Részletes Magyarázat a PizzaOrderingApp WPF Alkalmazáshoz

Martin

November 18, 2024

Contents

1 Bevezetés

Ez a dokumentum részletesen bemutatja a **PizzaOrderingApp** WPF alkalmazás kódját. Minden egyes fájl és annak sorai részletes magyarázattal vannak ellátva, hogy könnyebben érthetővé váljon az alkalmazás felépítése és működése.

2 Models/Pizza.cs

2.1 Kód

```
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace PizzaOrderingApp.Models
{
    public class Pizza : INotifyPropertyChanged
    {
        private string name;
        private decimal basePrice;
        private ObservableCollection<Topping> toppings;

        public Pizza(string name, decimal basePrice)
        {
            Name = name;
            BasePrice = basePrice;
            Toppings = new ObservableCollection<Topping>();
            Toppings.CollectionChanged +=
    Toppings_CollectionChanged;
        }

        public string Name
        {
            get => name;
            set { name = value; OnPropertyChanged();
    OnPropertyChanged(nameof(Total)); }
        }

        public decimal BasePrice
        {
            get => basePrice;
            set { basePrice = value; OnPropertyChanged();
    OnPropertyChanged(nameof(Total)); }
        }

        public ObservableCollection<Topping> Toppings
        {
            get => toppings;
            set
            {
                if (toppings != null)
                {
```

```
40                      toppings.CollectionChanged -=
    Toppings_CollectionChanged;
41                  }
42                  toppings = value;
43                  if (toppings != null)
44                  {
45                      toppings.CollectionChanged +=
    Toppings_CollectionChanged;
46                  }
47                  OnPropertyChanged();
48                  OnPropertyChanged(nameof(Total));
49              }
50          }
51
52          private void Toppings_CollectionChanged(object sender,
    NotifyCollectionChangedEventArgs e)
53          {
54              OnPropertyChanged(nameof(Total));
55          }
56
57          public decimal Total => CalculateTotalPrice();
58
59          public decimal CalculateTotalPrice()
60          {
61              decimal total = BasePrice;
62              foreach (var topping in Toppings)
63              {
64                  total += topping.Price;
65              }
66              return total;
67          }
68
69          public event PropertyChangedEventHandler PropertyChanged;
70
71          protected void OnPropertyChanged([CallerMemberName] string
    name = "")
72          {
73              PropertyChanged?.Invoke(this, new
    PropertyChangedEventArgs(name));
74          }
75
76          public override string ToString()
77          {
78              return $"{Name} - ${Total:0.00}";
79          }
80      }
81  }
```

Listing 1: Models/Pizza.cs

2.2 Magyarázat

1. **Sor 1-3**: Az alkalmazás szükséges névtereit importáljuk, amelyek lehetővé teszik a `ObservableCollection`, `INotifyPropertyChanged` interfész és az `CallerMemberName` attribútum használatát.

2. **Sor 5**: A `namespace PizzaOrderingApp.Models` deklaráció meghatározza azt a névteret, amelyben a modell osztályok találhatók.

3. **Sor 7**: A `public class Pizza : INotifyPropertyChanged` osztálydefiníció létrehozza a `Pizza` osztályt, amely implementálja az `INotifyPropertyChanged` interfészt. Ez az interfész lehetővé teszi, hogy a tulajdonságok változásáról értesítést küldjünk a felhasználói felületnek.

4. **Sor 9-11**: Privát mezők deklarálása:

   - `name`: A pizza neve.
   - `basePrice`: A pizza alapára.
   - `toppings`: A pizzához hozzáadott toppingok gyűjteménye.

5. **Sor 13-18**: A `Pizza` konstruktor:

   - `Name = name;`: Beállítja a pizza nevét.
   - `BasePrice = basePrice;`: Beállítja a pizza alapárát.
   - `Toppings = new ObservableCollection<Topping>();`: Inicializálja a `Toppings` gyűjteményt.
   - `Toppings.CollectionChanged += Toppings_CollectionChanged;`: Feliratkozik a `CollectionChanged` eseményre, hogy értesítést kapjon a gyűjtemény változásairól.

6. **Sor 20-24**: A `Name` tulajdonság:

   - `get => name;`: Getter, amely visszaadja a `name` értékét.
   - `set { ... }`: Setter, amely beállítja a `name` értékét, majd értesíti a felhasználói felületet a változásról, valamint a `Total` tulajdonság frissüléséről.

7. **Sor 26-30**: A `BasePrice` tulajdonság hasonló a `Name` tulajdonsághoz:

   - Getter és setter, amelyek beállítják az alapárat és értesítik a felhasználói felületet a változásról.

8. **Sor 32-43**: A `Toppings` tulajdonság:

   - `get => toppings;`: Getter, amely visszaadja a `toppings` gyűjteményt.
   - `set { ... }`: Setter, amely először eltávolítja a korábbi `CollectionChanged` eseménykezelőt, majd beállítja az új gyűjteményt, és újra feliratkozik az eseményre. Végül értesíti a felhasználói felületet a `Total` tulajdonság frissüléséről.

9. **Sor 45-48**: A `Toppings_CollectionChanged` metódus:

   - Ez a metódus akkor hívódik meg, amikor a `Toppings` gyűjtemény megváltozik (topping hozzáadása vagy eltávolítása).

- `OnPropertyChanged(nameof(Total));`: Értesíti a felhasználói felületet a `Total` tulajdonság változásáról.

10. **Sor 50**: A `Total` tulajdonság:
    - Csak egy getter, amely a `CalculateTotalPrice()` metódus értékét adja vissza.
11. **Sor 52-58**: A `CalculateTotalPrice` metódus:
    - `decimal total = BasePrice;`: Elindítja az összegzést az alapárral.
    - `foreach (var topping in Toppings) { ... }`: Végigiterál a toppingokon és hozzáadja azok árát a teljes összeghez.
    - `return total;`: Visszaadja a kiszámított teljes árat.
12. **Sor 60**: Az `event PropertyChangedEventHandler PropertyChanged;` deklarációja, amelyet az `INotifyPropertyChanged` interfész megkövetel.
13. **Sor 62-66**: Az `OnPropertyChanged` metódus:
    - Ez a metódus hívódik meg, amikor egy tulajdonság értéke megváltozik.
    - `PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));`: Esemény generálása a megadott tulajdonság névvel.
14. **Sor 68-71**: A `ToString()` metódus felülírása:
    - `return` "$Name-$`Total:0.00";`: Visszaadja a pizza nevét és a teljes árát formázott stringként.

3 Models/Topping.cs

3.1 Kód

```
1  namespace PizzaOrderingApp.Models
2  {
3      public class Topping
4      {
5          public string Name { get; set; }
6          public decimal Price { get; set; }
7
8          public Topping(string name, decimal price)
9          {
10             Name = name;
11             Price = price;
12         }
13     }
14 }
```

Listing 2: Models/Topping.cs

3.2 Magyarázat

1. **Sor 1**: A `namespace PizzaOrderingApp.Models` deklaráció meghatározza azt a névteret, amelyben a modell osztályok találhatók.

2. **Sor 3**: A `public class Topping` osztálydefiníció létrehozza a `Topping` osztályt, amely a pizzákhoz hozzáadható toppingokat reprezentálja.

3. **Sor 5-6**: Tulajdonságok deklarálása:

   - `public string Name { get; set; }`: A topping neve.
   - `public decimal Price { get; set; }`: A topping ára.

4. **Sor 8-12**: A `Topping` konstruktor:

   - `Name = name;`: Beállítja a topping nevét.
   - `Price = price;`: Beállítja a topping árát.

4 Models/Cart.cs

4.1 Kód

```
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Collections.Specialized;

namespace PizzaOrderingApp.Models
{
    public class Cart : INotifyPropertyChanged
    {
        public ObservableCollection<Pizza> Pizzas { get; set; }

        private decimal total;
        public decimal Total
        {
            get => total;
            set
            {
                if (total != value)
                {
                    total = value;
                    OnPropertyChanged();
                }
            }
        }

        public Cart()
        {
            Pizzas = new ObservableCollection<Pizza>();
            Pizzas.CollectionChanged += Pizzas_CollectionChanged;
        }
```

```
31
32      private void Pizzas_CollectionChanged(object sender,
    NotifyCollectionChangedEventArgs e)
33      {
34          if (e.NewItems != null)
35          {
36              foreach (Pizza p in e.NewItems)
37              {
38                  p.PropertyChanged += Pizza_PropertyChanged;
39              }
40          }
41          if (e.OldItems != null)
42          {
43              foreach (Pizza p in e.OldItems)
44              {
45                  p.PropertyChanged -= Pizza_PropertyChanged;
46              }
47          }
48          CalculateTotal();
49      }
50
51      private void Pizza_PropertyChanged(object sender,
    PropertyChangedEventArgs e)
52      {
53          if (e.PropertyName == "Total")
54          {
55              CalculateTotal();
56          }
57      }
58
59      public void AddPizza(Pizza pizza)
60      {
61          Pizzas.Add(pizza);
62      }
63
64      public void RemovePizza(Pizza pizza)
65      {
66          Pizzas.Remove(pizza);
67      }
68
69      private void CalculateTotal()
70      {
71          decimal sum = 0;
72          foreach (var pizza in Pizzas)
73          {
74              sum += pizza.Total;
75          }
76          Total = sum;
77      }
78
79      public event PropertyChangedEventHandler PropertyChanged;
80
81      protected void OnPropertyChanged([CallerMemberName] string
    name = "")
82      {
83          PropertyChanged?.Invoke(this, new
```

```
    PropertyChangedEventArgs(name));
84          }
85      }
86  }
```

Listing 3: Models/Cart.cs

4.2 Magyarázat

1. **Sor 1-4**: Névterek importálása, amelyek lehetővé teszik az `ObservableCollection`, `INotifyPropertyChanged` interfész és a `NotifyCollectionChangedEventArgs` használatát.

2. **Sor 6**: A `namespace PizzaOrderingApp.Models` deklaráció meghatározza azt a névteret, amelyben a modell osztályok találhatók.

3. **Sor 8**: A `public class Cart : INotifyPropertyChanged` osztálydefiníció létrehozza a `Cart` osztályt, amely implementálja az `INotifyPropertyChanged` interfészt. Ez az osztály felelős a felhasználó kosarának kezeléséért.

4. **Sor 10-12**: Tulajdonságok deklarálása:

   - `public ObservableCollection<Pizza> Pizzas { get; set; }`: Az összesített pizzák gyűjteménye a kosárban.

5. **Sor 14-21**: A `Total` tulajdonság:

   - `private decimal total;`: Privát mező a teljes összeg tárolására.
   - `public decimal Total { get => total; set { ... } }`: Getter és setter, ahol a setter csak akkor állítja be az értéket, ha az megváltozott, és értesíti a felhasználói felületet a változásról.

6. **Sor 23-28**: A `Cart` konstruktor:

   - `Pizzas = new ObservableCollection<Pizza>();`: Inicializálja a `Pizzas` gyűjteményt.
   - `Pizzas.CollectionChanged += Pizzas_CollectionChanged;`: Feliratkozik a `CollectionChanged` eseményre, hogy értesítést kapjon a gyűjtemény változásairól.

7. **Sor 30-43**: A `Pizzas_CollectionChanged` metódus:

   - `if (e.NewItems != null) { ... }`: Feliratkozik a hozzáadott pizzák `PropertyChanged` eseményére.
   - `if (e.OldItems != null) { ... }`: Eltávolítja a feliratkozást a kivont pizzák `PropertyChanged` eseményéről.
   - `CalculateTotal();`: Újraszámolja a kosár teljes összegét.

8. **Sor 45-50**: A `Pizza_PropertyChanged` metódus:

- `if (e.PropertyName == "Total") { ... }`: Ha egy pizza `Total` tulajdonsága megváltozott, újraszámolja a kosár teljes összegét.

9. **Sor 52-58**: `AddPizza` és `RemovePizza` metódusok:
   - `AddPizza(Pizza pizza)`: Hozzáad egy pizzát a `Pizzas` gyűjteményhez.
   - `RemovePizza(Pizza pizza)`: Eltávolít egy pizzát a `Pizzas` gyűjteményből.
10. **Sor 60-67**: `CalculateTotal` metódus:
    - `decimal sum = 0;`: Kezdeti összeg beállítása.
    - `foreach (var pizza in Pizzas) { ... }`: Végigiterál a kosárban lévő pizzákon és összeadja azok `Total` értékét.
    - `Total = sum;`: Beállítja a `Total` tulajdonság értékét.
11. **Sor 69-73**: Az `INotifyPropertyChanged` interfész implementálása:
    - `public event PropertyChangedEventHandler PropertyChanged;`: Az esemény deklarálása.
    - `protected void OnPropertyChanged([CallerMemberName] string name = "") { ... }`: Metódus az esemény hívására.

5 Views/MainWindow.xaml

5.1 Kód

```
<Window x:Class="PizzaOrderingApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:PizzaOrderingApp"
        Title="Pizza Ordering" Height="600" Width="900">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="3*" />
            <ColumnDefinition Width="2*" />
        </Grid.ColumnDefinitions>

        <!-- Pizza Selection Area -->
        <StackPanel Grid.Column="0" Margin="10">
            <TextBlock Text="V laszd ki a pizz id" FontSize="24" Margin="0,0,0,20" HorizontalAlignment="Center"/>
            <WrapPanel x:Name="PizzaWrapPanel" HorizontalAlignment="Center" />
            <Button Content="Toppingok M dos t sa" Width="150" Height="40" Margin="0,20,0,0" Click="ModifyToppings_Click"/>
            <Button Content="Rendel s Lead sa" Width="150" Height="40" Margin="0,10,0,0" Click="PlaceOrder_Click"/>
        </StackPanel>

```

```
20          <!-- Cart Area -->
21          <StackPanel Grid.Column="1" Margin="10">
22              <TextBlock Text="A Kosarad" FontSize="24" Margin="
    0,0,0,20" HorizontalAlignment="Center"/>
23              <ListBox x:Name="CartListBox" Height="400" ItemsSource=
    "{Binding Pizzas}">
24                  <ListBox.ItemTemplate>
25                      <DataTemplate>
26                          <Border BorderBrush="Gray" BorderThickness=
    "1" Padding="5" Margin="5">
27                              <StackPanel>
28                                  <TextBlock Text="{Binding Name}"
    FontWeight="Bold" FontSize="16"/>
29                                  <TextBlock Text="{Binding BasePrice
    , StringFormat=Alap r: ${0:0.00}}" />
30                                  <TextBlock Text="Toppingok:"
    FontWeight="SemiBold" Margin="0,5,0,0"/>
31                                  <ItemsControl ItemsSource="{Binding
     Toppings}">
32                                      <ItemsControl.ItemTemplate>
33                                          <DataTemplate>
34                                              <TextBlock Text="{
    Binding Name}" />
35                                          </DataTemplate>
36                                      </ItemsControl.ItemTemplate>
37                                  </ItemsControl>
38                                  <TextBlock Text="{Binding Total,
    StringFormat= sszesen: ${0:0.00}}" FontWeight="Bold" Margin="
    0,5,0,0"/>
39                              </StackPanel>
40                          </Border>
41                      </DataTemplate>
42                  </ListBox.ItemTemplate>
43              </ListBox>
44              <TextBlock Text="{Binding Total, StringFormat=
     sszesen: ${0:0.00}}" FontSize="20" Margin="0,10,0,0"
    HorizontalAlignment="Center"/>
45              <Button Content="Fizet s" Width="150" Height="40"
    Margin="0,20,0,0" Click="Checkout_Click"/>
46          </StackPanel>
47      </Grid>
48  </Window>
```

Listing 4: Views/MainWindow.xaml

5.2 Magyarázat

1. **Sor 1-6**: A `Window` definíciója, amely meghatározza az ablak osztályát (`PizzaOrderingApp.Mai` a névtereket és az ablak címét, magasságát, valamint szélességét.

2. **Sor 7-10**: A `Grid` elrendezés létrehozása két oszloppal:

   - Első oszlop szélessége `3*`, ami azt jelenti, hogy három részt kap a rendelkezésre álló helyből.

- Második oszlop szélessége `2*`, két részt kap.

3. **Sor 12-23**: A pizzák kiválasztási területének létrehozása (`Grid.Column=0`):
   - `TextBlock`: Cím a kiválasztási terület tetején.
   - `WrapPanel`: Dinamikusan elrendezett pizzák megjelenítésére szolgál, ahol minden pizza egy gombbal rendelkezik.
   - `Button`: Gomb a toppingok módosításához, amely a `ModifyToppings_Click` eseménykezelőt hívja meg.
   - `Button`: Gomb a rendelés leadásához, amely a `PlaceOrder_Click` eseménykezelőt hívja meg.
4. **Sor 25-47**: A kosár területének létrehozása (`Grid.Column=1`):
   - `TextBlock`: Cím a kosár terület tetején.
   - `ListBox`: A kosárban lévő pizzák listájának megjelenítése. Az `ItemsSource` a `Pizzas` tulajdonságra van kötve a `DataContext` révén.
   - `ListBox.ItemTemplate`: Egy `DataTemplate`, amely meghatározza, hogyan jelenjenek meg a kosárban lévő pizzák.
     - `Border`: Keret a pizza információknak.
     - `StackPanel`: A pizza adatok függőleges elrendezése.
     - `TextBlock`: A pizza neve.
     - `TextBlock`: A pizza alapárának megjelenítése.
     - `TextBlock`: Toppingok címkéje.
     - `ItemsControl`: A pizza toppingjainak listázása.
     - `TextBlock`: A pizza teljes árának megjelenítése.
   - `TextBlock`: A kosár teljes összegének megjelenítése.
   - `Button`: Gomb a fizetéshez, amely a `Checkout_Click` eseménykezelőt hívja meg.

6 Views/MainWindow.xaml.cs

6.1 Kód

```
using PizzaOrderingApp.Models;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;

namespace PizzaOrderingApp
{
    public partial class MainWindow : Window
    {
```

```
12          private List<Pizza> Pizzas;
13          private List<Topping> AvailableToppings;
14          private Cart CurrentCart;
15
16          public MainWindow()
17          {
18              InitializeComponent();
19              InitializeData();
20              DisplayPizzas();
21              CurrentCart = new Cart();
22              this.DataContext = CurrentCart;
23          }
24
25          private void InitializeData()
26          {
27              // Toppings inicializ l sa
28              AvailableToppings = new List<Topping>
29              {
30                  new Topping("Pepperoni", 1.0m),
31                  new Topping("Mushrooms", 0.5m),
32                  new Topping("Onions", 0.5m),
33                  new Topping("Sausage", 1.0m),
34                  new Topping("Bacon", 1.0m),
35                  new Topping("Extra cheese", 0.75m),
36                  new Topping("Black olives", 0.5m),
37                  new Topping("Green peppers", 0.5m),
38                  new Topping("Pineapple", 0.75m),
39                  new Topping("Spinach", 0.5m)
40              };
41
42              // Pizz k inicializ l sa
43              Pizzas = new List<Pizza>
44              {
45                  new Pizza("Margherita", 5.0m),
46                  new Pizza("Pepperoni", 6.0m),
47                  new Pizza("BBQ Chicken", 7.0m),
48                  new Pizza("Veggie", 6.5m)
49              };
50          }
51
52          private void DisplayPizzas()
53          {
54              foreach (var pizza in Pizzas)
55              {
56                  // Border l trehoz sa a pizza inform ci knak
    s  gombnak
57                  Border border = new Border
58                  {
59                      BorderBrush = Brushes.Black,
60                      BorderThickness = new Thickness(1),
61                      CornerRadius = new CornerRadius(5),
62                      Margin = new Thickness(10),
63                      Padding = new Thickness(10),
64                      Width = 200,
65                      Height = 150,
66                      Background = Brushes.LightYellow
```

```
                };

                // StackPanel l trehoz sa a border belsej be
                StackPanel stack = new StackPanel();

                // Pizza neve
                TextBlock name = new TextBlock
                {
                    Text = pizza.Name,
                    FontSize = 18,
                    FontWeight = FontWeights.Bold,
                    TextAlignment = TextAlignment.Center
                };

                // Alap r megjelen t se
                TextBlock price = new TextBlock
                {
                    Text = $"Alap r: ${pizza.BasePrice}",
                    FontSize = 14,
                    TextAlignment = TextAlignment.Center,
                    Margin = new Thickness(0, 10, 0, 10)
                };

                // Kos rhoz ad s gomb
                Button addButton = new Button
                {
                    Content = "Kos rba",
                    Width = 100,
                    Height = 30,
                    Tag = pizza
                };
                addButton.Click += AddButton_Click;

                stack.Children.Add(name);
                stack.Children.Add(price);
                stack.Children.Add(addButton);

                border.Child = stack;
                PizzaWrapPanel.Children.Add(border);
            }
        }

        private void AddButton_Click(object sender, RoutedEventArgs
    e)
        {
            Button btn = sender as Button;
            Pizza selectedPizza = btn.Tag as Pizza;

            // Kl nozzuk a pizz t , hogy minden kos ri elemnek
    saj t toppingjai legyenek
            Pizza pizzaToAdd = new Pizza(selectedPizza.Name,
    selectedPizza.BasePrice);
            CurrentCart.AddPizza(pizzaToAdd);
        }

        private void ModifyToppings_Click(object sender,
```

```
    RoutedEventArgs e)
120      {
121          if (CurrentCart.Pizzas.Count == 0)
122          {
123              MessageBox.Show("A kosarad  res . K rlek adj
    hozz  pizz kat a toppingok m dos t sa el tt.", " res
    Kos r", MessageBoxButton.OK, MessageBoxImage.Warning);
124              return;
125          }
126
127          // Felugr  ablak a pizza kiv laszt s hoz , amelyet
    m dos tani szeretn l
128          SelectPizzaWindow selectPizzaWindow = new
    SelectPizzaWindow(CurrentCart.Pizzas);
129          if (selectPizzaWindow.ShowDialog() == true)
130          {
131              Pizza selectedPizza = selectPizzaWindow.
    SelectedPizza;
132              if (selectedPizza != null)
133              {
134                  // Konvert ljuk a toppings kollekci t List<
    Topping>-ra
135                  ToppingModificationWindow toppingWindow = new
    ToppingModificationWindow(AvailableToppings, selectedPizza.
    Toppings.ToList());
136                  if (toppingWindow.ShowDialog() == true)
137                  {
138                      // Toppings m dos t sa a megl v
    ObservableCollection-ben
139                      selectedPizza.Toppings.Clear();
140                      foreach (var topping in toppingWindow.
    SelectedToppings)
141                      {
142                          selectedPizza.Toppings.Add(topping);
143                      }
144                      MessageBox.Show("A toppingok sikeresen
    friss tve!", "Siker", MessageBoxButton.OK, MessageBoxImage.
    Information);
145                      // Friss tj k a kos r megjelen t s t
146                      CartListBox.Items.Refresh();
147                  }
148              }
149          }
150      }
151
152      private void PlaceOrder_Click(object sender,
    RoutedEventArgs e)
153      {
154          if (CurrentCart.Pizzas.Count == 0)
155          {
156              MessageBox.Show("A kosarad  res . K rlek adj
    hozz  pizz kat a rendel s lead sa el tt.", " res  Kos r",
    MessageBoxButton.OK, MessageBoxImage.Warning);
157              return;
158          }
159
```

```
160             PaymentWindow paymentWindow = new PaymentWindow(
    CurrentCart);
161             if (paymentWindow.ShowDialog() == true)
162             {
163                 // Kos r    rtse    sikeres fizet s  ut n
164                 CurrentCart.Pizzas.Clear();
165                 MessageBox.Show("K  sz  nj  k  a rendel st!", "
    Rendel s Leadva", MessageBoxButton.OK, MessageBoxImage.
    Information);
166             }
167         }
168 
169         private void Checkout_Click(object sender, RoutedEventArgs
    e)
170         {
171             // Ugyanaz, mint a PlaceOrder_Click
172             PlaceOrder_Click(sender, e);
173         }
174     }
175 }
```

Listing 5: Views/MainWindow.xaml.cs

6.2 Magyarázat

1. **Sor 1-6**: A szükséges névterek importálása, amelyek lehetővé teszik a modell osztályok használatát, valamint a LINQ és WPF komponensek használatát.

2. **Sor 8**: A `namespace PizzaOrderingApp` deklaráció meghatározza azt a névteret, amelyben az alkalmazás nézetei találhatók.

3. **Sor 10**: A `public partial class MainWindow : Window` osztálydefiníció létrehozza a `MainWindow` osztályt, amely a fő ablakot reprezentálja.

4. **Sor 12-15**: Privát mezők deklarálása:

   - `Pizzas`: A rendelkezésre álló pizzák listája.
   - `AvailableToppings`: A rendelkezésre álló toppingok listája.
   - `CurrentCart`: A felhasználó aktuális kosara.

5. **Sor 17-24**: A `MainWindow` konstruktor:

   - `InitializeComponent();`: Inicializálja a komponenseket a XAML fájlból.
   - `InitializeData();`: Inicializálja a toppingokat és a pizzákat.
   - `DisplayPizzas();`: Megjeleníti a pizzákat a felhasználói felületen.
   - `CurrentCart = new Cart();`: Létrehozza a kosár objektumot.
   - `this.DataContext = CurrentCart;`: Beállítja a `DataContext`-et a kosárra, hogy a bindingok működjenek.

6. **Sor 26-43**: `InitializeData` metódus:

- `AvailableToppings = new List<Topping> { ... };`: Inicializálja a rendelkezésre álló toppingok listáját.
- `Pizzas = new List<Pizza> { ... };`: Inicializálja a rendelkezésre álló pizzák listáját.

7. **Sor 45-71**: `DisplayPizzas` metódus:
   - `foreach (var pizza in Pizzas) { ... }`: Végigiterál a pizzák listáján és létrehoz egy UI elemet minden pizzához.
   - `Border`: Készít egy keretet a pizza információknak.
   - `StackPanel`: Függőleges elrendezést biztosít a pizza adatoknak.
   - `TextBlock`: Megjeleníti a pizza nevét.
   - `TextBlock`: Megjeleníti a pizza alapárát.
   - `Button`: Gomb a pizzák kosárba helyezéséhez, amely az `AddButton_Click` eseménykezelőt hívja meg.
8. **Sor 73-82**: `AddButton_Click` eseménykezelő:
   - `Button btn = sender as Button;`: A kattintott gomb objektumának lekérése.
   - `Pizza selectedPizza = btn.Tag as Pizza;`: A gomb `Tag` tulajdonságából lekéri a kiválasztott pizzát.
   - `Pizza pizzaToAdd = new Pizza(selectedPizza.Name, selectedPizza.BasePrice);` Klónozza a pizzát, hogy minden kosári elemnek saját toppingjai legyenek.
   - `CurrentCart.AddPizza(pizzaToAdd);`: Hozzáadja a pizzát a kosárhoz.
9. **Sor 84-107**: `ModifyToppings_Click` eseménykezelő:
   - Ellenőrzi, hogy a kosár üres-e. Ha igen, figyelmeztetést jelenít meg.
   - `SelectPizzaWindow`: Megnyit egy ablakot, ahol a felhasználó kiválaszthatja, melyik pizzát szeretné módosítani.
   - `ToppingModificationWindow`: Megnyit egy ablakot, ahol a felhasználó hozzáadhat vagy eltávolíthat toppingokat a kiválasztott pizzáról.
   - Frissíti a kosár megjelenítését, miután a toppingok módosultak.
10. **Sor 109-125**: `PlaceOrder_Click` és `Checkout_Click` eseménykezelők:
    - Ellenőrzi, hogy a kosár üres-e. Ha igen, figyelmeztetést jelenít meg.
    - `PaymentWindow`: Megnyitja a fizetési ablakot, ahol a felhasználó megerősítheti a rendelést.
    - Ha a fizetés sikeres, üríti a kosarat és köszönetet mond a felhasználónak.

7 Views/ToppingModificationWindow.xaml

7.1 Kód

```
<Window x:Class="PizzaOrderingApp.ToppingModificationWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:PizzaOrderingApp"
        Title="Modify Toppings" Height="450" Width="350">
    <Grid>
        <StackPanel Margin="10">
            <TextBlock Text="Select Toppings" FontSize="20" Margin="0,0,0,10" HorizontalAlignment="Center"/>
            <ListBox x:Name="ToppingsListBox" SelectionMode="Multiple" Height="300" />
            <StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Margin="0,10,0,0">
                <Button Content="OK" Width="80" Margin="5" Click="OK_Click"/>
                <Button Content="Cancel" Width="80" Margin="5" Click="Cancel_Click"/>
            </StackPanel>
        </StackPanel>
    </Grid>
</Window>
```

Listing 6: Views/ToppingModificationWindow.xaml

7.2 Magyarázat

1. **Sor 1-7**: A `Window` definíciója, amely meghatározza az ablak osztályát (`PizzaOrderingApp.Top` a névtereket és az ablak címét, magasságát, valamint szélességét.

2. **Sor 9-19**: A `Grid` elrendezés létrehozása:

   - `StackPanel`: Függőleges elrendezést biztosít a tartalomnak.
   - `TextBlock`: Cím a toppingok kiválasztásához.
   - `ListBox`: Lista a rendelkezésre álló toppingok megjelenítésére, többes választási lehetőséggel.
   - `StackPanel`: Felső vagy alsó gombok elhelyezése, `OK` és `Cancel` gombokkal.

8 Views/ToppingModificationWindow.xaml.cs

8.1 Kód

```
using PizzaOrderingApp.Models;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Windows;
using System.Windows.Controls;

namespace PizzaOrderingApp
{
    public partial class ToppingModificationWindow : Window
    {
        public List<Topping> SelectedToppings { get; private set; }
        private List<Topping> AvailableToppings;
        private List<Topping> CurrentToppings;

        public ToppingModificationWindow(List<Topping>
   availableToppings, List<Topping> currentToppings)
        {
            InitializeComponent();
            AvailableToppings = availableToppings;
            CurrentToppings = currentToppings;
            PopulateToppings();
        }

        private void PopulateToppings()
        {
            foreach (var topping in AvailableToppings)
            {
                var item = new CheckBox
                {
                    Content = $"{topping.Name} (+${topping.Price})
   ",
                    Tag = topping,
                    Margin = new Thickness(5)
                };
                if (CurrentToppings.Any(t => t.Name == topping.Name
   ))
                {
                    item.IsChecked = true;
                }
                ToppingsListBox.Items.Add(item);
            }
        }

        private void OK_Click(object sender, RoutedEventArgs e)
        {
            SelectedToppings = new List<Topping>();
            foreach (CheckBox cb in ToppingsListBox.Items)
            {
                if (cb.IsChecked == true)
                {
                    SelectedToppings.Add(cb.Tag as Topping);
                }
            }
            this.DialogResult = true;
        }

        private void Cancel_Click(object sender, RoutedEventArgs e)
        {
            this.DialogResult = false;
```

```
57            }
58        }
59    }
```

Listing 7: Views/ToppingModificationWindow.xaml.cs

8.2 Magyarázat

1. **Sor 1-6**: Névterek importálása, amelyek lehetővé teszik a modell osztályok használatát, valamint a WPF komponensek használatát.

2. **Sor 8**: A `namespace PizzaOrderingApp` deklaráció meghatározza azt a névteret, amelyben az alkalmazás nézetei találhatók.

3. **Sor 10**: A `public partial class ToppingModificationWindow : Window` osztálydefiníció létrehozza a `ToppingModificationWindow` osztályt, amely a toppingok módosítására szolgáló ablakot reprezentálja.

4. **Sor 12-15**: Tulajdonságok és mezők deklarálása:

   - `SelectedToppings`: A felhasználó által kiválasztott toppingok listája.
   - `AvailableToppings`: A rendelkezésre álló toppingok listája.
   - `CurrentToppings`: A pizza aktuális toppingjainak listája.

5. **Sor 17-22**: A `ToppingModificationWindow` konstruktor:

   - `InitializeComponent();`: Inicializálja a komponenseket a XAML fájlból.
   - `AvailableToppings = availableToppings;`: Beállítja a rendelkezésre álló toppingokat.
   - `CurrentToppings = currentToppings;`: Beállítja a pizza aktuális toppingjait.
   - `PopulateToppings();`: Feltölti a `ListBox`-ot a toppingokkal.

6. **Sor 24-34**: `PopulateToppings` metódus:

   - `foreach (var topping in AvailableToppings) { ... }`: Végigiterál a rendelkezésre álló toppingokon.
   - `CheckBox`: Minden toppinghoz létrehoz egy `CheckBox`-ot, amely lehetővé teszi a felhasználó számára a kiválasztást.
   - `if (CurrentToppings.Any(t => t.Name == topping.Name)) { ... }`: Ellenőrzi, hogy a topping már hozzá lett-e adva a pizzához, és ha igen, bejelöli a `CheckBox`-ot.
   - `ToppingsListBox.Items.Add(item);`: Hozzáadja a `CheckBox`-ot a `ListBox`-hoz.

7. **Sor 36-47**: `OK_Click` és `Cancel_Click` eseménykezelők:

- `OK_Click`: Gyűjti a felhasználó által kiválasztott toppingokat, és beállítja a `SelectedToppings` tulajdonságot, majd bezárja az ablakot `DialogResult = true`-ra állítva.
- `Cancel_Click`: Bezárja az ablakot `DialogResult = false`-ra állítva.

9 Views/PaymentWindow.xaml

9.1 Kód

```
<Window x:Class="PizzaOrderingApp.PaymentWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:PizzaOrderingApp"
        Title="Payment" Height="500" Width="500">
    <Grid>
        <StackPanel Margin="20">
            <TextBlock Text="Order Summary" FontSize="24" Margin="0,0,0,20" HorizontalAlignment="Center"/>
            <ListBox x:Name="OrderDetailsListBox" Height="300" />
            <TextBlock Text="{Binding Total, StringFormat=Total Paid: ${0:0.00}}" FontSize="20" Margin="0,10,0,20" HorizontalAlignment="Center"/>
            <Button Content="Confirm Payment" Width="150" Height="40" HorizontalAlignment="Center" Click="ConfirmPayment_Click"/>
        </StackPanel>
    </Grid>
</Window>
```

Listing 8: Views/PaymentWindow.xaml

9.2 Magyarázat

1. **Sor 1-7**: A `Window` definíciója, amely meghatározza az ablak osztályát (`PizzaOrderingApp.Pay` a névtereket és az ablak címét, magasságát, valamint szélességét.

2. **Sor 9-19**: A `Grid` elrendezés létrehozása:

   - `StackPanel`: Függőleges elrendezést biztosít a tartalomnak.
   - `TextBlock`: Cím az rendelés összegzéséhez.
   - `ListBox`: A rendelés részleteinek megjelenítése.
   - `TextBlock`: A kosár teljes összegének megjelenítése.
   - `Button`: Gomb a fizetés megerősítéséhez, amely a `ConfirmPayment_Click` eseménykezelőt hívja meg.

10 Views/PaymentWindow.xaml.cs

10.1 Kód

```
1  using PizzaOrderingApp.Models;
2  using System.Windows;
3
4  namespace PizzaOrderingApp
5  {
6      public partial class PaymentWindow : Window
7      {
8          private Cart OrderCart;
9
10         public PaymentWindow(Cart cart)
11         {
12             InitializeComponent();
13             OrderCart = cart;
14             this.DataContext = OrderCart;
15             DisplayOrderSummary();
16         }
17
18         private void ConfirmPayment_Click(object sender,
       RoutedEventArgs e)
19         {
20             // Fizet s meger s t se logik ja
21             MessageBox.Show("Fizet s sikeres!", "Siker",
       MessageBoxButton.OK, MessageBoxImage.Information);
22             this.DialogResult = true;
23         }
24
25         private void DisplayOrderSummary()
26         {
27             foreach (var pizza in OrderCart.Pizzas)
28             {
29                 OrderDetailsListBox.Items.Add(pizza.ToString());
30             }
31         }
32     }
33 }
```

Listing 9: Views/PaymentWindow.xaml.cs

10.2 Magyarázat

1. **Sor 1-6**: Névterek importálása, amelyek lehetővé teszik a modell osztályok használatát, valamint a WPF komponensek használatát.

2. **Sor 8**: A `namespace PizzaOrderingApp` deklaráció meghatározza azt a névteret, amelyben az alkalmazás nézetei találhatók.

3. **Sor 10**: A `public partial class PaymentWindow : Window` osztálydefiníció létrehozza a `PaymentWindow` osztályt, amely a fizetési ablakot reprezentálja.

4. **Sor 12-15**: Privát mezők deklarálása:

   - `OrderCart`: A felhasználó aktuális kosara.

5. **Sor 17-24**: A `PaymentWindow` konstruktor:

- `InitializeComponent();`: Inicializálja a komponenseket a XAML fájlból.
- `OrderCart = cart;`: Beállítja a kosár objektumot.
- `this.DataContext = OrderCart;`: Beállítja a `DataContext`-et a kosárra, hogy a bindingok működjenek.
- `DisplayOrderSummary();`: Megjeleníti a rendelés összegzését.

6. **Sor 26-35**: `ConfirmPayment_Click` eseménykezelő:
   - Megerősíti a fizetést, megjelenít egy sikeres üzenetet, majd beállítja az ablak `DialogResult`-ját.
7. **Sor 37-43**: `DisplayOrderSummary` metódus:
   - Végigiterál a kosárban lévő pizzákon és hozzáadja azok leírását a `ListBox`-hoz.