



T.C.

ISTANBUL AREL UNIVERSITY

FACULTY OF ENGINEERING & ARCHITECTURE

| | | |
|-------------------|-----------------------------------|---|
| NAME | : İbrahim Başar | COURSE: Visual Based Programming |
| SURNAME | : YARGICI | Term: 2020-2021 Spring |
| STUDENT ID | : 180303006 | Lecturer: Metin Zontul |
| DEPARTMENT | : Computer Engineering (2) | DEADLINE: 07.06.2021 |

Midterm-2 Assignment

That document contains the solutions, pseudocodes and flowcharts of **8.8, 12.6, 13.8 and 17.8** questions, from “Deitel C Sharp How to Program” book.

I uploaded my works to GitHub as a public repo. After you say okay, now you can make it private, I will. You can visit and see my Visual Based Programming repo from:

<https://github.com/basarYargici/VisualBasedProgrammingMidterm2>

Format of solution papers:

1. Pseudocode
2. Flowchart
3. Source Code
4. Output of program

Question 8.8:

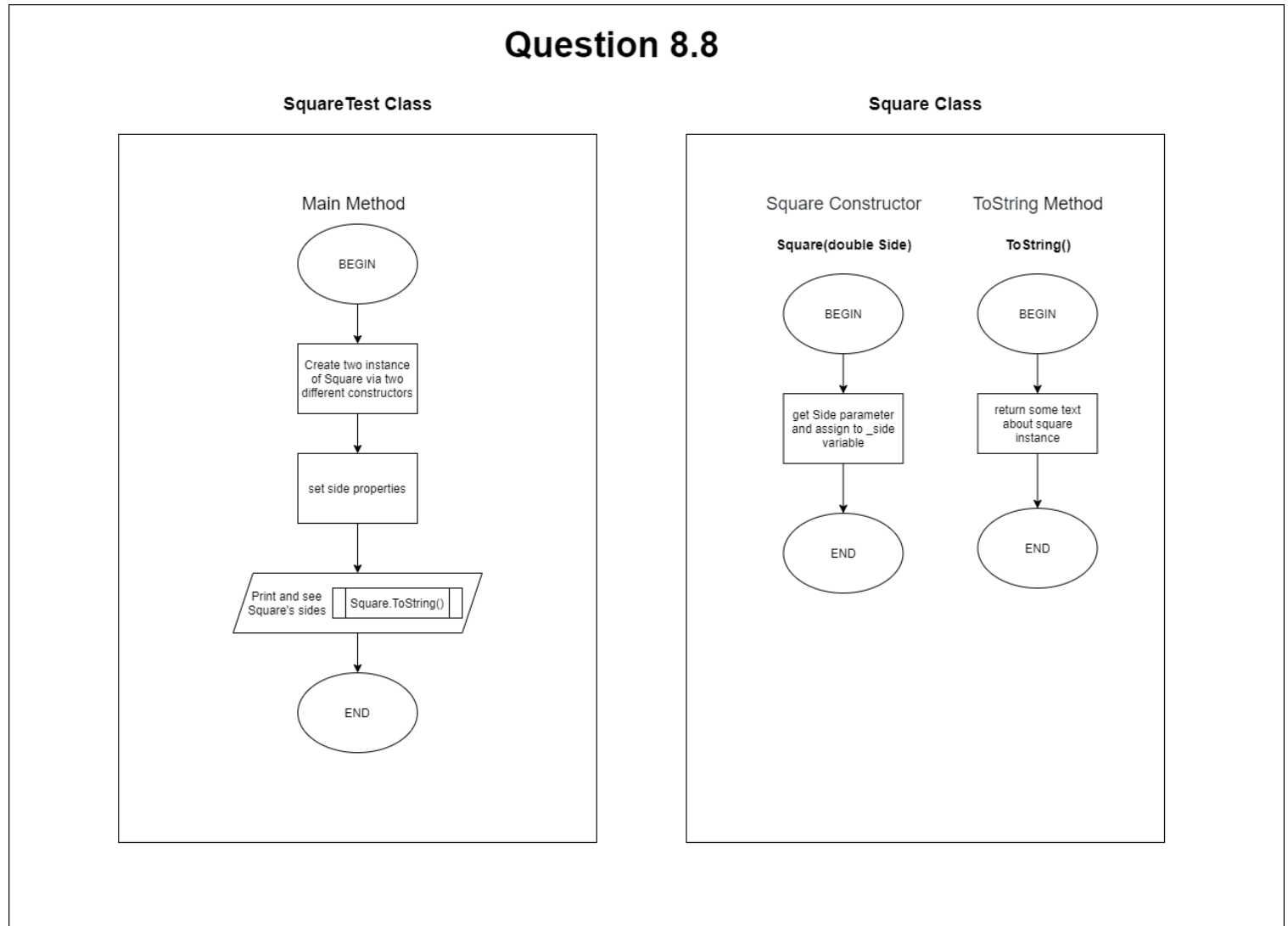
Write a console application that implements a Square shape. Class Square should contain an instance property Side that has get and set accessors for private data. Provide two constructors: one that takes no arguments and another that takes a side length as a value. Write an application class that tests class Square's functionality.

Solution 8.8:

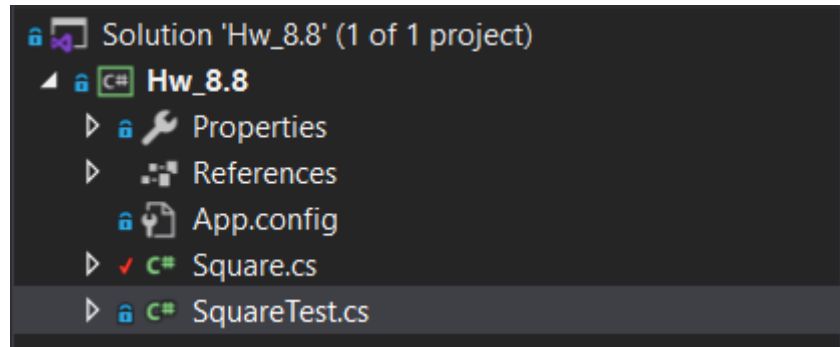
PSEUDOCODE:

- There should be a class named Square
- Square class should have side attribute which will be private and will be accessed via getters and setters
- Square class will have two constructors, one will have side parameter
- SquareTest class will test square instances.
- One instances' side property will be given via parameterized constructor, others via setter
- Both sides will be printed

FLOWCHART:



CODE:



```
1 namespace HW_8._8
2 {
3     6 references
4     class Square
5     {
6         private double _side;
7
8         2 references
9         public double Side { get => _side; set => _side = value; }
10
11         1 reference
12         public Square()
13         {
14         }
15
16         1 reference
17         public Square(double Side)
18         {
19             this.Side = Side;
20         }
21
22         2 references
23         public override string ToString()
24         {
25             return "Square side = " + _side;
26         }
27     }
28 }
```

```
1 using System;
2
3 namespace HW_8._8
4 {
5     class SquareTest
6     {
7         static void Main(string[] args)
8         {
9             // test square class without argument
10            Square firstSquare = new Square();
11            firstSquare.Side = 3.5;
12            Console.WriteLine(firstSquare.ToString());
13
14            // test square class side argument
15            Square secondSquare = new Square(34);
16            Console.WriteLine(secondSquare.ToString());
17
18            // to stop console disappearing in debug mode
19            Console.WriteLine("Press any key to continue . . .");
20            Console.ReadKey();
21        }
22    }
23 }
24 }
```

Outputs of program:

C:\Users\YARGICI\Desktop\Ceng2\Visual2\Hw_8.8\Hw_8.8\bin\Debug\Hw_8.8.exe

```
Square side = 3,5
Square side = 34
Press any key to continue . . .
```

Question 12.6:

Extend the program of Fig. 12.30 to include options for changing the size and color of the lines drawn. Create a GUI similar to the one in Fig. 12.35. [Hint: Have variables to keep track of the currently selected size (int) and color (Color object). Set them using the event handlers for the radio buttons. For the color, use the various Color constants (such as Color.Blue). When responding to the mouse moves, simply use the size and color variables to determine the proper size and color.]

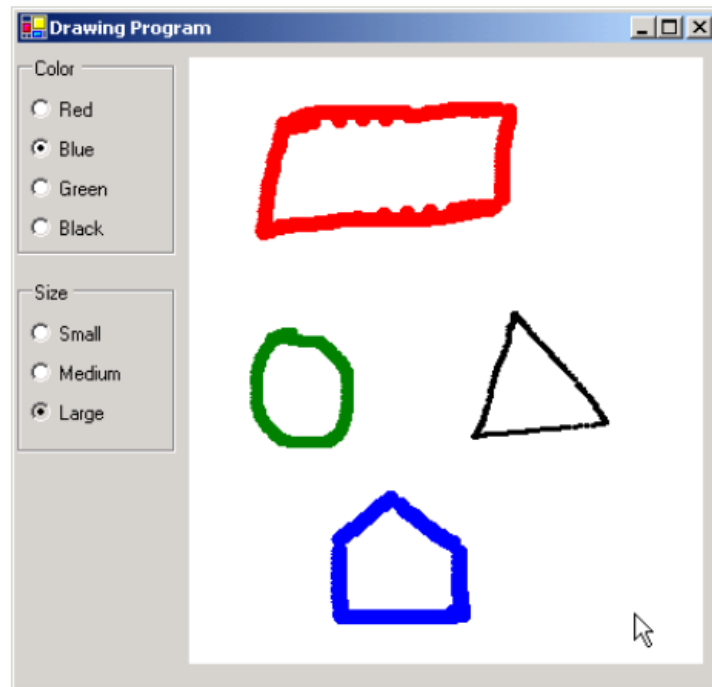


Fig. 12.35 GUI for Exercise 12.6.

Solution 12.6:

PSEUDOCODE:

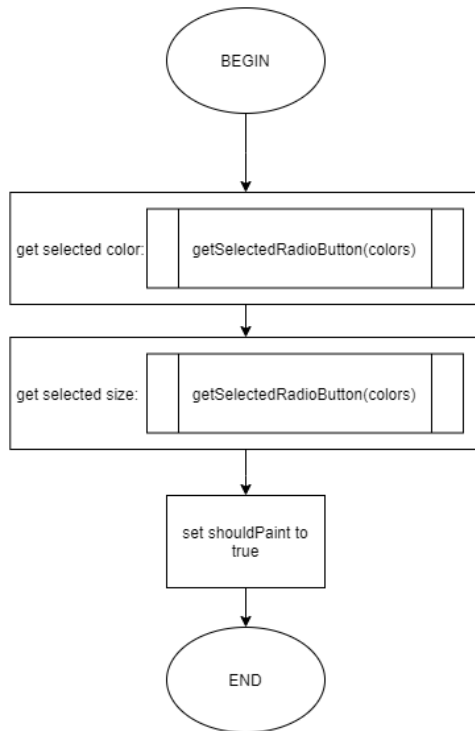
- There should be two GroupBoxes which will contains colors and sizes as RadioButtons
- On the right side, there should be drawable area which we will draw our shapes
- Program should take the selected size and color RadioButtons and later it should adjust the paint brush

Question 12.6

Painter Class

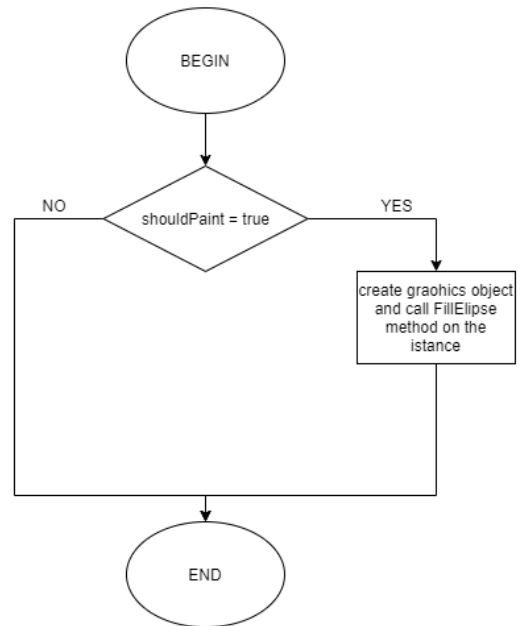
Mouse Down Method

Painter_MouseDown(object sender, MouseEventArgs e)



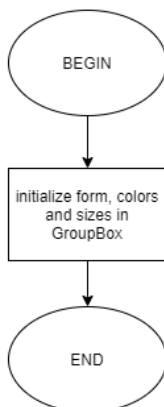
Mouse Move Method

Painter_MouseMove(object sender, MouseEventArgs e)



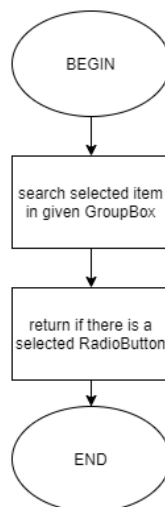
Painter Constructor

Painter()



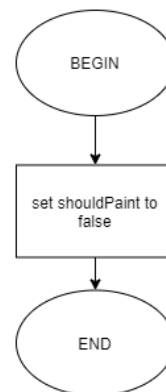
Get Selected RadioButton Method

getSelectedRadioButton(colors)

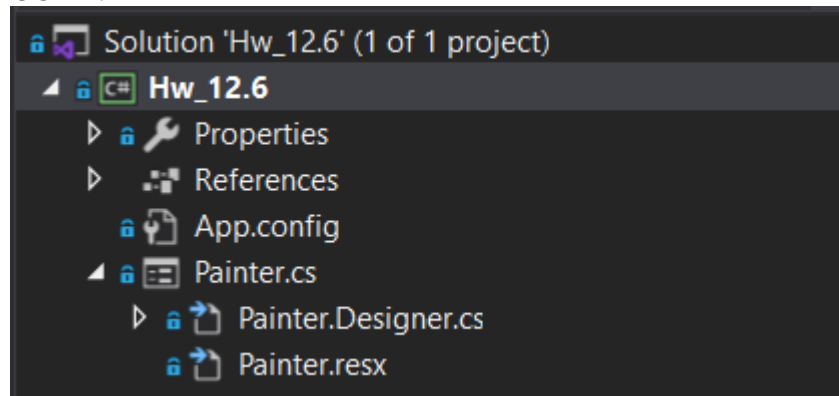


Mouse Up Method

Painter_MouseDown(object sender, MouseEventArgs e)



CODE:



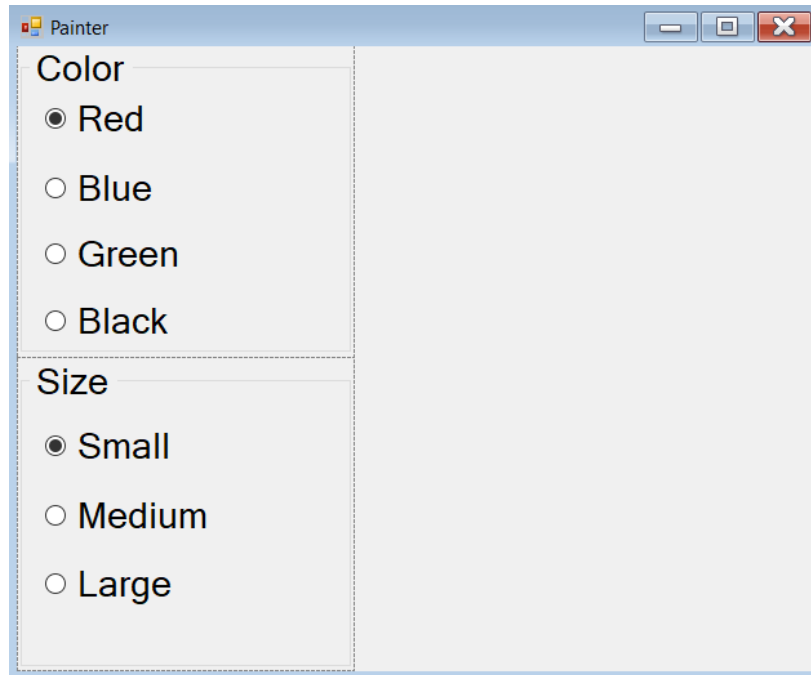
```
1 using System;
2 using System.Collections.Generic;
3 using System.Drawing;
4 using System.Linq;
5 using System.Windows.Forms;
6
7 namespace Hw_12._6
8 {
9     public partial class Painter : Form
10     {
11         bool shouldPaint = false; // whether to paint
12         GroupBox colors;
13         GroupBox sizes;
14         Dictionary<String, Color> colorTypes = new Dictionary<String, Color>()
15         {
16             { "Red", Color.Red },
17             { "Blue", Color.Blue },
18             { "Green", Color.Green },
19             { "Black", Color.Black }
20         };
21         Dictionary<String, int> sizeTypes = new Dictionary<String, int>()
22         {
23             { "Small", 4 },
24             { "Medium", 8 },
25             { "Large", 12 }
26         };
27
28         Graphics graphics;
29         Color selectedColor;
30         int selectedSize;
```

```

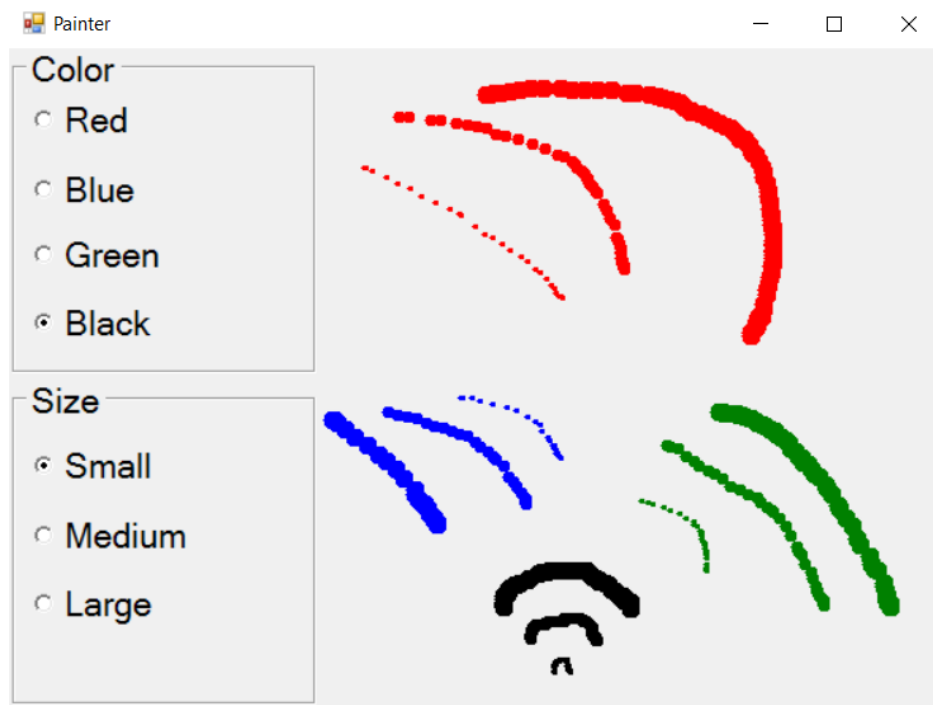
31
32     /// creates a form as a drawing surface
33     1 reference
34     public Painter()
35     {
36         InitializeComponent();
37         colors = gbColors;
38         sizes = gbSizes;
39     }
40
41     /// The main entry point for the application.
42     [STAThread]
43
44     static void Main()
45     {
46         Application.Run(new Painter());
47     }
48
49     /// should paint after mouse button has been pressed
50
51     private void Painter_MouseDown(object sender, MouseEventArgs e)
52     {
53         RadioButton checkedColor = getSelectedRadioButton(colors);
54         RadioButton checkedSize = getSelectedRadioButton(sizes);
55         selectedColor = colorTypes[checkedColor.Text];
56         selectedSize = sizeTypes[checkedSize.Text];
57         shouldPaint = true;
58     }
59
60
61     2 references
62     private RadioButton getSelectedRadioButton(GroupBox groupBox)
63     {
64         foreach (RadioButton rdo in groupBox.Controls.OfType<RadioButton>())
65         {
66             if (rdo.Checked)
67             {
68                 return rdo;
69             }
70         }
71         throw new Exception("Radio button should be selected");
72     }
73
74     /// stop painting when mouse button released
75     1 reference
76     private void Painter_MouseUp(object sender, MouseEventArgs e)
77     {
78         shouldPaint = false;
79     }
80
81     /// draw circle whenever mouse button moves (and mouse is down)
82     1 reference
83     protected void Painter_MouseMove(object sender, MouseEventArgs e)
84     {
85         if (shouldPaint)
86         {
87             graphics = CreateGraphics();
88             graphics.FillEllipse(new SolidBrush(selectedColor), e.X, e.Y, selectedSize, selectedSize);
89         }
90     } // end Painter_MouseMove
91
92 } // end class Painter

```

Design:



Outputs of program:



Color

- ☐ Red
- ☐ Blue
- ☐ Green
- ☒ Black

Size

- ☒ Small
- ☐ Medium
- ☐ Large



Question 13.8:

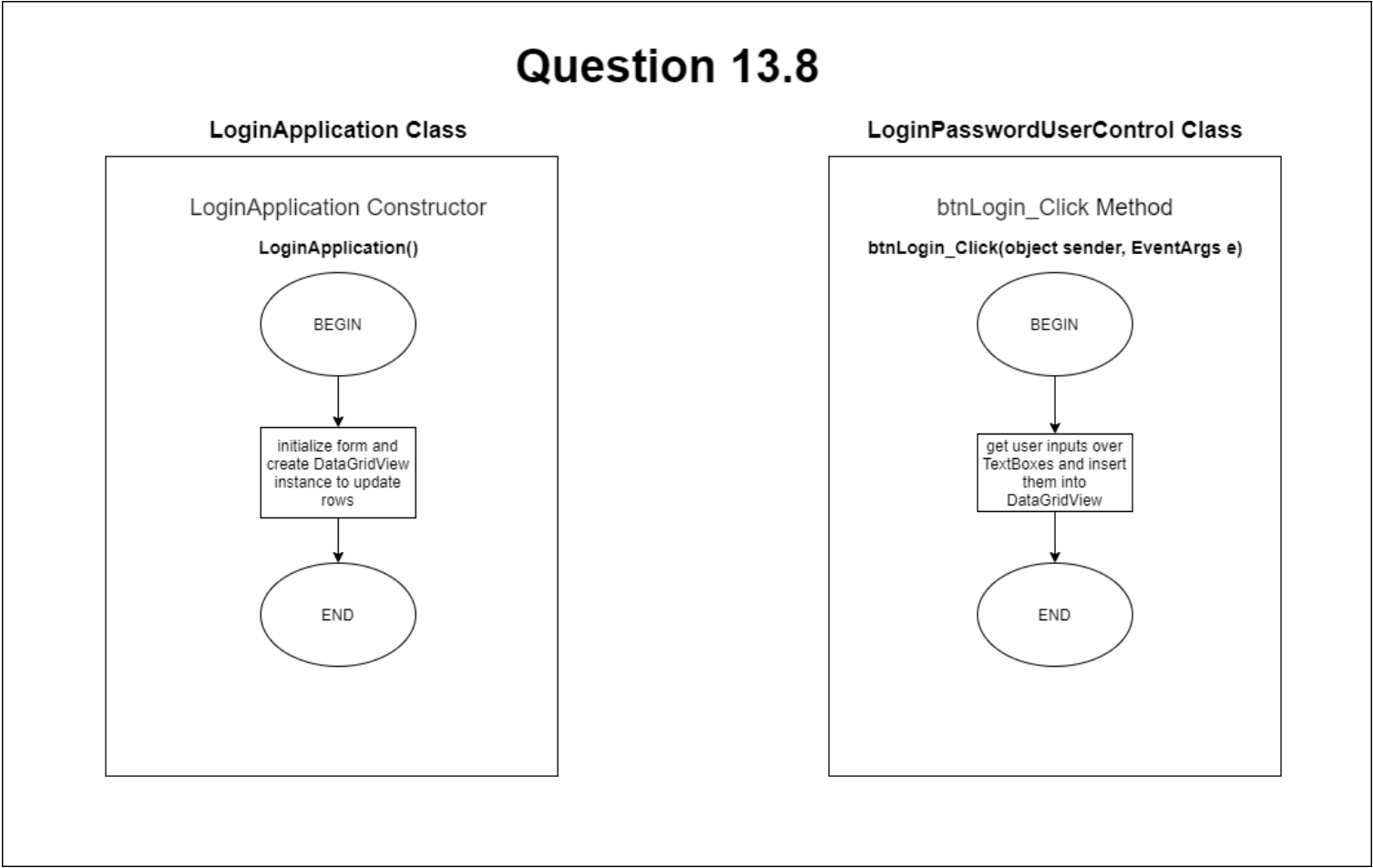
Create a UserControl called LoginPasswordUserControl. The LoginPasswordUserControl contains a Label (loginLabel) that displays String "Login:", a TextBox (loginTextBox) where the user inputs a login name, a Label (passwordLabel) that displays the String "Password:" and finally, a TextBox (passwordTextBox) where a user inputs a password (don't forget to set property PasswordChar to "*" in the TextBox's Properties window). LoginPasswordUserControl must provide public read-only properties Login and Password that allow an application to retrieve the user input from loginTextBox and passwordTextBox. The UserControl must be exported to an application that displays the values input by the user in LoginPasswordUserControl.

Solution 13.8:

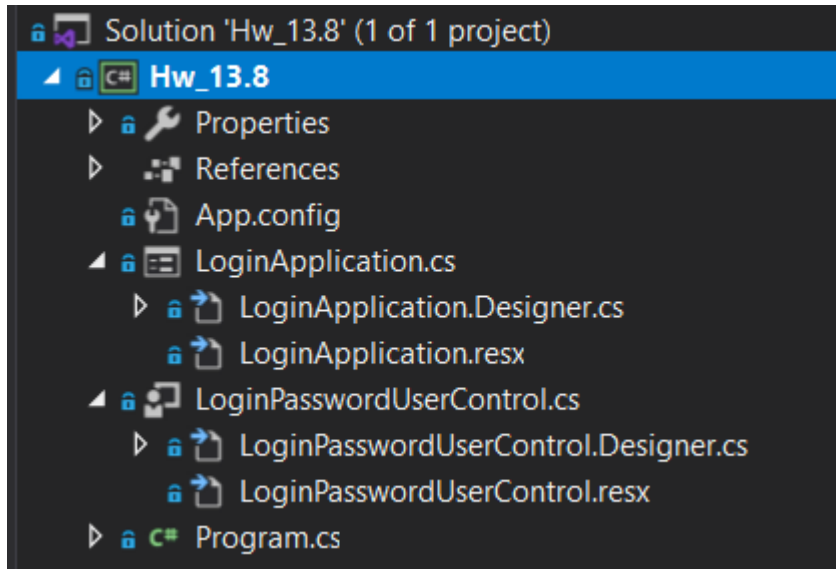
PSEUDOCODE:

- There should be a UserControl which will contain login and password Labels and TextBoxes
- There can be a DataGridView to show login and password fields
- When user enters input, it should be hidden as '*'
- Later on, created UserControl should be implemented on a Form Application

FLOWCHART:



CODE:



```
1 using System;
2 using System.Windows.Forms;
3
4 namespace Hw_13._8
5 {
6     4 references
7     public partial class LoginPasswordUserControl : UserControl
8     {
9         public String Login = "";
10        public String Password = "";
11        1 reference
12        public LoginPasswordUserControl()
13        {
14            InitializeComponent();
15        }
16        1 reference
17        private void btnLogin_Click(object sender, EventArgs e)
18        {
19            // taking the input of TextBoxes and assigning them in a variable
20            Login = loginTextBox.Text.ToString();
21            Password = passwordTextBox.Text.ToString();
22
23            // to show user inputs in DataGridView we are adding those datas
24            LoginApplication.dataGridView.Rows.Add(Login, Password);
25        }
26    }
27 }
```

```
1 using System.Windows.Forms;
2
3 namespace Hw_13._8
4 {
5     4 references
6     public partial class LoginApplication : Form
7     {
8         // there will be only one DataGridView. Making it static will improve performance.
9         public static DataGridView dataGridView;
10        1 reference
11        public LoginApplication()
12        {
13            InitializeComponent();
14            dataGridView = dataGridView;
15        }
16    }
17 }
```

Design:

| | |
|-----------|--------------------------|
| Login: | <input type="text"/> |
| | |
| Password: | <input type="password"/> |
| | |
| | LOGIN |

LoginApplication—□×

Login:

Password:

LOGIN

| | Login | Password |
|---|-------|----------|
| * | | |
| | | |

LoginApplication—□×

Login:

Password:

LOGIN

| | Login | Password |
|---|-------|----------|
| * | | |
| | | |

Outputs of program:

LoginApplication

Login:

let's

Password:

LOGIN

| | Login | Password |
|----|---------------|----------|
| | ibrahim başar | YARGICI |
| | Login | password |
| | try | it |
| | let's | do it |
| ▶▶ | | |

LoginApplication

Login:

let's

Password:

LOGIN

| | Login | Password |
|----|---------------|----------|
| | ibrahim başar | YARGICI |
| | Login | password |
| | try | it |
| | let's | do it |
| ▶▶ | | |

Question 17.8:

You are the owner of a hardware store and need to keep an inventory of the different tools you sell, how many of each are currently in stock and the cost of each. Write a program that initializes the random-access file hardware.dat to 100 empty records, lets you input data relating to each tool, enables you to list all your tools, lets you delete a record for a tool that you no longer have and lets you update any information in the file. The tool identification number should be the record number. Use the information in Fig. 17.24 to start your file.

| Record # | Tool name | Quantity | Price |
|----------|-----------------|----------|-------|
| 3 | Electric sander | 18 | 35.99 |
| 19 | Hammer | 128 | 10.00 |
| 26 | Jig saw | 16 | 14.25 |
| 39 | Lawn mower | 10 | 79.50 |
| 56 | Power saw | 8 | 89.99 |
| 76 | Screwdriver | 236 | 4.99 |
| 81 | Sledge hammer | 32 | 19.75 |
| 88 | Wrench | 65 | 6.48 |

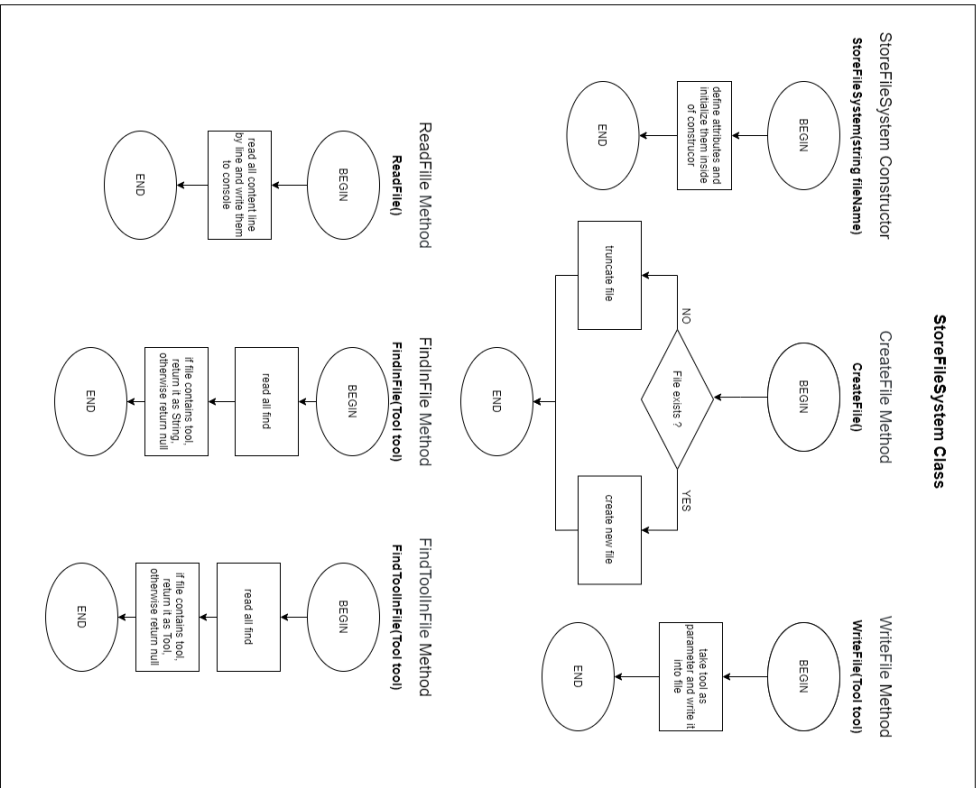
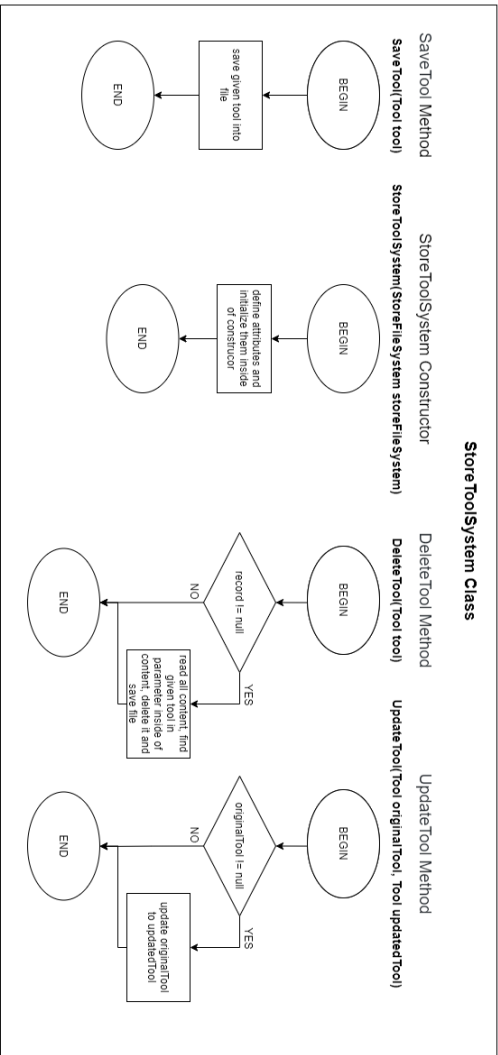
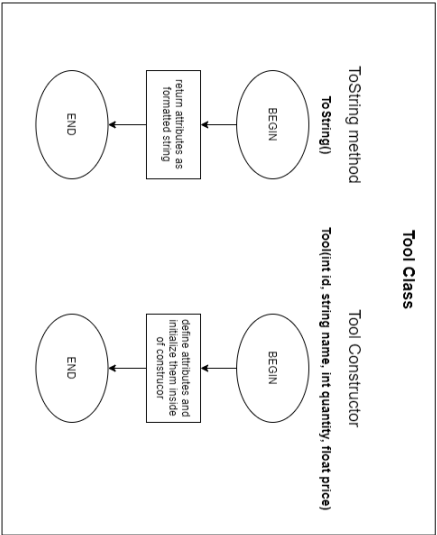
Fig. 17.24 Inventory of a hardware store.

Solution 17.8:

PSEUDOCODE:

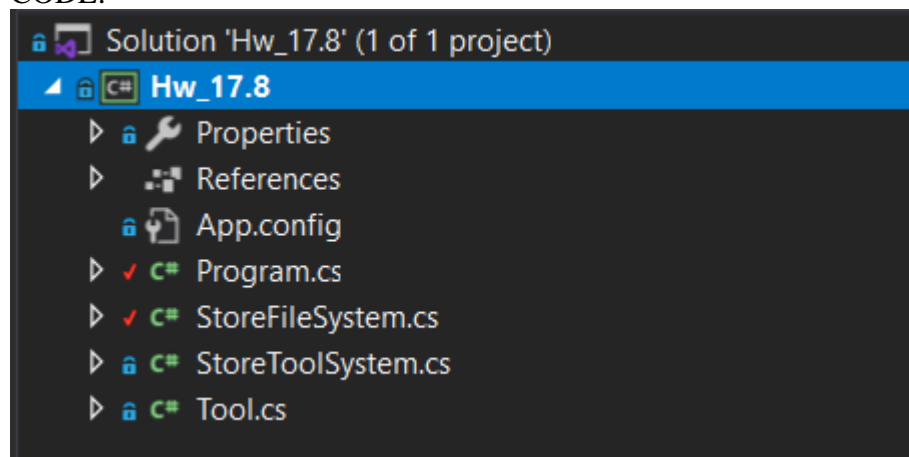
- There should be a Tool entity class which represents hardware tool in our program.
- There can be two different system classes: StoreFileSystem, which is responsible of file processes, and StoreToolSystem, which is responsible of adding, updating and deleting product from database.
- Program lets user input data relating to each tool, enables user to list all your tools, lets user delete a record for a tool that user no longer have and lets user update any information in the file.
- The tool identification number should be the record number.

Question 17.8



FLOWCHART:

CODE:



```
1  using System.Threading;
2
3  namespace Hw_17_8
4  {
5      /// <summary>
6      /// This class is an entity class which describes Tool objects
7      /// </summary>
8      class Tool
9      {
10         private int id = 0;
11         private string name;
12         private int quantity;
13         private float price;
14         public static int globalID;
15
16         4 references
17         public float Price
18         {
19             get { return price; }
20             set { price = value; }
21         }
22
23         4 references
24         public int Quantity
25         {
26             get { return quantity; }
27             set { quantity = value; }
28         }
29
30         4 references
31         public string Name
32         {
33             get { return name; }
34             set { name = value; }
35         }
36
37         5 references
38         public int RecordId
39         {
40             get { return id; }
41             set { id = value; }
42         }
43
44         1 reference
45         public Tool(int id, string name, int quantity, float price)
46         {
47             this.id = id;
48             this.name = name;
49             this.quantity = quantity;
50             this.price = price;
51         }
52
53         10 references
54         public Tool(string name, int quantity, float price)
55         {
56             id = Interlocked.Increment(ref globalID);
57             this.name = name;
58             this.quantity = quantity;
59             this.price = price;
60         }
61
62         /// <summary>
63         /// method formatted as written in hardware.dat file
64         /// </summary>
65         /// <returns></returns>
66         1 reference
67         public override string ToString()
68         {
69             return string.Format("{0,-10} {1,-15} {2,-10} {3,-10}", RecordId, Name, Quantity, Price);
70         }
71     }
72 }
```



```

1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5
6  namespace Hw_17._8
7  {
8      3 references
9      class StoreToolSystem
10     {
11         StoreFileSystem storeFileSystem;
12
13         /// <summary>
14         ///     Constructor for initializing StoreToolSystem
15         /// </summary>
16         /// <param name="storeFileSystem"> to access file processes</param>
17         1 reference
18         public StoreToolSystem(StoreFileSystem storeFileSystem)
19         {
20             this.storeFileSystem = storeFileSystem;
21         }
22
23         /// <summary>
24         ///     Saves given tool into hardware.dat file via storeFileSystem instance
25         /// </summary>
26         /// <param name="tool"> is a Tool object that will be saved </param>
27         2 references
28         public void SaveTool(Tool tool)
29         {
30             if (tool is null)
31             {
32                 throw new ArgumentNullException(nameof(tool));
33             }
34
35             storeFileSystem.WriteFile(tool);
36         }
37
38         /// <summary>
39         ///     Saves given list of tools into hardware.dat file via storeFileSystem instance
40         /// </summary>
41         /// <param name="toolList"> is a list of Tool objects that will be saved </param>
42         1 reference
43         public void SaveTool(List<Tool> toolList)
44         {
45             toolList.ForEach(l => SaveTool(l));
46         }
47
48         /// <summary>
49         ///     Deletes given tool from hardware.dat file via storeFileSystem instance
50         /// </summary>
51         /// <param name="tool"> is a Tool object that will be deleted </param>
52         1 reference
53         public void DeleteTool(Tool tool)
54         {
55             if (tool is null)
56             {
57                 throw new ArgumentNullException(nameof(tool));
58             }
59
60             String record = storeFileSystem.FindInFile(tool);
61
62             // create temporary file, make the changes and restore main file
63             if (record != null)
64             {
65                 var tempFile = Path.GetTempFileName();
66                 var linesToKeep = File.ReadLines(storeFileSystem.FileName).Where(l => l.Substring(0, 5) != record.Substring(0, 5));
67                 File.WriteAllLines(tempFile, linesToKeep);
68                 File.Delete(storeFileSystem.FileName);
69                 File.Move(tempFile, storeFileSystem.FileName);
70             }
71         }
72     }
73 }

```

```

70     /// <summary>
71     ///     Updates given originalTool to updatedTool via storeFileSystem instance
72     /// </summary>
73     /// <param name="originalTool"> original instance </param>
74     /// <param name="updatedTool"> the form of updated instance </param>
75     1 reference
76     public void UpdateTool(Tool originalTool, Tool updatedTool)
77     {
78         Tool t = storeFileSystem.FindToolInFile(originalTool);
79
80         if (t != null)
81         {
82             t.Name = updatedTool.Name;
83             t.Quantity = updatedTool.Quantity;
84             t.Price = updatedTool.Price;
85
86             // read all content of hardware.dat and replace required places
87             string str = File.ReadAllText(storeFileSystem.FileName);
88             File.ReadLines(storeFileSystem.FileName).ToList().ForEach(l =>
89             {
90                 string subString = l.Substring(0, 5).Trim();
91
92                 if (subString.Equals(t.RecordId.ToString()))
93                 {
94                     File.WriteAllText(storeFileSystem.FileName, str.Replace(l.ToString(), t.ToString()));
95                 }
96             });
97         }
98     }
99 }
100

```

```

1  using System;
2  using System.IO;
3  using System.Linq;
4
5  namespace Hw_17_8
6  {
7      5 references
8      class StoreFileSystem
9      {
10         private string fileName;
11
12         6 references
13         public string FileName
14         {
15             get { return fileName; }
16             set { fileName = value; }
17         }
18
19         FileStream fs;
20
21         /// <summary>
22         ///     Constructor for initializing StoreFileSystem
23         /// </summary>
24         /// <param name="fileName"> name of .dat file </param>
25         1 reference
26         public StoreFileSystem(string fileName)
27         {
28             this.fileName = fileName;
29         }
30
31         /// <summary>
32         ///     If given file with a fileName name does not exists, creates file and writes column names,
33         ///     otherwise truncates file and writes column names
34         /// </summary>
35         1 reference
36         public void CreateFile()
37         {
38             if (fileName is null)
39             {
40                 throw new ArgumentNullException(nameof(fileName));
41             }
42             string columnNames = string.Format("{0,-10} {1,-15} {2,-10} {3,-10}", "Record #", "Tool name", "Quantity", "Price");
43
44             if (File.Exists(fileName))
45             {
46                 fs = new FileStream(fileName, FileMode.Truncate, FileAccess.Write);
47                 Console.WriteLine(columnNames);
48                 fs.Close();
49                 return;
50             }
51
52             try
53             {
54                 fs = new FileStream(fileName, FileMode.CreateNew, FileAccess.Write);
55                 Console.WriteLine(columnNames);
56                 fs.Close();
57             }
58             catch (Exception e)
59             {
60                 throw new Exception(e.Message);
61             }
62         }
63     }
64 }

```

```

61 /// <summary>
62 ///     Writes given Tool into hardware.dat file
63 /// </summary>
64 /// <param name="tool"> is a Tool object that will be saved </param>
65 1 reference
66 public void WriteFile(Tool tool)
67 {
68     if (tool is null)
69     {
70         throw new ArgumentNullException(nameof(tool));
71     }
72     // formatted as written in hardware.dat file
73     string text = string.Format("{0,-10} {1,-15} {2,-10} {3,-10}", tool.RecordId, tool.Name, tool.Quantity, tool.Price);
74
75     using (FileStream fs = new FileStream(fileName, FileMode.Append, FileAccess.Write))
76     {
77         using (StreamWriter write = new StreamWriter(fs))
78         {
79             write.WriteLine(text);
80         }
81         fs.Close();
82     }
83
84     /// <summary>
85     ///     Reads all file and prints to the console
86     /// </summary>
87 1 reference
88 public void ReadFile()
89 {
90     File.ReadLines(fileName).ToList().ForEach(l => Console.WriteLine(l));
91 }
92
93 /// <summary>
94 ///     Finds given tool in hardware.dat file and returns its line as string
95 /// </summary>
96 /// <param name="tool"> is a Tool object that will be found </param>
97 /// <returns> the line that tool is written </returns>
98 1 reference
99 public String FindInFile(Tool tool)
100 {
101     try
102     {
103         return File.ReadLines(fileName).First(l => Int32.Parse(l.Substring(0, 5)).Equals(tool.RecordId));
104     }
105     catch (Exception e)
106     {
107         Console.WriteLine(e.Message);
108     }
109     return null;
110 }

```

```

110
111 /// <summary>
112 ///     Finds given tool in hardware.dat file and returns it as Tool object
113 /// </summary>
114 /// <param name="tool"> is a Tool object that will be found </param>
115 /// <returns> the tool in file </returns>
116 1 reference
117 public Tool FindToolInFile(Tool tool)
118 {
119     try
120     {
121         String foundTool = File.ReadLines(fileName).First(l => Int32.Parse(l.Substring(0, 5)).Equals(tool.RecordId));
122         if (foundTool != null)
123         {
124             int toolId = Int32.Parse(foundTool.Substring(0, 10));
125             string toolName = foundTool.Substring(11, 15);
126             int toolQuantity = Int32.Parse(foundTool.Substring(27, 10));
127             float toolPrice = float.Parse(foundTool.Substring(38, 10));
128             return new Tool(toolId, toolName, toolQuantity, toolPrice);
129         }
130     }
131     catch (Exception e)
132     {
133         Console.WriteLine(e.Message);
134     }
135     return null;
136 }

```

```

1  using System;
2  using System.Collections.Generic;
3
4  namespace Hw_17_8
5  {
6      0 references
7      class Program
8      {
9          private const string FILE_NAME = "hardware.dat";
10
11          0 references
12          static void Main(string[] args)
13          {
14              // creating StoreFileSystem to manage file processes, StoreToolSystem for editing the file
15              StoreFileSystem storeFileSystem = new StoreFileSystem(FILE_NAME);
16              StoreToolSystem storeToolSystem = new StoreToolSystem(storeFileSystem);
17
18              // creating Tool instances
19              Tool electricSander = new Tool("Electric sander", 18, 35.99f);
20              Tool hammer = new Tool("Hammer", 128, 10.00f);
21              Tool jigSaw = new Tool("Jig saw", 16, 14.25f);
22              Tool lawnMower = new Tool("Lawn mower", 10, 79.50f);
23              Tool powerSaw = new Tool("Power saw", 8, 89.99f);
24              Tool screwdriver = new Tool("Screwdriver", 236, 4.99f);
25              Tool sledgehammer = new Tool("Sledge hammer", 32, 19.75f);
26              Tool wrench = new Tool("Wrench", 65, 6.48f);
27              Tool newTool = new Tool("New Tool", 9999, 99.99f);
28              Tool updatedNewTool = new Tool("Updated Tool", 12345, 1.11f);
29
30              List<Tool> toolList = new List<Tool>()
31              {
32                  electricSander,
33                  hammer,
34                  jigSaw,
35                  lawnMower,
36                  powerSaw,
37                  screwdriver,
38                  sledgehammer,
39                  wrench
40              };
41
42              storeFileSystem.CreateFile();
43
44              // saving toolList into hardware.dat file
45              storeToolSystem.SaveTool(toolList);
46
47              // newTool will be saved to hardware.dat file but before toolList list
48              toolList.Add(newTool);
49              storeToolSystem.SaveTool(newTool);
50
51              // deleting an element of toolList
52              storeToolSystem.DeleteTool(electricSander);
53
54              // finding an element of toolList
55              // Console.WriteLine(storeFileSystem.FindInFile(wrench));
56
57              // updating tool which is added by SaveTool() method
58              storeToolSystem.UpdateTool(newTool, updatedNewTool);
59
60              // read content of hardware.dat file
61              storeFileSystem.ReadFile();
62
63              // to hold console
64              Console.ReadKey();
65          }
66      }
67  }

```

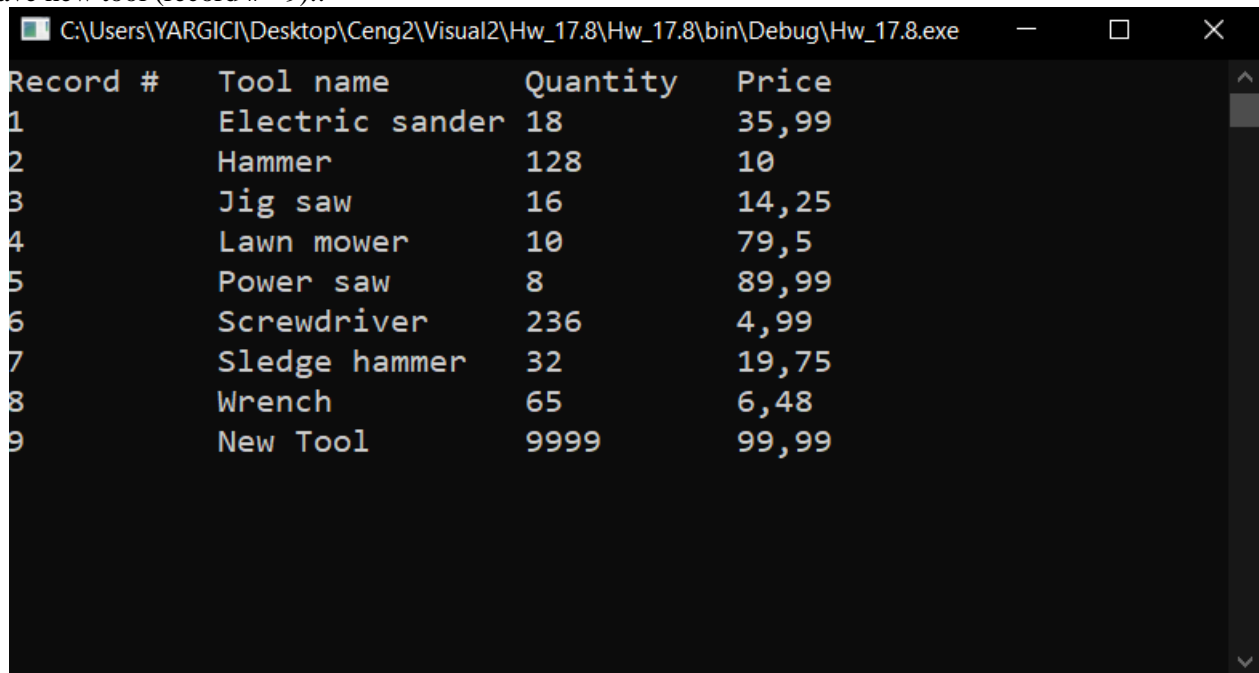
Outputs of program:

- List all:



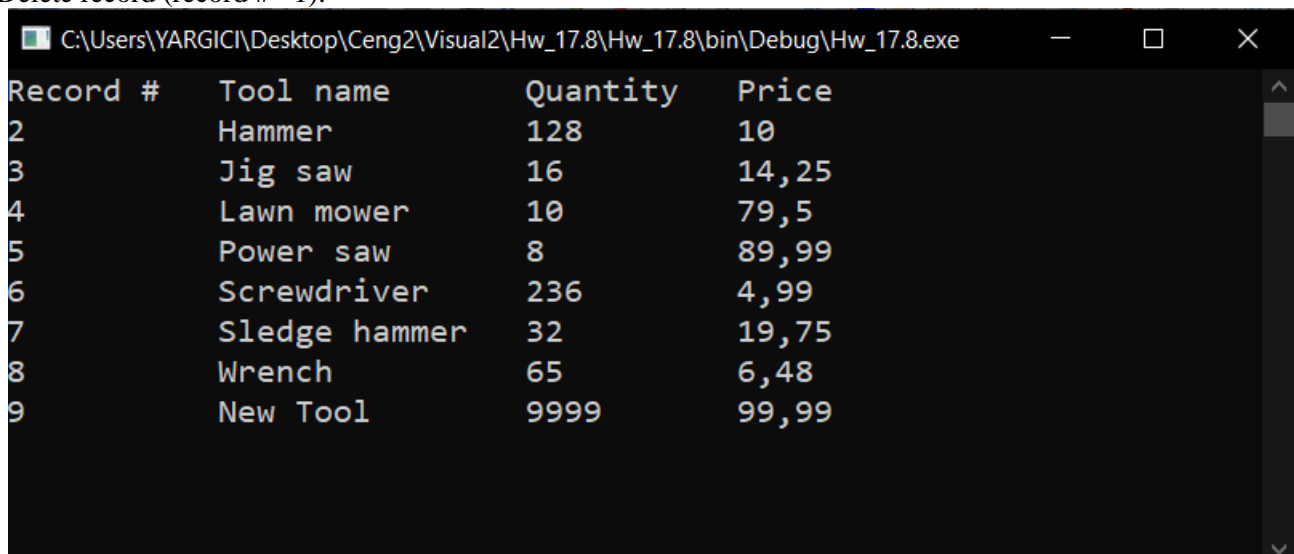
| Record # | Tool name | Quantity | Price |
|----------|-----------------|----------|-------|
| 1 | Electric sander | 18 | 35,99 |
| 2 | Hammer | 128 | 10 |
| 3 | Jig saw | 16 | 14,25 |
| 4 | Lawn mower | 10 | 79,5 |
| 5 | Power saw | 8 | 89,99 |
| 6 | Screwdriver | 236 | 4,99 |
| 7 | Sledge hammer | 32 | 19,75 |
| 8 | Wrench | 65 | 6,48 |

- Save new tool (record #= 9):



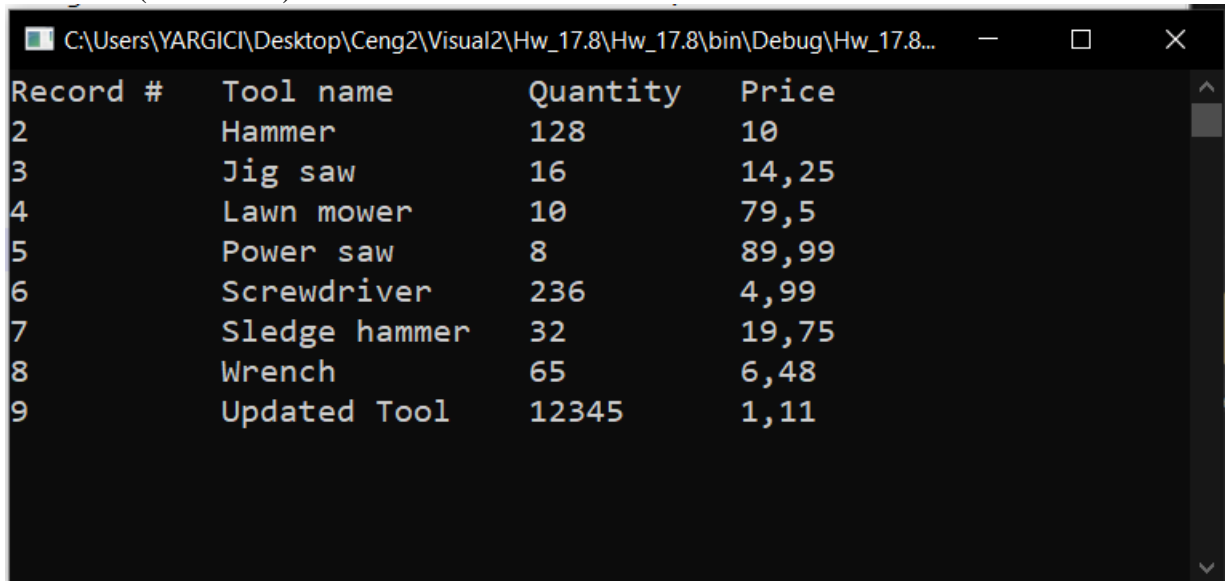
| Record # | Tool name | Quantity | Price |
|----------|-----------------|----------|-------|
| 1 | Electric sander | 18 | 35,99 |
| 2 | Hammer | 128 | 10 |
| 3 | Jig saw | 16 | 14,25 |
| 4 | Lawn mower | 10 | 79,5 |
| 5 | Power saw | 8 | 89,99 |
| 6 | Screwdriver | 236 | 4,99 |
| 7 | Sledge hammer | 32 | 19,75 |
| 8 | Wrench | 65 | 6,48 |
| 9 | New Tool | 9999 | 99,99 |

- Delete record (record #= 1):



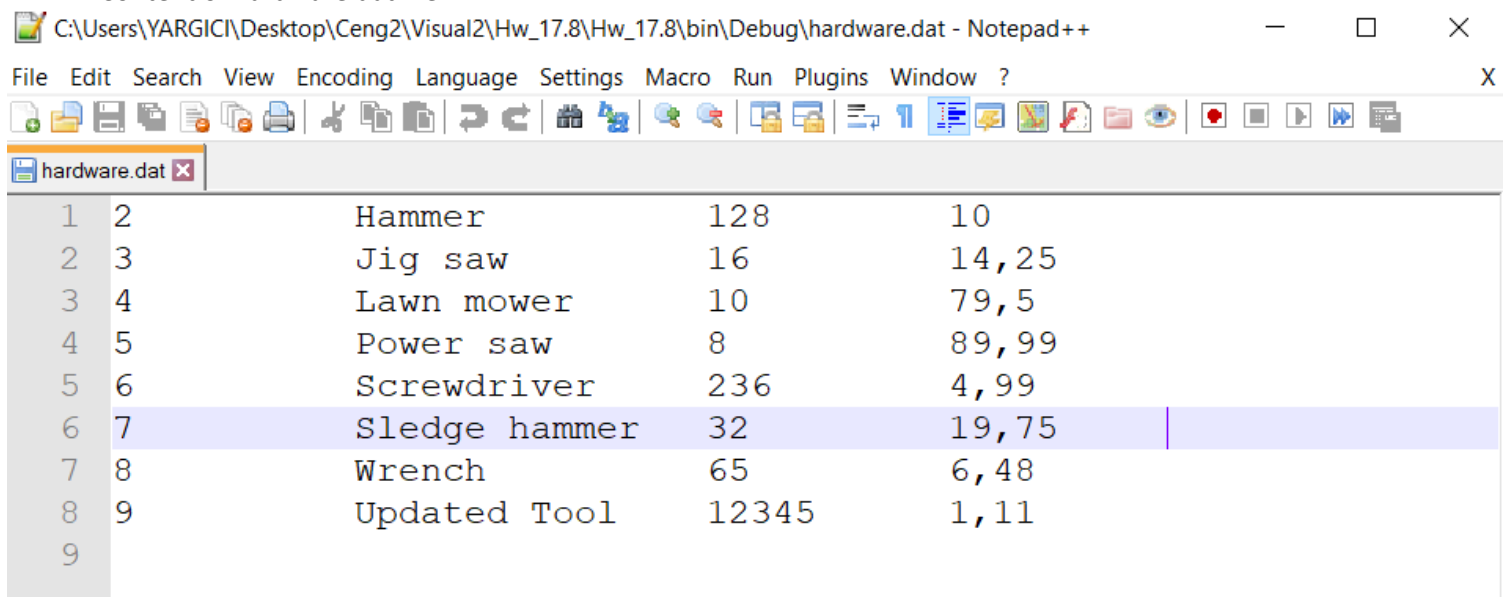
| Record # | Tool name | Quantity | Price |
|----------|---------------|----------|-------|
| 2 | Hammer | 128 | 10 |
| 3 | Jig saw | 16 | 14,25 |
| 4 | Lawn mower | 10 | 79,5 |
| 5 | Power saw | 8 | 89,99 |
| 6 | Screwdriver | 236 | 4,99 |
| 7 | Sledge hammer | 32 | 19,75 |
| 8 | Wrench | 65 | 6,48 |
| 9 | New Tool | 9999 | 99,99 |

- Update record (record #= 9):



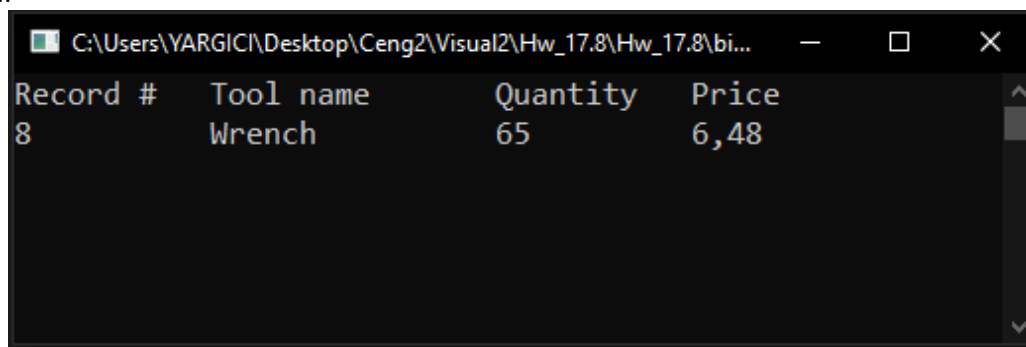
| Record # | Tool name | Quantity | Price |
|----------|---------------|----------|-------|
| 2 | Hammer | 128 | 10 |
| 3 | Jig saw | 16 | 14,25 |
| 4 | Lawn mower | 10 | 79,5 |
| 5 | Power saw | 8 | 89,99 |
| 6 | Screwdriver | 236 | 4,99 |
| 7 | Sledge hammer | 32 | 19,75 |
| 8 | Wrench | 65 | 6,48 |
| 9 | Updated Tool | 12345 | 1,11 |

- Content of hardware.dat file:



| Record # | Tool name | Quantity | Price |
|----------|---------------|----------|-------|
| 1 2 | Hammer | 128 | 10 |
| 2 3 | Jig saw | 16 | 14,25 |
| 3 4 | Lawn mower | 10 | 79,5 |
| 4 5 | Power saw | 8 | 89,99 |
| 5 6 | Screwdriver | 236 | 4,99 |
| 6 7 | Sledge hammer | 32 | 19,75 |
| 7 8 | Wrench | 65 | 6,48 |
| 8 9 | Updated Tool | 12345 | 1,11 |

- Find record:



| Record # | Tool name | Quantity | Price |
|----------|-----------|----------|-------|
| 8 | Wrench | 65 | 6,48 |