**Group Project 2 – Bank Group**

**Daniele Ricciardelli**

**Instructions/Objectives**

A picture containing diagram, line

Description automatically generatedA community has 7 swimming pools, as shown in the following map.

Your Project must satisfy the following requirements:

1. Create three classes: **Pool, Temperature**, and **Location**.

2. A swimming ***pool*** shall have an object of “***Temperature***” and an object of “***Location***” as data

members. (“Compositionally” speaking: A ***pool*** has-a temperature, and a ***pool*** has-a location)

3. A *temperature* has “degree” and “scale”.

4. A *location* is defined by (*x, y*); hence a location has an ‘x’ value and a ‘y’ value.

5. Provide ***ToString*** method for all three classes.

6. Provide other member methods for each of the classes properly . For example, a ***FindDistance*** method to find the distance between pools.

7. All methods shall be public.

8. The **Pool** class shall have a ***static*** data member ‘Count’ The driver program shall print out the static member ***count*** before any pool object is created, and also after each pool is instantiated.

9. A maintenance crew can set the temperature for each pool within the range of 98 oF and 104 oF.

Note: generate a **random number** between [*98, 104*].

10. A maintenance person will go through all seven pools and set the temperatures. The person starts from location (0,0) and will go to the ***nearest pool*** after she/he finishes the job. The process continues until all the pools are visited. Each pool can only be visited once.

Your driver program determines and displays the route she/he takes to visit all the pools,

such as:

**(0,0) >> B with temperature at 99 degree F >> C with temperature at 103 degree F >> ... ....**

**Pseudo-code**

Temperature Class

* Has protected data members degree (double) and scale (bool).
* Has a default constructor and an overloaded constructor that takes a double and a boolean.
* Contains properties to get and set data member values.
* Has public method ToF to return the degree value in Fahrenheit.
  + If scale is true, return the degree.
  + If scale is false, convert the degree to Fahrenheit, return degree and set scale to true.
* Has public method ToC to return the degree value in Celsius.
  + If scale is true, convert the degree to Celsius, return degree and set scale to false.
  + If scale is false, return the degree.
* ToString method that will display the temperature degree and scale.
* Has a destructor.

Location Class

* Has protected data members x (int) and y (int).
* Has a default constructor and an overloaded constructor that takes two int values.
* Contains properties to get and set data member values.
* ToString method that will indicate the position of the pool.
* Has a destructor.

Pool Class

* Has data members temp, an object of Temperature, and location, and object of Location.
* Has maintDone as a Boolean data member to indicate whether or not we have already visited the same pool.
* Also has a static int, Count, to be increment after each pool is instantiated.
* Has default constructor and an overloaded constructor that takes Temperature and Location objects.
* Has method ToString that will display the temperature and location of the pool.
* Has method FindDistance to find the distance between pools.
* Has properties for maintdone, location, and temperature to get and set the values.
* Has method setTemp to set the temperature at the pool and to mark that the pool has been visited.
* Has a destructor.

Main Driver Program

* Instantiate an array of pools of object type Pool.
  + In the argument for creating each pool, set the location of each individual pool.
* Create a loop for the path of maintenance.
  + Use setTemp method to set the temperature and to mark that the pool has been visited.
  + Input the location and the temperature of the pool into a string array.
  + Find the nearest pool with nextPool function.nextPool function.
* Create an array of object type double to hold the distances to the next pool.
* First loop
  + Fill the array with the distances to the next pool using FindDistance method.
* Second loop
  + Sort the distances to find the smallest distance.
  + Return the index value of the smallest distance.

**Code (Driver program)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace groupProject2Final

{

class Program

{

static void Main(string[] args)

{

// Console.WriteLine("step 1");

//Console.ReadKey();

Console.WriteLine($"Starting Program...\nCurrent Pool count is: {Pool.count+1}.");

Pool[] pool = new Pool[7];

pool[0] = new Pool(new Temperature(), new Location(1, 3)); //B

pool[1] = new Pool(new Temperature(), new Location(4, 2)); //C

pool[2] = new Pool(new Temperature(), new Location(4, 8)); //A

pool[3] = new Pool(new Temperature(), new Location(6, 6)); //G

pool[4] = new Pool(new Temperature(), new Location(10, 5)); //F

pool[5] = new Pool(new Temperature(), new Location(12, 9)); //E

pool[6] = new Pool(new Temperature(), new Location(13, 1)); //D

// pool[0].setTemp();

//int index = nextPool(pool[0], pool);

//pool[index].setTemp();

//index = nextPool(pool[index],pool);

int index = 6;

string[] route = new string[pool.Length];

Console.WriteLine("The Maintenance Man went to:\nStarts at location (0,0) -->");

// Console.ReadKey();

for (int i = 0; i < pool.Length; i++)//repeats pool.length times

{

if(i==0)//so maintenance man start at 0

{

index = NextPool(pool);//for 0,0 start

pool[index].SetTemp();

route[i] = pool[index].Location.ToString() + " " + pool[index].Temperature.ToString();

Console.Write($"pool[{index}] {route[i]} ");

if (i != pool.Length - 1)

Console.WriteLine("-->");

index = NextPool(pool[index], pool);//sets the index for the next iteration

}

else

{

pool[index].SetTemp();//sets the temp at the pool

route[i] = pool[index].Location.ToString() + " " + pool[index].Temperature.ToString();

Console.Write($"pool[{index}] {route[i]} ");

if (i != pool.Length - 1)

Console.WriteLine("-->");

index = NextPool(pool[index], pool);//sets the index for the next iteration

}

/\*

pool[index].SetTemp();//sets the temp at the pool

route[i] = pool[index].Location.ToString() + " " + pool[index].Temperature.ToString();

Console.Write($"pool[{index}] {route[i]} ");

if(i!=pool.Length-1)

Console.WriteLine("-->");

index = NextPool(pool[index], pool);//sets the index for the next iteration\*/

//"Starting at ({startx},{starty}) >> ({.location.location\_x},{.location.location\_y). {.temp.ToString()})

//if ((index == 0) && (i != 0)) break; // If all pools are done, or nextPool fails.

}

// 0,0) >> B with temperature at 99 degree F >> C with temperature at 103 degree F >> ... ....

// distances[0]=pool[0].FindDistance(pool[1]);//this will find the distance between pool 0 and pool 1

//Console.WriteLine("tetststtststststtststs");

//testing convert between celsius and farenheit

//pool[0].Temperature.ToC();

// Console.WriteLine(pool[0].ToString());

Console.ReadLine();

//testing convert between celsius and farenheit

}

public static int NextPool(Pool[] myPools)//for the maintenance man to start at 0,0

{

int indexLow = 0;

double[] distances = new double[myPools.Length];

for (int i = 0; i < myPools.Length; i++)

{

distances[i] = Pool.FindDistanceFromOrigin(myPools[i]);

}

double lowest = double.MaxValue;

for (int j = 0; j < myPools.Length; j++)

{

if ((distances[j] < lowest) && (distances[j] > 0) && (myPools[j].MaintDone == false))

{

lowest = distances[j];

indexLow = j;

}

}

return indexLow;

}

public static int NextPool(Pool currentPool, Pool[] myPools)

{

int indexLow = 0;

double[] distances = new double[myPools.Length];

for (int i = 0; i < myPools.Length ; i++)

{

distances[i] = currentPool.FindDistance(myPools[i]);

}

double lowest = double.MaxValue;

for (int j = 0; j < myPools.Length; j++)

{

if ((distances[j] < lowest) && (distances[j] > 0) && (myPools[j].MaintDone == false))

{

lowest = distances[j];

indexLow = j;

}

}

return indexLow;

}

/\*

pool[i].SetTemp

Console.WriteLine($"Starting at ({startx},{starty}) >> ({.location.location\_x},{.location.location\_y). {.temp.ToString()})

\*/

}

}

**Code (Pool class)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace groupProject2Final

{

class Pool

{

// Data Members

protected Temperature temp;

protected Location location;

public static int count = -1; // array begins at zero.

protected bool maintDone = false;

public Pool()

{

count++;

Console.WriteLine($"New Pool created... current Pool count is: {Pool.count+1}.");

}

// Constructor

public Pool(Temperature set\_t, Location set\_l)

{

temp = set\_t;

location = set\_l;

count++;

Console.WriteLine($"New Pool created... current Pool count is: {Pool.count+1}.");

}

// Methods

public override string ToString()

{

string str;

str = $"The pool count is {count}, maintenance status: {maintDone}, Location: {location.ToString()}, Temp: {temp.ToString()}. ";//do this later

return str;

}

// DISTANCE FROM ORIGIN, OR DISTANCE FROM ANAOTHER POOL??

public double FindDistance(Pool nextPool)

{

double distance = 0.0;

distance = Math.Sqrt(Math.Pow(Convert.ToDouble(nextPool.location.Location\_x - this.location.Location\_x), 2) + Math.Pow(Convert.ToDouble(nextPool.location.Location\_y - this.location.Location\_y), 2));

//Console.WriteLine($"Distance from {this.location} to {nextPool.location} is : " + distance);

return distance;

}

public static double FindDistanceFromOrigin(Pool nextPool)

{

double distance = 0.0;

distance = Math.Sqrt(Math.Pow(Convert.ToDouble(nextPool.location.Location\_x ), 2) + Math.Pow(Convert.ToDouble(nextPool.location.Location\_y), 2));

//Console.WriteLine($"Distance from {this.location} to {nextPool.location} is : " + distance);

return distance;

}

public bool MaintDone

{

get { return maintDone; }

set { maintDone = value; }

}

public Location Location

{

get { return location; }

set { location = value; }

}

public Temperature Temperature

{

get { return temp; }

set { temp = value; }

}

public static Random rng = new Random();//in order to generate true random nums

public void SetTemp()

{

//Random rng = new Random();

temp.Degree = rng.Next(98, 105); // 98-104+1

temp.Scale = true; // is this F?

string temperatureString = temp.ToString(); // Get the string representation of the temperature

// Console.WriteLine(temperatureString); // Print the temperature

maintDone = true;

}

~Pool() { count--; }

}

}

**Code (Temperature class)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace groupProject2Final

{

class Temperature

{

// Data Members

protected double degree;

protected bool scale;

// Default constructor

public Temperature()

{

degree = 0;

scale = true;

}

// Overload constructor

public Temperature(double d, bool s)

{

degree = d;

scale = s;

}

// Get n Set

public double Degree

{

get { return degree; }

set { degree = value; }

}

public bool Scale

{

get { return scale; }

set { scale = value; }

}

public double ToF()

{

if (scale) return degree;

else

{

degree = (degree \* 9 / 5) + 32;

scale = true;

}

return degree;

}

public double ToC()

{

if (!scale) return degree;

else

{

degree = (degree - 32) \* 5 / 9;

scale = false;

}

return degree;

}

// Methods

public override string ToString()

{

string str;

if (scale == true)

{

str = "The temperature is " + degree + " fahrenheit";

}

else

{

str = "The temperature is " + degree + " celsius";

}

return str;

}

~Temperature() { }

}

}

**Code (Location class)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace groupProject2Final

{

class Location

{

// Data Members

protected int x;

protected int y;

// Default constructor

public Location()

{

x = 0;

y = 0;

}

// Overload constructor

public Location(int locX, int locY)

{

x = locX;

y= locY;

}

// Get n Set

public int Location\_x

{

get { return x; }

set { x = value; }

}

public int Location\_y

{

get { return y; }

set { y = value; }

}

// Methods

public override string ToString()

{

string str;

str = " (" + x + "," + y + ") ";

return str;

}

~Location() { }

}

}

**Screenshot**

**A picture containing text, screenshot, font, menu

Description automatically generated**

**UML**

A screenshot of a computer code

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence**A screenshot of a computer program

Description automatically generated with medium confidence**