekdays = addLabel(Context.getWeekdays(), 35, 55, 700, 45, this);

        int firstDayCounter = 1;
        int currentDayCounter = 1;

        if(days != null) {
            for (Button day : days) {
                day.removeNotify();
            }
        }

        days = new ArrayList<Button>();

        out:
        for (int i = 0; i < 6; i++) {
            for (int j = 0; j < 7; j++) {
```

```
                        if (firstDayCounter >=
calendarDate.getWeekdayOfFirstDay()) {

      days.add(addButton(String.valueOf(currentDayCounter), j * 100, 100 + i * 100,
100, 100, this));
                              currentDayCounter++;
                        }
                        if (currentDayCounter > calendarDate.getDays()) {
                              break out;
                        }
                        firstDayCounter++;
                  }
            }

            setTitle("Calendar");
          setSize(700,700);
          title_calendar.setFont(new Font("Arial",0,20));
          weekdays.setFont(new Font("Arial",0,15));

            if (calendarFromLogin) {
            title_login.removeNotify();
                  login_confirm.removeNotify();
                  login_username.removeNotify();
                  usernameLoginBox.removeNotify();
                  login_password.removeNotify();
                  passwordLoginBox.removeNotify();
            } else if (calendarFromDiet) {
                  title_diet.removeNotify();
                diet_date.removeNotify();
                dailyFoods.removeNotify();
                  calendarButton.removeNotify();
                  addFood.removeNotify();
                  addWorkoutItem.removeNotify();
                  chooseFood.removeNotify();
                  chooseWorkout.removeNotify();
                  title_diet_workout.removeNotify();
                  dailyWorkouts.removeNotify();
                  caloriesFromFood.removeNotify();
                  caloriesBurnedFromWorkout.removeNotify();
                  title_diet_health.removeNotify();
                  healthReport.removeNotify();
            }

            calendarFromLogin = false;
            calendarFromDiet = false;
      }

      public void changeToDietScreen() {

            double netCalories = adjudicator.netCalories();
            String healthMessage = adjudicator.healthMessage(netCalories);

            diet_date = addLabel(calendarDate.getMonth() + " " + dateFromButton + ",
" + calendarDate.getYear(), 50, 45, 200, 50, this);
            title_diet = addLabel("Today's Foods", 50, 100, 200, 50, this);
```

```
            title_diet_workout = addLabel("Today's Workouts", 325, 100, 200, 50,
this);
            title_diet_health = addLabel("Health Report", 50, 400, 200, 50, this);

            addFood = addButton("Add New Food", 625, 100, 125, 40, this);
            chooseFood = addButton("Choose Food", 625, 150, 125, 40, this);
            addWorkoutItem = addButton("Add New Workout", 625, 200, 125, 40, this);
            chooseWorkout = addButton("Choose Workout", 625, 250, 125, 40, this);
            calendarButton = addButton("Calendar", 625, 300, 125, 40, this);

            dailyFoods = addList("", 50, 150, 250, 175, this);
            dailyWorkouts = addList("", 325, 150, 250, 175, this);
            healthReport = addList("You burn " +
Context.getUser().getTargetCalories() + " calories at rest every day on average." +
"|Net calories: " + netCalories + "|" + healthMessage, 50, 450, 525, 100, this);
            caloriesFromFood = addList("Total Calories Gained: " +
adjudicator.calculateDailyCalories(fileHandler.getMealItems()), 50, 350, 250, 30,
this);
            caloriesBurnedFromWorkout = addList("Total Calories Burned (workouts): "
+ adjudicator.calculateDailyCaloriesBurned(fileHandler.getWorkoutSessions()), 325,
350, 250, 30, this);

            dailyFoods.removeAll();
            dailyWorkouts.removeAll();

            fileHandler.printMealItems(dailyFoods);
            fileHandler.printWorkoutSessions(dailyWorkouts);

            setTitle("Diet Tracker");
          setSize(800,600);
          title_diet.setFont(new Font("Arial",0,20) );
          diet_date.setFont(new Font("Arial",0,20) );
          title_diet_workout.setFont(new Font("Arial",0,20));
          title_diet_health.setFont(new Font("Arial",0,20));

          if (fromAddFood) {
            title_addFood.removeNotify();
                  addFoodConfirm.removeNotify();
                  foodBack.removeNotify();
                  foodName.removeNotify();
                  foodNameBox.removeNotify();
                  calories.removeNotify();
                  caloriesBox.removeNotify();
          } else if (fromAddWorkout) {
            title_addWorkout.removeNotify();
                  addWorkoutConfirm.removeNotify();
                  workoutBack.removeNotify();
                  workoutName.removeNotify();
                  workoutNameBox.removeNotify();
                  caloriesBurned.removeNotify();
                  caloriesBurnedBox.removeNotify();
          } else if (fromChooseFood) {
            title_chooseFood.removeNotify();
            chooseFoodConfirm.removeNotify();
            chooseFoodBack.removeNotify();
```

```
            choiceOfFood.removeNotify();
            meal.removeNotify();
                  portionAmount.removeNotify();
                  portionAmountBox.removeNotify();
         } else if (fromChooseWorkout) {
            title_chooseWorkout.removeNotify();
            chooseWorkoutConfirm.removeNotify();
            chooseWorkoutBack.removeNotify();
            choiceOfWorkout.removeNotify();
            workoutDuration.removeNotify();
            workoutDurationBox.removeNotify();
         } else if (fromCalendar) {
            fileHandler.createDateFile();
            title_calendar.removeNotify();
            weekdays.removeNotify();
            for (Button button : days) {
                  button.removeNotify();
                  }
            prevMonth.removeNotify();
            nextMonth.removeNotify();
         } else {
              title_login.removeNotify();
                  login_confirm.removeNotify();
                  login_username.removeNotify();
                  usernameLoginBox.removeNotify();
                  login_password.removeNotify();
                  passwordLoginBox.removeNotify();
         }

         fromAddFood = false;
         fromAddWorkout = false;
         fromChooseFood = false;
         fromChooseWorkout = false;
         fromCalendar = false;

      }

      public void changeToAddFoodScreen() {

            title_addFood = addLabel("Add Food (per portion)", 50, 50, 300, 50,
this);
            addFoodConfirm = addButton("Add", 300, 200, 60, 50, this);
            foodBack = addButton("Back", 50, 200, 60, 50, this);

            foodName = addLabel("Food Name: ", 50, 100, 100, 50, this);
            foodNameBox  = addTextField("", 175, 100, 175, 40, this);

            calories = addLabel("Calories: ", 50, 150, 100, 50, this);
            caloriesBox  = addTextField("", 175, 150, 175, 40, this);

            setTitle("Add Food");
         setSize(400,300);
         title_addFood.setFont(new Font("Arial",0,20) );
         foodNameBox.setFont(new Font("Arial",0,20) );
         caloriesBox.setFont(new Font("Arial",0,20) );
```

```
            title_diet.removeNotify();
        diet_date.removeNotify();
            calendarButton.removeNotify();
            addFood.removeNotify();
            addWorkoutItem.removeNotify();
            chooseFood.removeNotify();
            chooseWorkout.removeNotify();
            dailyFoods.removeNotify();
            title_diet_workout.removeNotify();
            dailyWorkouts.removeNotify();
            caloriesFromFood.removeNotify();
            caloriesBurnedFromWorkout.removeNotify();
            title_diet_health.removeNotify();
            healthReport.removeNotify();

    }

    public void changeToAddWorkoutItemScreen() {

            title_addWorkout = addLabel("Add Workout (per hour)", 50, 50, 300, 50,
this);
            addWorkoutConfirm = addButton("Add", 300, 200, 60, 50, this);
            workoutBack = addButton("Back", 50, 200, 60, 50, this);

            workoutName = addLabel("Workout Name: ", 50, 100, 100, 50, this);
            workoutNameBox  = addTextField("", 175, 100, 175, 40, this);

            caloriesBurned = addLabel("Calories Burned: ", 50, 150, 100, 50, this);
            caloriesBurnedBox  = addTextField("", 175, 150, 175, 40, this);

            setTitle("Add Workout");
        setSize(400,300);
        title_addWorkout.setFont(new Font("Arial",0,20) );
        workoutNameBox.setFont(new Font("Arial",0,20) );
        caloriesBurnedBox.setFont(new Font("Arial",0,20) );

            title_diet.removeNotify();
        diet_date.removeNotify();
            calendarButton.removeNotify();
            addFood.removeNotify();
            addWorkoutItem.removeNotify();
            chooseFood.removeNotify();
            chooseWorkout.removeNotify();
            dailyFoods.removeNotify();
            title_diet_workout.removeNotify();
            dailyWorkouts.removeNotify();
            caloriesFromFood.removeNotify();
            caloriesBurnedFromWorkout.removeNotify();
            title_diet_health.removeNotify();
            healthReport.removeNotify();

    }

    public void changeToChooseFoodScreen() {
```

```
            foods = fileHandler.getFoodItems();
            foodNames = "";

            for (FoodItem f : foods) {
                  foodNames += "|" + f.getName();
            }

            choiceOfFood = addChoice("Choose Food" + foodNames,50,100,200,50,this);
            meal = addChoice("Choose
Meal|Breakfast|Lunch|Dinner|Snack",50,150,200,50,this);
            portionAmount = addLabel("How many portions?",50,180,200,50,this);
            portionAmountBox = addTextField("",50,230,200,30,this);

            title_chooseFood = addLabel("Choose Food", 50, 50, 200, 50, this);
            chooseFoodConfirm = addButton("Confirm", 300, 280, 60, 50, this);
            chooseFoodBack = addButton("Back", 50, 280, 60, 50, this);

            setTitle("Choose Food");
          setSize(400,400);
          title_chooseFood.setFont(new Font("Arial",0,20));
          choiceOfFood.setFont(new Font("Arial",0,20));
          meal.setFont(new Font("Arial",0,20));
          portionAmount.setFont(new Font("Arial",0,20));
          portionAmountBox.setFont(new Font("Arial",0,20));

            title_diet.removeNotify();
          diet_date.removeNotify();
            calendarButton.removeNotify();
            addFood.removeNotify();
            addWorkoutItem.removeNotify();
            chooseFood.removeNotify();
            chooseWorkout.removeNotify();
            dailyFoods.removeNotify();
            title_diet_workout.removeNotify();
            dailyWorkouts.removeNotify();
            caloriesFromFood.removeNotify();
            caloriesBurnedFromWorkout.removeNotify();
            title_diet_health.removeNotify();
            healthReport.removeNotify();

      }

      public void changeToChooseWorkoutScreen() {

            workouts = fileHandler.getWorkoutItems();
            workoutNames = "";

            for (WorkoutItem w : workouts) {
                  workoutNames += "|" + w.getName();
            }

            choiceOfWorkout = addChoice("Choose Workout" +
workoutNames,50,100,200,50,this);
            workoutDuration = addLabel("Workout Duration:",50,130,200,50,this);
```

```
            workoutDurationBox = addTextField("",50,180,200,30,this);

            title_chooseWorkout = addLabel("Add Workout", 50, 50, 200, 50, this);
            chooseWorkoutConfirm = addButton("Add", 300, 230, 60, 50, this);
            chooseWorkoutBack = addButton("Back", 50, 230, 60, 50, this);

            setTitle("Choose Workout");
        setSize(400,330);
        title_chooseWorkout.setFont(new Font("Arial",0,20));
        choiceOfWorkout.setFont(new Font("Arial",0,20));
        workoutDuration.setFont(new Font("Arial",0,20));
        workoutDurationBox.setFont(new Font("Arial",0,20));

            title_diet.removeNotify();
        diet_date.removeNotify();
            calendarButton.removeNotify();
            addFood.removeNotify();
            addWorkoutItem.removeNotify();
            chooseFood.removeNotify();
            chooseWorkout.removeNotify();
            dailyFoods.removeNotify();
            title_diet_workout.removeNotify();
            dailyWorkouts.removeNotify();
            caloriesFromFood.removeNotify();
            caloriesBurnedFromWorkout.removeNotify();
            title_diet_health.removeNotify();
            healthReport.removeNotify();

    }
}
```

Model Package

```
package ca.tdsb.vpci.jasonzhu.healthier.model;

public class FoodItem {

	private double calories;
	private String name;

	public void setCalories(double d) {
		this.calories = d;
	}

	public double getCalories() {
		return this.calories;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getName() {
		return this.name;
	}

	public String toString() {
		return name + ", " + calories;
	}

}
```

```java
package ca.tdsb.vpci.jasonzhu.healthier.model;

public class MealItem extends FoodItem {

	private String name;
	private String meal;
	private double calories;
	private double portions;

	public MealItem(FoodItem foodItem) {

		name = foodItem.getName();
		calories = foodItem.getCalories();

	}

	public MealItem() {}

	public String getMeal() {
		return meal;
	}

	public void setMeal(String meal) {
		this.meal = meal;
	}

	public double getPortions() {
		return portions;
	}

	public void setPortions(double d) {
		this.portions = d;
	}

	public String toString() {

		return name + ", " + calories + ", " + meal + ", " + portions;

	}

}
```

```java
package ca.tdsb.vpci.jasonzhu.healthier.model;

public class User {

	private double age;
	private double weight;
	private double height;

	private String gender;
	private String username;
	private String password;
	private String confirmPassword;
	private double targetCalories;

	public void setAge(double age) {
		this.age = age;
	}

	public double getAge() {
		return this.age;
	}

	public void setWeight(double weight) {
		this.weight = weight;
	}

	public double getWeight() {
		return this.weight;
	}

	public void setHeight(double height) {
		this.height = height;
	}

	public double getHeight() {
		return this.height;
	}

	public void setGender(String gender) {
		this.gender = gender;
	}

	public String getGender() {
		return this.gender;
	}

	public void setUsername(String username) {
		this.username = username;
	}

	public String getUsername() {
		return this.username;
	}

	public void setPassword(String password) {
```

```
		this.password = password;
	}

	public String getPassword() {
		return this.password;
	}

	public void setConfirmPassword(String confirmPassword) {
		this.confirmPassword = confirmPassword;
	}

	public String getConfirmPassword() {
		return this.confirmPassword;
	}

	public double getTargetCalories() {
		return targetCalories;
	}

	public void setTargetCalories(double targetCalories) {
		this.targetCalories = targetCalories;
	}

	public String toString() {
		return username + ", " + password + ", " + confirmPassword + ", " + age
+ ", " + weight + ", " + height + ", " + gender + ", " + targetCalories;
	}

}
```

```java
package ca.tdsb.vpci.jasonzhu.healthier.model;

public class WorkoutItem {

	private double caloriesBurned;
	private String name;

	public void setCaloriesBurned(double d) {
		this.caloriesBurned = d;
	}

	public double getCaloriesBurned() {
		return this.caloriesBurned;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getName() {
		return this.name;
	}

	public String toString() {
		return name + ", " + caloriesBurned;
	}

}
```

```java
package ca.tdsb.vpci.jasonzhu.healthier.model;

public class WorkoutSession extends WorkoutItem {

	private String name;
	private double caloriesBurned;
	private double sessionTime;

	public WorkoutSession(WorkoutItem workoutItem) {

		name = workoutItem.getName();
		caloriesBurned = workoutItem.getCaloriesBurned();

	}

	public WorkoutSession() {}

	public double getSessionTime() {
		return sessionTime;
	}

	public void setSessionTime(double sessionTime) {
		this.sessionTime = sessionTime;
	}

	public String toString() {

		return name + ", " + caloriesBurned + ", " + sessionTime;

	}

}
```