WORLDQUANT UNIVERSITY

MASTERS OF SCIENCE IN FINANCIAL ENGINEERING

DATA FEEDS AND TECHNOLOGY (C18-S3)

THOMAS DARLINGTON ADISENU

CEPHAS AKORMEDIE-TAY

DAVID KOFI GOGOVIE

DAVID KWASI NYONYO MENSAH-GBEKOR

GROUP WORK PROJECT – FIRST SUBMISSION

C# APPLICATION TO TRACK PROPERTY INFORMATION IN EXCEL

GROUP 2-A

2019

ABSTRACT

This project seeks to develop a C# console application to track property information in excel using Microsoft Visual Studio Integrated Development Environment. We further compute basic statistics on the property data such as mean price, variance, minimum and maximum price.

Keywords: C#, Excel, Visual Studio, Basic Statistics.

Data Selection:

No data provided and due to lack of real property price data, we randomly chose some arbitrary figures and used in the console application.

Statistical Analysis:

As shown in the graphs in the appendix below, the mean, variance, minimum and maximum value of property price for the randomly chosen data was computed with average price being 400, variance 4,600, minimum prince of 320 and maximum price of 500 for house properties located in the cities of predominantly Tema and then Accra.

Conclusion:

C# is object oriented and is well integrated with excel as both were developed by Microsoft which provides a library for easy interoperability and since excel is widely used in industry, it warrants the use of programming languages like C# in addition to VB and Macros to automate some processes and we can imagine how useful and powerful this easy integration can provide to solve very huge complex problems.

References

value, H., Down, S. and Sopko, D. (2020). *How to read single Excel cell value*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/18993735/how-to-read-single-excel-cell-value [Accessed 16 Jan. 2020].

Coderslexicon.com. (2020). *Variance and Standard Deviation of An Array in C# : The Coders Lexicon*. [online] Available at: https://www.coderslexicon.com/variance-and-standard-deviation-of-an-array-in-c/ [Accessed 16 Jan. 2020].

Support.office.com. (2020). *Excel functions (by category)*. [online] Available at: https://support.office.com/en-us/article/excel-functions-by-category-5f91f4e9-7b42-46d2-9bd1-63f26a86c0eb [Accessed 16 Jan. 2020].

Docs.microsoft.com. (2020). *Microsoft.Office.Interop.Excel Namespace*. [online] Available at: https://docs.microsoft.com/en-us/dotnet/api/microsoft.office.interop.excel?view=excel-pia [Accessed 16 Jan. 2020].

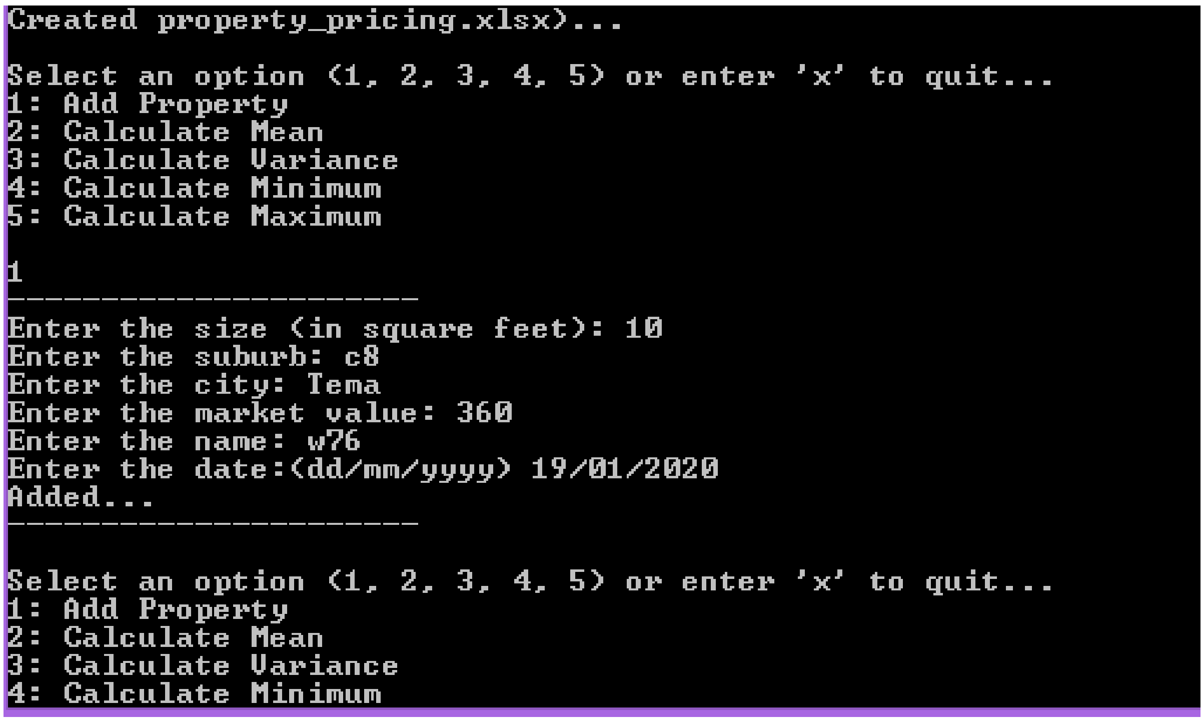
Appendix

Project Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Excel = Microsoft.Office.Interop.Excel;

namespace GWP2A_Properties
{
    class Program
    {
        static Excel.Workbook workbook;
        static Excel.Application app;

        static void Main(string[] args)
        {
            app = new Excel.Application();
            app.Visible = true;
            try
            {
                workbook = app.Workbooks.Open("property_pricing.xlsx", ReadOnly:
false);
            }
            catch
            {
                SetUp(app);
            }

            var input = "";
            while (input != "x")
            {
                PrintMenu();
                input = Console.ReadLine();
                try
                {
                    Console.WriteLine("----------------------");
                    var option = int.Parse(input);
                    switch (option)
                    {
                        case 1:
                            try
                            {
                                Console.Write("Enter the size: ");
                                var size = float.Parse(Console.ReadLine());
                                Console.Write("Enter the suburb: ");
                                var suburb = Console.ReadLine();
                                Console.Write("Enter the city: ");
                                var city = Console.ReadLine();
                                Console.Write("Enter the market value: ");
                                var value = float.Parse(Console.ReadLine());
                                Console.Write("Enter the name: ");
                                var name = Console.ReadLine();
                                Console.Write("Enter the date:(dd/mm/yyyy) ");
                                var date = Console.ReadLine();

                                AddPropertyToWorksheet(size, suburb, city, value,
name, date);
                                Console.WriteLine("Added...");
                            }
                            catch
                            {
                                Console.WriteLine("Error: couldn't parse input");
                            }
                            break;
                        case 2:
                            Console.WriteLine("Mean price: " + CalculateMean());
                            break;
                        case 3:
                            Console.WriteLine("Price variance: " +
CalculateVariance());
                            break;
                        case 4:
                            Console.WriteLine("Minimum price: " + CalculateMinimum());
                            break;
                        case 5:
                            Console.WriteLine("Maximum price: " + CalculateMaximum());
                            break;
                        default:
                            break;
                    }
                    Console.WriteLine("----------------------");
                } catch { }
            }

            // save before exiting
            try
            {
                workbook.Save();
                workbook.Close();
            }
            catch { }
            app.Quit();
        }

        static void PrintMenu()
        {
            Console.WriteLine();
            Console.WriteLine("Select an option (1, 2, 3, 4, 5) " +
                              "or enter 'x' to quit...");
            Console.WriteLine("1: Add Property");
            Console.WriteLine("2: Calculate Mean");
            Console.WriteLine("3: Calculate Variance");
            Console.WriteLine("4: Calculate Minimum");
            Console.WriteLine("5: Calculate Maximum");
            Console.WriteLine();
        }

        static void SetUp(Excel.Application app)
        {
            app.Workbooks.Add();
            Excel.Workbook workbook;
            workbook = app.ActiveWorkbook;
            workbook.Worksheets.Add();

            Excel.Worksheet currentSheet = workbook.Worksheets[1];
            currentSheet.Name = "Properties";
            currentSheet.Cells[1, "A"] = "Size (in square feet)";
            currentSheet.Cells[1, "B"] = "Suburb";
            currentSheet.Cells[1, "C"] = "City";
            currentSheet.Cells[1, "D"] = "Market Value";
            currentSheet.Cells[1, "E"] = "Name";
            currentSheet.Cells[1, "F"] = "Date";
            currentSheet.Cells[1, "G"] = "Counter";
            currentSheet.Cells[1, "H"] = 0;

            workbook.SaveAs("property_pricing.xlsx");
            Console.WriteLine("Created property_pricing.xlsx)...");
        }

        static void AddPropertyToWorksheet(float size, string suburb, string city,
float value, string name, string date)
        {
            int row;
            Excel.Workbook workbook;
            workbook = app.ActiveWorkbook;
            Excel.Worksheet currentSheet = workbook.Worksheets[1];
            row = currentSheet.UsedRange.Rows.Count + 1;// look for first empty row
            currentSheet.Cells[row, "A"] = size;
            currentSheet.Cells[row, "B"] = suburb;
            currentSheet.Cells[row, "C"] = city;
            currentSheet.Cells[row, "D"] = value;
            currentSheet.Cells[row, "E"] = name;
            currentSheet.Cells[row, "F"] = date;
            currentSheet.Cells[row, "G"] = (int)(currentSheet.Range["H1", "H1"].Value)
+ 1;
            currentSheet.Cells[1, "H"] = (int)(currentSheet.Range["H1", "H1"].Value) +
1;
            return;
        }

        static float CalculateMean()
        {
            float result = 0.0f;
            int row;
            Excel.Workbook workbook;
            workbook = app.ActiveWorkbook;
            Excel.Worksheet currentSheet = workbook.Worksheets[1];
            row = currentSheet.UsedRange.Rows.Count;

            if (row > 1) result =
(float)app.WorksheetFunction.Average(currentSheet.UsedRange.Range["D2", "D" + row]);
            return result;
        }

        static float CalculateVariance()
        {
            float result = 0.0f;
            int row;
            Excel.Workbook workbook;
            workbook = app.ActiveWorkbook;
            Excel.Worksheet currentSheet = workbook.Worksheets[1];
            row = currentSheet.UsedRange.Rows.Count;

            if (row > 1) result =
(float)app.WorksheetFunction.Var(currentSheet.UsedRange.Range["D2", "D" + row]);
            return result;
        }

        static float CalculateMinimum()
        {
            float result = 0.0f;
            int row;
            Excel.Workbook workbook;
            workbook = app.ActiveWorkbook;
            Excel.Worksheet currentSheet = workbook.Worksheets[1];
            row = currentSheet.UsedRange.Rows.Count;

            if (row > 1) result =
(float)app.WorksheetFunction.Min(currentSheet.UsedRange.Range["D2", "D" + row]);
            return result;
        }

        static float CalculateMaximum()
        {
            float result = 0.0f;
            int row;
            Excel.Workbook workbook;
            workbook = app.ActiveWorkbook;
            Excel.Worksheet currentSheet = workbook.Worksheets[1];
            row = currentSheet.UsedRange.Rows.Count;

            if (row > 1) result =
(float)app.WorksheetFunction.Max(currentSheet.UsedRange.Range["D2", "D" + row]);
            return result;
        }
    }
}

//------------------------------------------------------------------------END OF CODE---------------
```

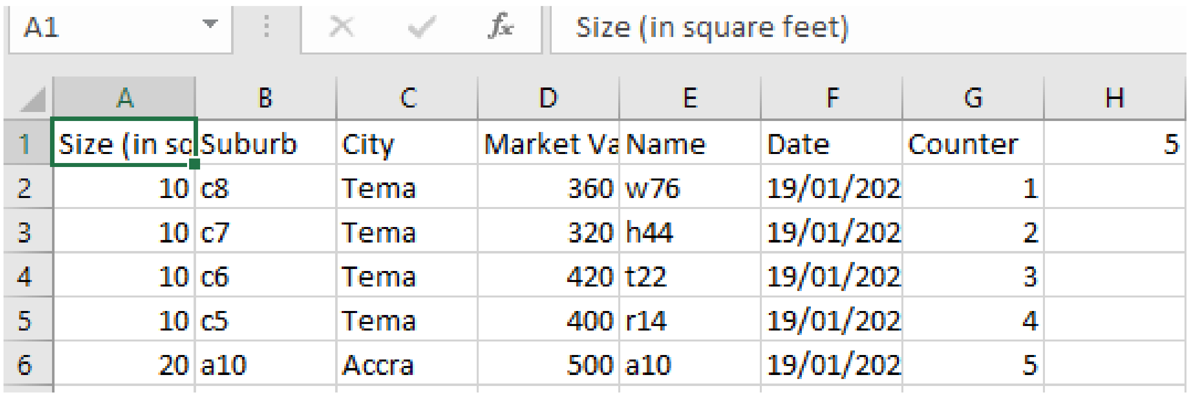
Project Graphics and Outputs (Using Arbitrary Sample Data)

Figure 1 – Command Line Interface



```
Created property_pricing.xlsx)...

Select an option (1, 2, 3, 4, 5) or enter 'x' to quit...
1: Add Property
2: Calculate Mean
3: Calculate Variance
4: Calculate Minimum
5: Calculate Maximum

1
----------------------
Enter the size (in square feet): 10
Enter the suburb: c8
Enter the city: Tema
Enter the market value: 360
Enter the name: w76
Enter the date:(dd/mm/yyyy) 19/01/2020
Added...
----------------------

Select an option (1, 2, 3, 4, 5) or enter 'x' to quit...
1: Add Property
2: Calculate Mean
3: Calculate Variance
4: Calculate Minimum
```

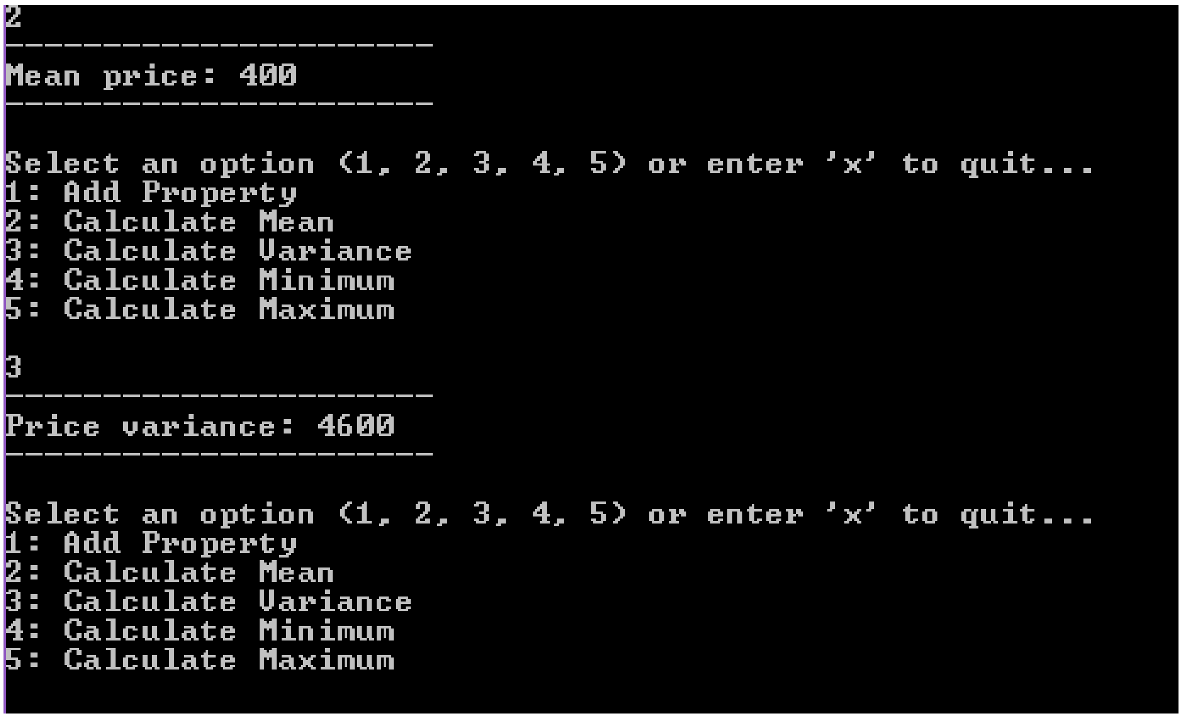
Figure 2 – Screenshot of Excel Sample Data



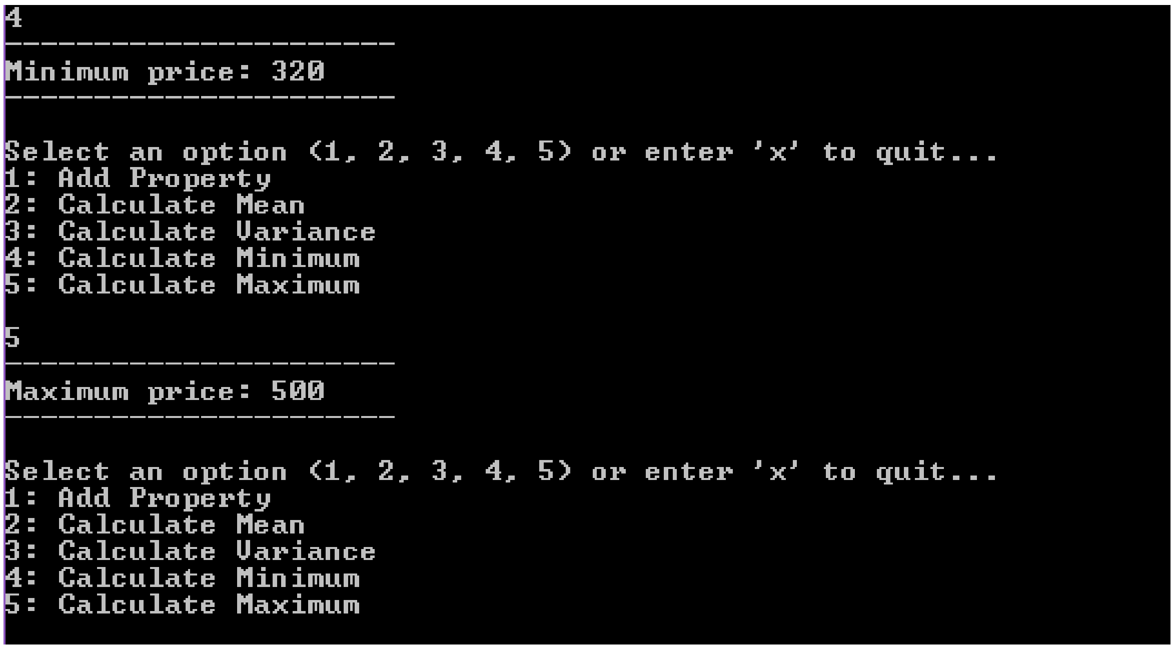
A1 | Size (in square feet)

	A	B	C	D	E	F	G	H
1	Size (in sc	Suburb	City	Market Va	Name	Date	Counter	5
2	10	c8	Tema	360	w76	19/01/202	1	
3	10	c7	Tema	320	h44	19/01/202	2	
4	10	c6	Tema	420	t22	19/01/202	3	
5	10	c5	Tema	400	r14	19/01/202	4	
6	20	a10	Accra	500	a10	19/01/202	5	

Figure 3 – Statistical Results



```
2
----------------------
Mean price: 400
----------------------

Select an option (1, 2, 3, 4, 5) or enter 'x' to quit...
1: Add Property
2: Calculate Mean
3: Calculate Variance
4: Calculate Minimum
5: Calculate Maximum

3
----------------------
Price variance: 4600
----------------------

Select an option (1, 2, 3, 4, 5) or enter 'x' to quit...
1: Add Property
2: Calculate Mean
3: Calculate Variance
4: Calculate Minimum
5: Calculate Maximum
```



```
4
-----------------------
Minimum price: 320
-----------------------

Select an option (1, 2, 3, 4, 5) or enter 'x' to quit...
1: Add Property
2: Calculate Mean
3: Calculate Variance
4: Calculate Minimum
5: Calculate Maximum

5
-----------------------
Maximum price: 500
-----------------------

Select an option (1, 2, 3, 4, 5) or enter 'x' to quit...
1: Add Property
2: Calculate Mean
3: Calculate Variance
4: Calculate Minimum
5: Calculate Maximum
```