

ELITE SERVICE CENTRE

Term Project



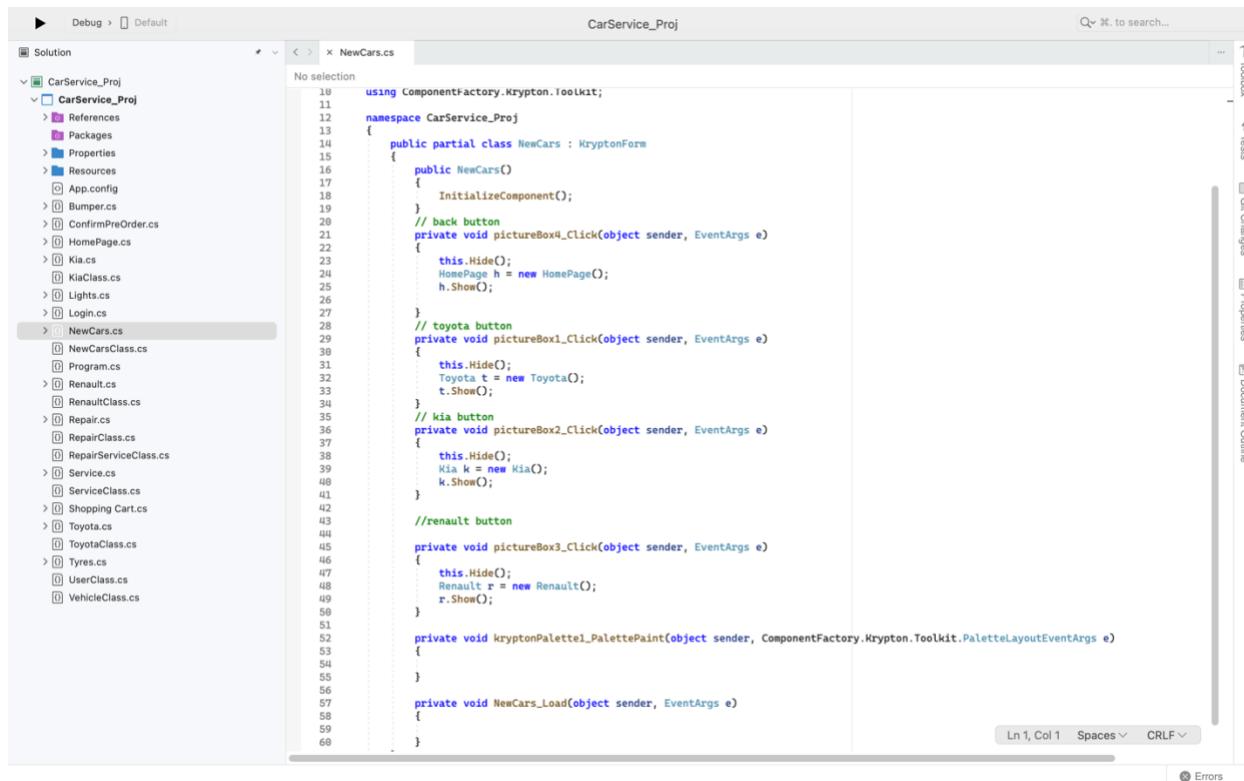
Prepared By:

Maysoon Wafa
Nada Hamada
Ahmed Elshennawy
Feras Anas

Handed To:

Eng. Mohamed Fawzy

New Cars Class



The screenshot shows the Visual Studio IDE interface with the following details:

- Solution Explorer:** Shows the project structure under "CarService_Proj". The file "NewCars.cs" is selected.
- Code Editor:** Displays the C# code for the "NewCars.cs" class. The code defines a partial class "NewCars" that inherits from "KryptonForm". It includes event handlers for button clicks (pictureBox1_Click, pictureBox2_Click, pictureBox3_Click) which hide the current form and show different car classes (HomePage, Toyota, Kia, Renault). It also includes a palette paint event handler (kryptonPalette1_PalettePaint) and a load event handler (NewCars_Load).
- Status Bar:** Shows "Ln 1, Col 1" and other standard status bar information.
- Toolbars and Menus:** Standard Visual Studio toolbars and menus are visible along the top.
- Right Panel:** Shows various tabs like "Toolbox", "Tests", "Get Changes", "Properties", and "Document Outline".

- Base class for all new cars class (Kia, Renault, Toyota) with abstract method for the Manufacturer label.
- Derived classes inherit this abstract method and declare new abstract methods for the other car spec labels.

Kia Form:

The screenshot shows the Visual Studio IDE with the solution 'CarService_Proj' open. The file 'Kia.cs' is selected in the Solution Explorer. The code editor displays the following C# code:

```

16     {
17         SportageClass sportage = new SportageClass();
18         CeratoClass cerato = new CeratoClass();
19         public Kia()
20         {
21             InitializeComponent();
22         }
23
24         private void Kia_Load(object sender, EventArgs e)
25         {
26             //cerato specs
27             MakeC.Text = cerato.Manufacturer();
28             ModelC.Text = cerato.Model();
29             YearC.Text = Convert.ToString(cerato.ModelYear());
30             ColorC.Text = cerato.Color();
31             PriceC.Text = Convert.ToString(cerato.Price());
32
33             //sportage specs
34             MakeS.Text = sportage.Manufacturer();
35             ModelS.Text = sportage.Model();
36             YearS.Text = Convert.ToString(sportage.ModelYear());
37             ColorS.Text = sportage.Color();
38             PriceS.Text = Convert.ToString(sportage.Price());
39
40         }
41
42         private void MakeC_Click(object sender, EventArgs e)
43         {
44         }
45
46         private void ModelC_Click(object sender, EventArgs e)
47         {
48         }
49
50         private void pictureBox4_Click(object sender, EventArgs e)
51         {
52             this.Hide();
53             NewCars n = new NewCars();
54             n.Show();
55         }
56
57         private void preorder_Click(object sender, EventArgs e)
58         {
59             ConfirmPreOrder order = new ConfirmPreOrder();
60             order.Show();
61         }
62
63         private void button2_Click(object sender, EventArgs e)
64         {
65             ConfirmPreOrder order = new ConfirmPreOrder();
66         }

```

The code implements a form with two buttons: 'Kia_Load' and 'button2_Click'. It uses two classes, 'SportageClass' and 'CeratoClass', to initialize car specifications. It also includes logic to show a 'NewCars' form and a 'ConfirmPreOrder' dialog.

- Shows and hides forms upon corresponding click
- Car specs are shown as labels where their text is modified using inherited methods from the base class of the car manufacturer.

The screenshot shows the Visual Studio IDE with the solution 'CarService_Proj' open. The file 'KiaClass.cs' is selected in the Solution Explorer. The code editor displays the following C# code:

```

3     public abstract class KiaClass : NewCarsClass
4     {
5         public string Make;
6
7         public override string Manufacturer()
8         {
9             return "Kia";
10        }
11
12        public abstract string Model();
13        public abstract int ModelYear();
14        public abstract string Color();
15        public abstract int Price();
16
17    }
18
19    public class SportageClass : KiaClass
20    {
21        public override string Manufacturer()
22        {
23            return "Kia";
24        }
25
26        int price;
27        public override int Price()
28        {
29            return 50000;
30        }
31
32        public string model;
33        public override string Model()
34        {
35            return "Sportage";
36        }
37
38        public int year;
39        public override int ModelYear()
40        {
41            return 2020;
42        }
43
44        public string colour;
45        public override string Color()
46        {
47            return "Grey";
48        }
49
50    }
51
52    public class CeratoClass : KiaClass
53    {
54        ...
55    }

```

The code defines an abstract base class 'KiaClass' that inherits from 'NewCarsClass'. It has four abstract properties: 'Make', 'Manufacturer', 'Model', and 'Price'. Two concrete classes, 'SportageClass' and 'CeratoClass', inherit from 'KiaClass'. They implement the abstract methods and provide concrete implementations for 'Model', 'Price', and 'Color'.

Repair Form:

The screenshot shows the Visual Studio IDE with the 'RepairClass.cs' file open in the editor. The code defines a class 'RepairClass' that inherits from 'RepairServiceClass'. It contains methods for calculating the price of different repair types based on size and number of tyres. The code also includes methods for lights and bumpers.

```

1  using System;
2  namespace Codes
3  {
4      public class RepairClass : RepairServiceClass
5      {
6          public int Tyres(int nentyres, int size)
7          {
8              int sizeA = 215;
9              int priceA = 500;
10             int sizeB = 235;
11             int priceB = 700;
12             if (size == sizeA)
13             {
14                 return nentyres * priceA;
15             }
16             else if (size == sizeB)
17             {
18                 return nentyres * priceB;
19             }
20             else return 0;
21         }
22         public int Lights(string type)
23         {
24             if (type == "Headlights")
25             {
26                 return 800;
27             }
28             if (type == "Taillights")
29             {
30                 return 600;
31             }
32             if (type == "FogLights")
33             {
34                 return 300;
35             }
36             else return 0;
37         }
38         public int Bumpers(string type)
39         {
40             if (type == "Rear")
41             {
42                 return 100;
43             }
44             if (type == "Front")
45             {
46                 return 100;
47             }
48             else return 0;
49         }
50     }
51 }

```

- This code determines the price of each repair option for when adding to cart.

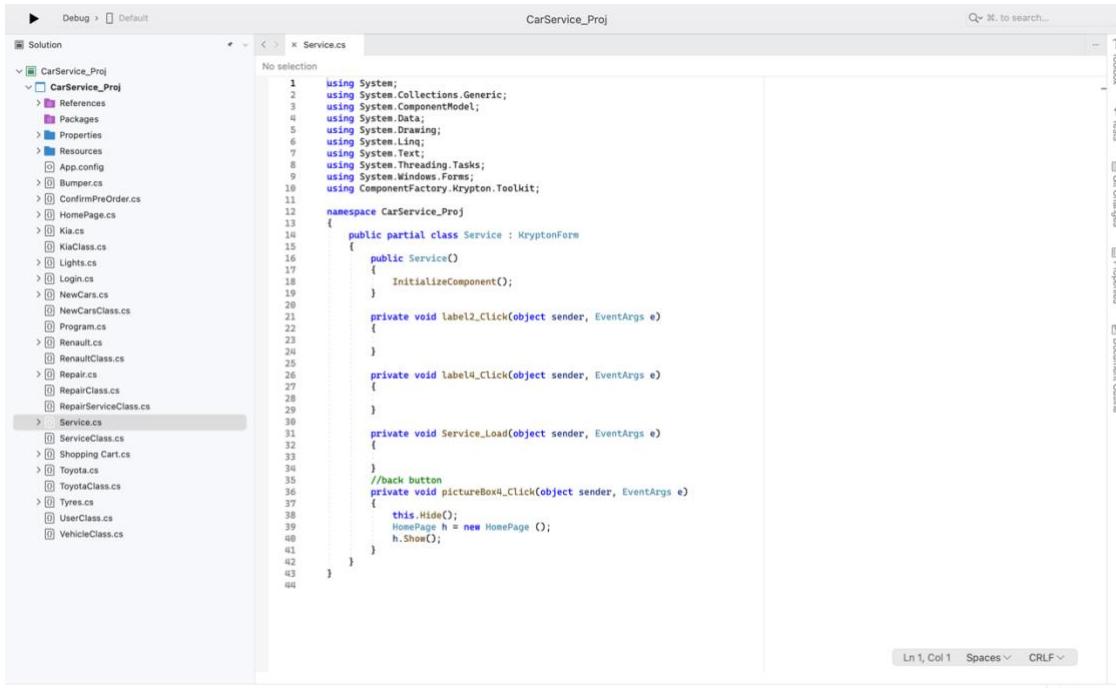
The screenshot shows the Visual Studio IDE with the 'Repair.cs' file open in the editor. The code defines a partial class 'Repair' that inherits from 'KryptonForm'. It contains several event handlers for buttons like 'label1_Click', 'pictureBox3_Click', etc., which perform actions such as hiding the current form and showing other forms like 'Lights', 'Tyres', or 'HomePage'.

```

11  namespace CarService_Proj
12  {
13      public partial class Repair : KryptonForm
14      {
15          public Repair()
16          {
17              InitializeComponent();
18          }
19          private void label1_Click(object sender, EventArgs e)
20          {
21          }
22          //lights
23          private void pictureBox3_Click(object sender, EventArgs e)
24          {
25              this.Hide();
26              Lights l = new Lights();
27              l.Show();
28          }
29          //tyres
30          private void pictureBox1_Click(object sender, EventArgs e)
31          {
32              this.Hide();
33              Tyres t = new Tyres();
34              t.Show();
35          }
36          //back button
37          private void pictureBox4_Click(object sender, EventArgs e)
38          {
39              this.Hide();
40              HomePage h = new HomePage();
41              h.Show();
42          }
43          private void label3_Click(object sender, EventArgs e)
44          {
45          }
46          //bumper button
47          private void pictureBox2_Click(object sender, EventArgs e)
48          {
49              this.Hide();
50              Bumper b = new Bumper();
51              b.Show();
52          }
53          private void Repair_Load(object sender, EventArgs e)
54          {
55          }
56      }
57 }

```

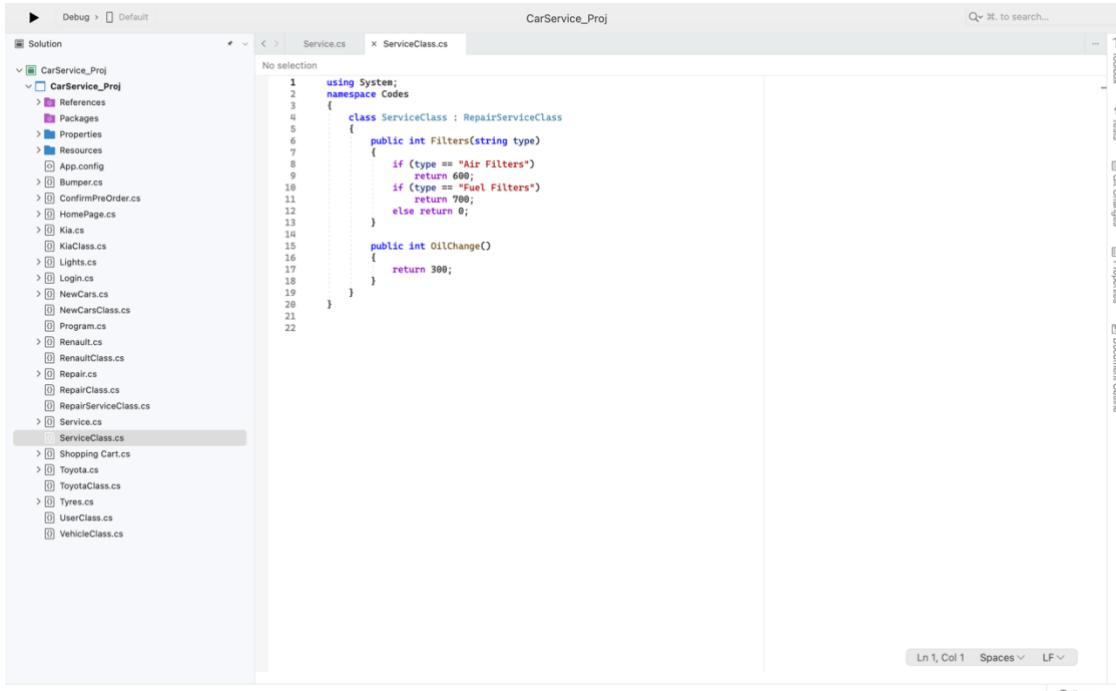
Service Form:



The screenshot shows the Visual Studio IDE with the 'Service.cs' file open in the code editor. The code defines a partial class 'Service' that inherits from 'KryptonForm'. It includes event handlers for button clicks and a method to load the service form. The code editor has syntax highlighting and line numbers.

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10 using ComponentFactory.Krypton.Toolkit;
11 
12 namespace CarService_Proj
13 {
14     public partial class Service : KryptonForm
15     {
16         public Service()
17         {
18             InitializeComponent();
19         }
20 
21         private void label2_Click(object sender, EventArgs e)
22         {
23         }
24 
25         private void label4_Click(object sender, EventArgs e)
26         {
27         }
28 
29         private void Service_Load(object sender, EventArgs e)
30         {
31         }
32 
33         //back button
34         private void pictureBox4_Click(object sender, EventArgs e)
35         {
36             this.Hide();
37             HomePage h = new HomePage();
38             h.Show();
39         }
40 
41     }
42 }
43 
```

- This code determines the price of each service option for when adding to cart.



The screenshot shows the Visual Studio IDE with the 'ServiceClass.cs' file open in the code editor. The code defines a class 'ServiceClass' that implements the 'RepairServiceClass' interface. It contains methods for calculating the price of filters based on type and for performing an oil change. The code editor has syntax highlighting and line numbers.

```
1  using System;
2  namespace Codes
3  {
4      class ServiceClass : RepairServiceClass
5      {
6          public int Filters(string type)
7          {
8              if (type == "Air Filters")
9                  return 600;
10             if (type == "Fuel Filters")
11                 return 700;
12             else return 0;
13         }
14 
15         public int OilChange()
16         {
17             return 300;
18         }
19     }
20 
```