*/\*  
Matthew Stulz  
CSC 425  
Programming Assignment #3  
Due April 3, 2017  
This program is an implementation of a solar panel controller that uses Fuzzy Logic for the implementation. The solar  
panel has an angle that it is set at for each given season and also rotates through the hours of the day following the  
sun. The fuzzy membership functions are implemented with the time of day (rated on max sun we can get) along with the  
current watts that the solar panel is outputting (200 being the maximum amount). With these two fuzzy values, I used  
Fuzzy Logic to OR these two and through the implication of these fuzzy sets, I came up with hard values (the change in  
angles to maximize the sun to regain the most wattage output.  
\*/***public class** Main {  
 **public static void** main(String[] args) {  
  
 *// Used current time for this solar panel* SunriseSunset s = **new** SunriseSunset();  
 s.output();  
 Fuzzy f = **new** Fuzzy(**"northern"**, **"spring"**, s.getLatitude().doubleValue());  
 **double** fuzzyWatts = f.fuzzificationOfWatts(160); *// membership function calls* **double** fuzzyTime = s.fuzzificationOfTimeOfDay(); *// membership function calls* **double** fuzzyValue = f.inferenceFuzzyLogic(fuzzyWatts, fuzzyTime); *// Fuzzy logic call* String fuzzyImplication = f.implicationOfFuzzySet(fuzzyValue, s, f); *// Gets hard numbers* System.***out***.println(fuzzyImplication+**"\n"**);  
  
 *// Re-adjusts the season angle since the input is for winter.* Fuzzy f2 = **new** Fuzzy(**"northern"**, **"winter"**, 47.551493);  
 fuzzyWatts = f2.fuzzificationOfWatts(40);  
 fuzzyTime = s.fuzzificationOfTimeOfDay();  
 fuzzyValue = f2.inferenceFuzzyLogic(fuzzyWatts, fuzzyTime);  
 fuzzyImplication = f2.implicationOfFuzzySet(fuzzyValue, s, f2);  
 System.***out***.println(fuzzyImplication+**"\n"**);  
  
 *// Third example ignores any fuzzy logic calculations because the input watts are currently at 200 (maximum).* Fuzzy f3 = **new** Fuzzy(**"northern"**, **"summer"**, 56.130366);  
 fuzzyWatts = f3.fuzzificationOfWatts(200);  
 fuzzyTime = s.fuzzificationOfTimeOfDay();  
 fuzzyValue = f3.inferenceFuzzyLogic(fuzzyWatts, fuzzyTime);  
 fuzzyImplication = f3.implicationOfFuzzySet(fuzzyValue, s, f3);  
 System.***out***.println(fuzzyImplication+**"\n"**);  
  
 *// Rotated to 145° to be at the prime angle for when the sun rises.* Fuzzy f4 = **new** Fuzzy(**"northern"**, **"spring"**, 40.712784);  
 fuzzyWatts = f4.fuzzificationOfWatts(160);  
 fuzzyTime = s.fuzzificationOfTimeOfDay();  
 fuzzyValue = f4.inferenceFuzzyLogic(fuzzyWatts, fuzzyTime);  
 fuzzyImplication = f4.implicationOfFuzzySet(fuzzyValue, s, f4);  
 System.***out***.println(fuzzyImplication+**"\n"**);  
 }  
}

**import** java.math.BigDecimal;  
**import** java.util.Calendar;  
**import** java.util.Date;  
**import** com.luckycatlabs.sunrisesunset.SunriseSunsetCalculator;  
**import** com.luckycatlabs.sunrisesunset.dto.Location;  
  
*// This class uses a dependency through Maven that calculates sunrise, sunset, and works  
// together with the Date and Calendar classes to calculate time of day. This is important  
// to ensure the panel is facing the sun during sunrise, is at a 180° degree angle at high  
// noon and follows it until sunset. Then resetting to face east once the time is greater  
// than 20:00.***public class** SunriseSunset {  
  
 *// Reference variables used* **private** String **officialSunrise**;  
 **private** String **officialSunset**;  
 **private** Date **today**;  
 **private** SunriseSunsetCalculator **calculator**;  
 **private** Location **location**;  
 **private** String **direction**;  
  
 *// SunriseSunset constructor that is a no arg constructor but initializes the fields since a lot  
 // of them don't require input and are creating instances of classes.* **public** SunriseSunset() {  
 **this**.**location** = **new** Location(**"39.033117"**, **"-84.451885"**); *// Highland Heights, KY* **this**.**calculator** = **new** SunriseSunsetCalculator(**location**, **"America/New\_York"**);  
 setDate();  
 setOfficialSunrise();  
 setOfficialSunset();  
 }  
  
 *// Setters and getters* **public void** setDate() {  
 **this**.**today** = Calendar.*getInstance*().getTime();  
 }  
 **public** Date getDate() {  
 **return today**;  
 }  
 **public void** setOfficialSunrise() {  
 **this**.**officialSunrise** = **calculator**.getOfficialSunriseForDate(Calendar.*getInstance*());  
 }  
 **public** String getOfficialSunrise() {  
 **return officialSunrise**;  
 }  
 **public void** setOfficialSunset() {  
 **this**.**officialSunset** = **calculator**.getOfficialSunsetForDate(Calendar.*getInstance*());  
 }  
 **public** String getOfficialSunset() {  
 **return officialSunset**;  
 }  
 **public** BigDecimal getLatitude() {  
 **return location**.getLatitude();  
 }  
 **public void** setDirection(String direction) { **this**.**direction** = direction; }  
 **public** String getDirection() { **return direction**; }  
  
 *// Checks for the northern hemisphere so see if the overall angle of the panel is correct per the given season  
 // by checking the current date compared to stored dates.* **public** String checkCalenderDate() {  
 Calendar min = Calendar.*getInstance*();  
 Calendar max = Calendar.*getInstance*();  
 Calendar current = Calendar.*getInstance*();  
 min.set(2017, Calendar.***APRIL***,18);  
 max.set(2017, Calendar.***AUGUST***, 24);  
 **if** (current.before(max) && current.after(min)) {  
 *// Adjust to summer angle* **return "summer"**;  
 }  
 min.set(2017, Calendar.***AUGUST***, 24);  
 max.set(2017, Calendar.***OCTOBER***, 7);  
 **if** (current.before(max) && current.after(min)) {  
 *// Adjust to fall/spring angle* **return "spring"**;  
 }  
 min.set(2017, Calendar.***OCTOBER***, 7);  
 max.set(2017, Calendar.***MARCH***, 5);  
 **if** (current.before(max) && current.after(min)) {  
 *// Adjust to winter angle* **return "fall"**;  
 }  
 min.set(2017, Calendar.***MARCH***, 7);  
 max.set(2017, Calendar.***APRIL***, 5);  
 **if** (current.before(max) && current.after(min)) {  
 *// Adjust to spring angle* }  
 **return "spring"**;  
 }  
  
 *// Membership function that returns a fuzzy value based on the time of day where  
 // we can get the greatest output from the panel.* **public double** fuzzificationOfTimeOfDay() {  
 *// Gets the hour only and returns a fuzzy value* Calendar c = Calendar.*getInstance*();  
 c.setTime(getDate());  
 **int** hour = c.get(Calendar.***HOUR\_OF\_DAY***);  
 **if**(hour >= 5) {  
 **return** 0.0;  
 } **else if** (hour >= 6) {  
 **return** 0.1;  
 } **else if** (hour >= 7) {  
 **return** 0.3;  
 } **else if** (hour >= 8) {  
 **return** 0.5;  
 } **else if** (hour >= 10) {  
 **return** 0.7;  
 } **else if** (hour >= 11) {  
 **return** 0.9;  
 } **else if** (hour >= 12) {  
 **return** 1.0;  
 } **else if** (hour >= 13) {  
 **return** 0.9;  
 } **else if** (hour >= 14) {  
 **return** 0.8;  
 } **else if** (hour >= 15) {  
 **return** 0.7;  
 } **else if** (hour >= 16) {  
 **return** 0.6;  
 } **else if** (hour >= 17) {  
 **return** 0.4;  
 } **else if** (hour >= 18) {  
 **return** 0.3;  
 } **else if** (hour >= 19) {  
 **return** 0.2;  
 } **else if** (hour >= 20) {  
 **return** 0.1;  
 } **else** {  
 **return** 0.0;  
 }  
 }  
  
 *// These degrees are calculated as if the solar panel is flat at 180°.* **public int** calculateOptimalTimeOfDayAngle() {  
 Calendar c = Calendar.*getInstance*();  
 c.setTime(getDate());  
 **int** hour = c.get(Calendar.***HOUR\_OF\_DAY***);  
 **if**(hour == 5) {  
 setDirection(**"east"**);  
 **return** 145;  
 } **else if** (hour == 6) {  
 setDirection(**"east"**);  
 **return** 150;  
 } **else if** (hour == 7) {  
 setDirection(**"east"**);  
 **return** 155;  
 } **else if** (hour == 8) {  
 setDirection(**"east"**);  
 **return** 160;  
 } **else if** (hour == 10) {  
 setDirection(**"east"**);  
 **return** 165;  
 } **else if** (hour == 11) {  
 setDirection(**"east"**);  
 **return** 170;  
 } **else if** (hour == 12) {  
 setDirection(**"north"**);  
 **return** 180;  
 } **else if** (hour == 13) {  
 setDirection(**"west"**);  
 **return** 175;  
 } **else if** (hour == 14) {  
 setDirection(**"west"**);  
 **return** 170;  
 } **else if** (hour == 15) {  
 setDirection(**"west"**);  
 **return** 165;  
 } **else if** (hour == 16) {  
 setDirection(**"west"**);  
 **return** 160;  
 } **else if** (hour == 17) {  
 setDirection(**"west"**);  
 **return** 155;  
 } **else if** (hour == 18) {  
 setDirection(**"west"**);  
 **return** 150;  
 } **else if** (hour == 19) {  
 setDirection(**"west"**);  
 **return** 145;  
 } **else if** (hour == 20) {  
 setDirection(**"west"**);  
 **return** 140;  
 } **else** {  
 setDirection(**"east"**);  
 **return** 145;  
 }  
 }  
  
 *// Outputs the sunrise and the sunset of the SunriseSunset constructor* **public void** output() {  
 System.***out***.println(getDate() + **" officialSunrise : "** + getOfficialSunrise());  
 System.***out***.println(getDate() + **" officialSunset : "** + getOfficialSunset());  
 }  
}

*// This class embodies nearly all the Fuzzy Logic and optimizes the overall angle during said season.***public class** Fuzzy {  
 *// 1. Fuzzification  
 // 2. Inference  
 // 3. Defuzzification  
  
 // private reference variables* **private** String **hemisphere**;  
 **private** String **season**;  
 **private double latitude**;  
  
 *// The fuzzy constructor that sets the hemisphere, season, and latitude.* **public** Fuzzy(String hemisphere, String season, **double** latitude) {  
 **this**.**hemisphere** = hemisphere;  
 **this**.**season** = season;  
 **this**.**latitude** = latitude;  
 }  
  
 *// Setters and getters* **public** String getHemisphere() {  
 **return hemisphere**;  
 }  
 **public void** setHemisphere(String hemisphere) {  
 **this**.**hemisphere** = hemisphere;  
 }  
 **public** String getSeason() {  
 **return season**;  
 }  
 **public void** setSeason(String season) {  
 **this**.**season** = season;  
 }  
 **public double** getLatitude() {  
 **return latitude**;  
 }  
 **public void** setLatitude (**double** latitude) {  
 **this**.**latitude** = latitude;  
 }  
  
 *// Will be used to change overall angle via certain dates  
 // Northern hemisphere Southern hemisphere  
 // Adjust to summer angle on April 18 October 18  
 // Adjust to autumn angle on August 24 February 23  
 // Adjust to winter angle on October 7 April 8  
 // Adjust to spring angle on March 5 September 4* **public double** calculateOptimalSeasonalAngle(Fuzzy input) {  
 **double** angle;  
 *// Checks the season and calculates the angle the solar panel will stay while it rotates through the  
 // daytime hours to maximize sun coverage.* **if** (input.getSeason().equals(**"summer"**)) {  
 angle = (input.getLatitude() \* 0.92) - 24.3;  
 System.***out***.println(**"The current angle has been adjusted to "** + angle + **"° to optimize for the summer season."**);  
 **return** angle;  
 } **else if** (input.getSeason().equals(**"spring"**) || (input.getSeason().equals(**"fall"**))) {  
 angle = (input.getLatitude() \* 0.98) - 2.3;  
 System.***out***.println(**"The current angle has been adjusted to "** + angle + **"° to optimize for the spring/fall season."**);  
 **return** angle;  
 } **else** angle = (input.getLatitude() \* 0.89) + 24;  
 System.***out***.println(**"The current angle has been adjusted to "** + angle + **"° to optimize for the winter season."**);  
 **return** angle;  
 }  
  
 *// If the fuzzy value of current watts or time of day is not one,  
 // there are improvements that can be made to optimize the sun.  
 // 1. Check the optimalAngle and adjust accordingly  
 // 2. Check the time of day and determine rotation* **public** String implicationOfFuzzySet(**double** fuzzyValue, SunriseSunset s, Fuzzy f) {  
 **if** (fuzzyValue != 1.0) {  
 *// Check to see if the current season angle is correct* String currentSeasonAngle = s.checkCalenderDate();  
 **if** (f.getSeason().equals(currentSeasonAngle)) {  
 *// Angled correctly for the given season* } **else** {  
 *// Set the current season to f* f.setSeason(currentSeasonAngle);  
 *// Calculate and se the new seasonal angle* calculateOptimalSeasonalAngle(f);  
 **return "The panel has been re-adjusted for the current season."**;  
 }  
 *// Re-adjust rotation of the panel according to time of day* **int** rotationDegree = s.calculateOptimalTimeOfDayAngle();  
 **return "With the current time: "** + s.getDate() + **", the panel has been adjusted to "** + rotationDegree + **"° due "** + s.getDirection() +  
 **" so that the panel is rotated for maximum sun optimization."**;  
 } **else** {  
 **return "There are currently no adjustments of the panel, maximized optimization is underway."**;  
 }  
 }  
  
 *// Arbitrary solar panel output of 200 watts per hour, the max output said solar panel  
 // can output being maximized.  
 // Will be used to rotate to the angle of the sun to ensure optimization of the sun.* **public double** fuzzificationOfWatts(**int** watts) {  
 **if** (watts == 200) {  
 **return** 1.0;  
 } **else if** (watts >= 180) {  
 **return** 0.9;  
 } **else if** (watts >= 160) {  
 **return** 0.8;  
 } **else if** (watts >= 140) {  
 **return** 0.7;  
 } **else if** (watts >= 120) {  
 **return** 0.6;  
 } **else if** (watts >= 100) {  
 **return** 0.5;  
 } **else if** (watts >= 80) {  
 **return** 0.4;  
 } **else if** (watts >= 60) {  
 **return** 0.3;  
 } **else if** (watts >= 40) {  
 **return** 0.2;  
 } **else if** (watts >= 20) {  
 **return** 0.1;  
 } **else** {  
 **return** 0.0;  
 }  
 }  
  
 *// Uses the fuzzy OR logic on the two fuzzy membership functions* **public double** inferenceFuzzyLogic(**double** fuzzyWatts, **double** fuzzyTime) {  
 **return** (fuzzyWatts>fuzzyTime?fuzzyWatts:fuzzyTime);  
 }  
  
}

I decided on a solar panel controller that changes its angle and rotates along with the sun during the daytime hours that checks the optimum output through fuzzy logic. I decided on fuzzy logic since it seems that when implementing a controller of some sort that it just fits with fuzzy logic since there are real data inputs and real data outputs for the controller to follow. Fuzzy values are getting the time of day and the output wattage. These are OR’d together to avoid problems with a cloudy day so even if the wattage output should be optimized but isn’t, the controller gets the time of day and still is in the perfect position. Each season there is an overall angle that the panel should be at which is calculated through the Calendar class.

I’d like the controller to know the location so less input would be required. I think I covered the main concepts of a solar panel controller but I’m sure there are other variables that I didn’t take into consideration such as implying that the panel is installed in the correct direction, the overall size so when it does rotate or tilt to the correct angle, that the panel doesn’t damage itself. I think this uncertainty approach is the best for this domain but just could be how fuzzy logic clicks with how I viewed this domain and implementation.