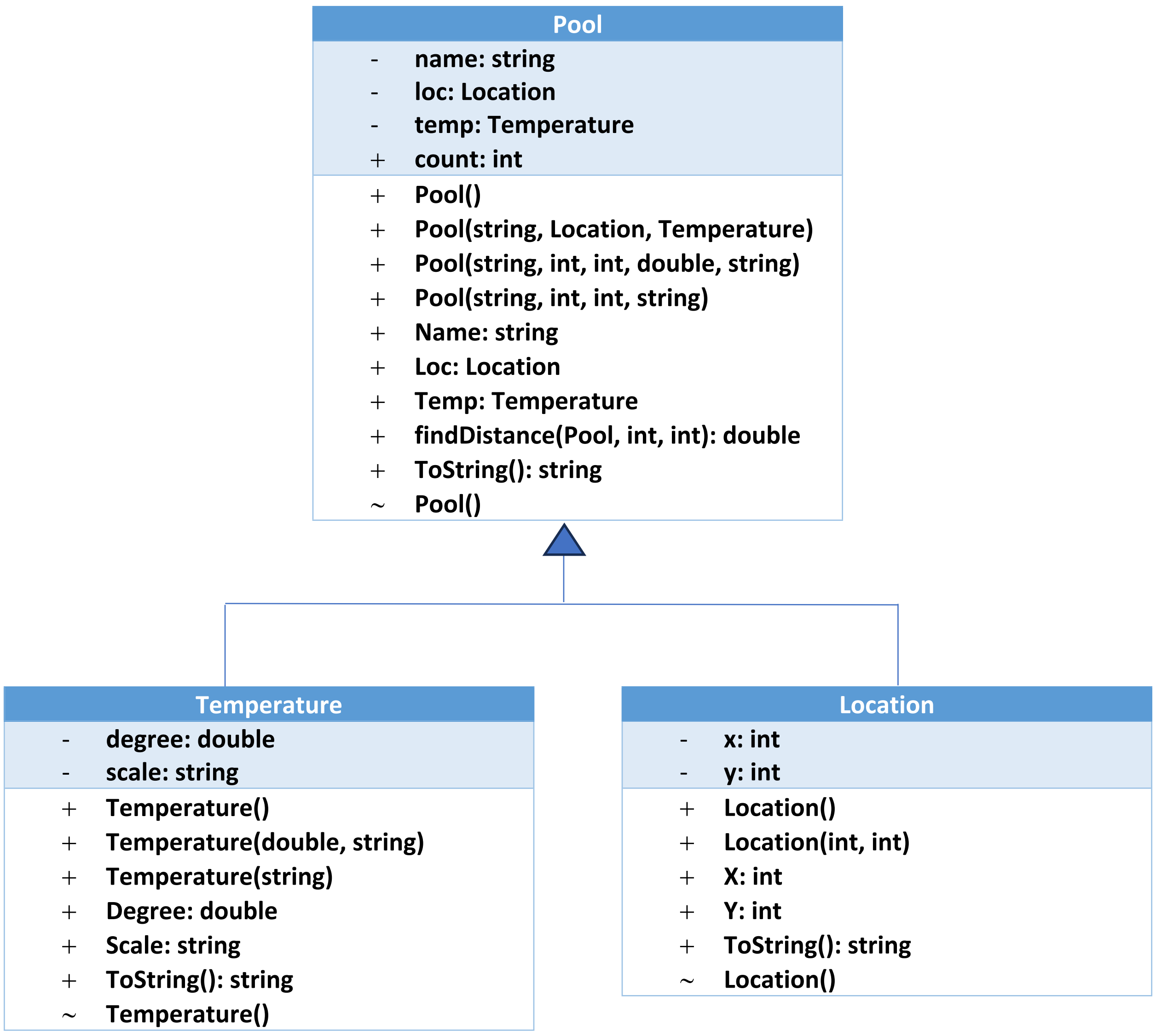
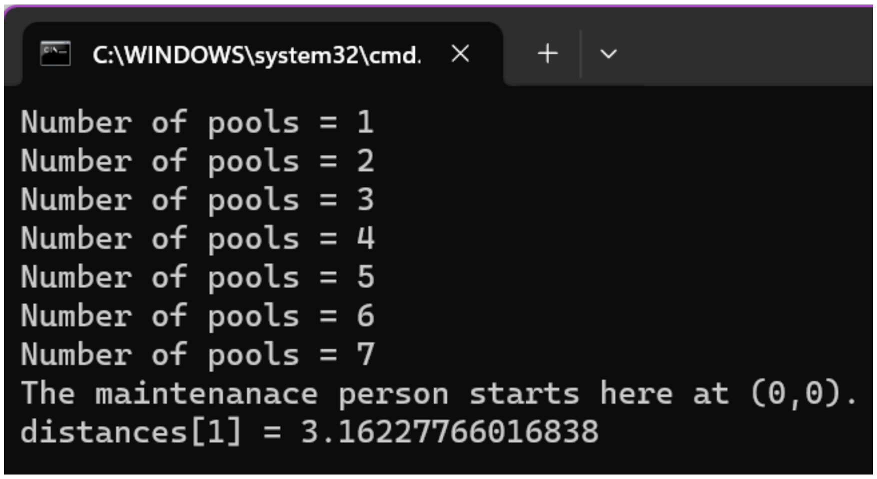
Final Project

UML

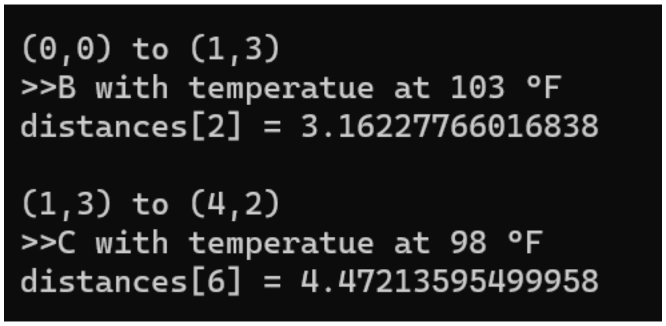


SCREENSHOTS



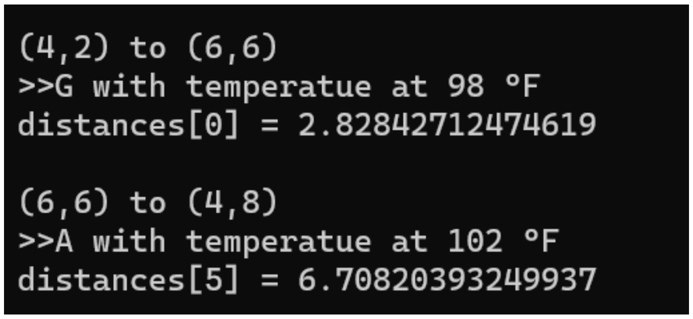
C:\WINDOWS\system32\cmd.

```
Number of pools = 1
Number of pools = 2
Number of pools = 3
Number of pools = 4
Number of pools = 5
Number of pools = 6
Number of pools = 7
The maintenanace person starts here at (0,0).
distances[1] = 3.16227766016838
```



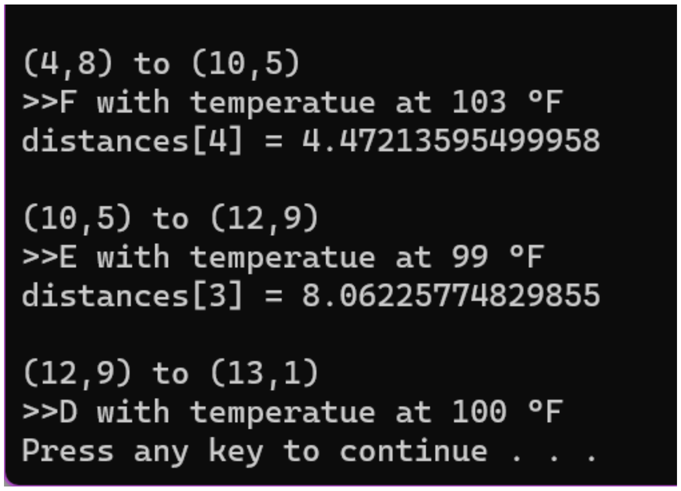
```
(0,0) to (1,3)
>>B with temperatue at 103 °F
distances[2] = 3.16227766016838

(1,3) to (4,2)
>>C with temperatue at 98 °F
distances[6] = 4.47213595499958
```



```
(4,2) to (6,6)
>>G with temperatue at 98 °F
distances[0] = 2.82842712474619

(6,6) to (4,8)
>>A with temperatue at 102 °F
distances[5] = 6.70820393249937
```



```
(4,8) to (10,5)
>>F with temperatue at 103 °F
distances[4] = 4.47213595499958

(10,5) to (12,9)
>>E with temperatue at 99 °F
distances[3] = 8.06225774829855

(12,9) to (13,1)
>>D with temperatue at 100 °F
Press any key to continue . . .
```

CODE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Final_Project
{
    class Pool
    {
        // data members
        private string name;
        private Location loc;
        private Temperature temp;
        public static int count = 0;

        public Pool()                                   // default constructor
        {
            name = "Unknown";
            loc = new Location();
            temp = new Temperature();
            count++;
            Console.WriteLine($"Number of pools = {count}");
        }
        public Pool(string n, Location l, Temperature t)        // overloading constructor
        {
            name = n;
            loc = l;
            temp = t;
            count++;
            Console.WriteLine($"Number of pools = {count}");
        }
        public Pool(string n, int x, int y, double d, string s)  // overloading constructor
        {
            name = n;
            loc = new Location(x, y);
            temp = new Temperature(d, s);
            count++;
            Console.WriteLine($"Number of pools = {count}");
        }
        public Pool(string n, int x, int y, string s)  // overloading constructor
        {
            name = n;
            loc = new Location(x, y);
            temp = new Temperature(s);
            count++;
            Console.WriteLine($"Number of pools = {count}");
        }
        public string Name                              // Name property
        {
            get { return name; }
            set { name = value; }
        }
        public Location Loc                             // Location property
        {
            get { return loc; }
            set { loc = value; }
        }
        public Temperature Temp                          // Temp property
        {
            get { return temp; }
            set { temp = value; }
        }
        public double findDistance(Pool p, int start_x, int start_y)
        {
            double distance = 0;
            double square_x = (p.Loc.X - start_x) * (p.Loc.X - start_x);
            double square_y = (p.Loc.Y - start_y) * (p.Loc.Y - start_y);

            distance = Math.Sqrt(square_x + square_y);
            return distance;
        }
        public override string ToString()
        {
            string strout = $"Pool's name = {name}\n";
            strout += Loc.ToString();
            strout += temp.ToString();

            return strout;
        }
        ~Pool() { }                                      // destructor
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace Final_Project
{
    class Temperature:Pool
    {
        // data members
        private double degree;
        private string scale;

        public Temperature()                             // default constructor
        {
            degree = 0;
            scale = "Unknown";
        }
        public Temperature(double d, string s)           // overloading constructor
        {
            degree = d;
            scale = s;
        }
        public Temperature(string s)
        {
            degree = 0;    // will be replaced by a random number for the degree
            scale = s;
        }
        public double Degree                             // Degree property
        {
            get { return degree; }
            set { degree = value; }
        }
        public string Scale                              // Scale property
        {
            get { return scale; }
            set { scale = value; }
        }
        public override string ToString()
        {
            string strout = $"Pool temperature = {degree} {scale}\n";
            return strout;
        }
        ~Temperature() { }                                // destructor
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Final_Project
{
    class Location:Pool
    {
        // data members
        private int x;
        private int y;

        public Location()                                 // default constructor
        {
            x = 0;
            y = 0;
        }
        public Location(int pos_x, int pos_y)            // overloading constructor
        {
            x = pos_x;
            y = pos_y;
        }
        public int X                                      // X position property
        {
            get { return x; }
            set { x = value; }
        }
        public int Y                                      // Y position property
        {
            get { return y; }
            set { y = value; }
        }
        public override string ToString()
        {
            string strout = $"Pool x position = {x}\nPool y position = {y}\n";
            return strout;
        }
        ~Location() { }                                   // destructor
    }
```

```
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Final_Project
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Random rnd = new Random();                  // Random class
            Pool[] pools = new Pool[7];                // 7 swimming pools within Pool class
            pools[0] = new Pool("A", 4, 8, "°F");
            pools[1] = new Pool("B", 1, 3, "°F");
            pools[2] = new Pool("C", 4, 2, "°F");
            pools[3] = new Pool("D", 13, 1, "°F");
            pools[4] = new Pool("E", 12, 9, "°F");
            pools[5] = new Pool("F", 10, 5, "°F");
            pools[6] = new Pool("G", 6, 6, "°F");

            // variables needed for the shortest distance calculations
            int start_x = 0;
            int start_y = 0;
            int loc = 0;
            double smallest = 0;

            // list of distances between each of the 7 pools and the starting location
            double[] distances = new double[7];
            Console.WriteLine($"The maintenanace person starts here at ({start_x},{start_y}).");

            // main for loop
            for (int i = 0; i < pools.Length; i++)
            {
                // finds the distances needed
                for (int j = 0; j < distances.Length; j++)
                {
                    distances[j] = pools[j].findDistance(pools[j], start_x, start_y);
                }

                // function gives the value and index of smallest distance
                SmallestNum(distances, ref loc, ref smallest);
                Console.Write($"\n({start_x},{start_y}) to ");

                // starting location is changed to the pool's location
                start_x = pools[loc].Loc.X;
                start_y = pools[loc].Loc.Y;
                Console.WriteLine($"({start_x},{start_y})");

                // the poot's location is set a large value
                pools[loc].Loc.X = 100;
                pools[loc].Loc.Y = 100;

                // a random temp between 94 and 104 is set to that pool
                pools[loc].Temp.Degree = rnd.Next(98, 104);
                Console.WriteLine($">>{pools[loc].Name} with temperatue at
{pools[loc].Temp.Degree} {pools[loc].Temp.Scale}");
            }
        }

        static void SmallestNum(double[] s, ref int loc, ref double small)
        {
            loc = Array.IndexOf(s, s.Min());
            small = s.Min();
            Console.WriteLine($"distances[{loc}] = {small}");
        }
    }
}
```