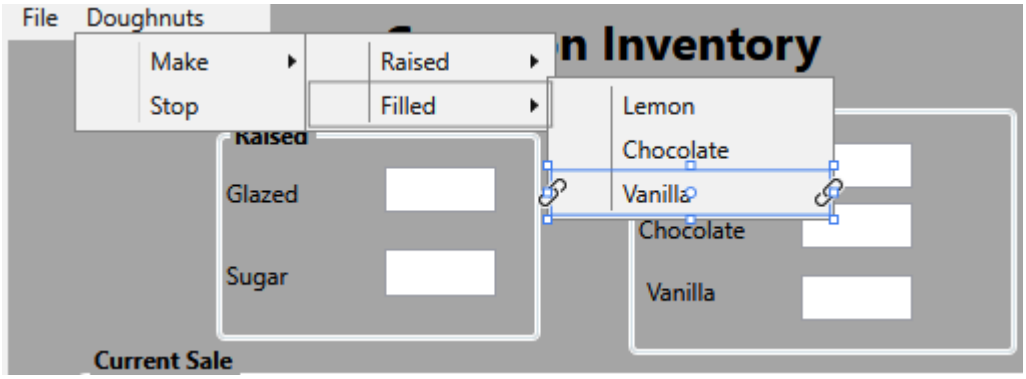
Laborator 4

I. Evenimente rutate

1. Vom continua dezvoltarea proiectului adăugând în meniul **Doughnuts > Make > Filled** următoarele elemente:



2. Pentru fiecare dintre cele 3 item-uri noi adăugate la meniu vom seta propritatea **IsCheckable=True** şi le vom asocia câte un nume astfel:

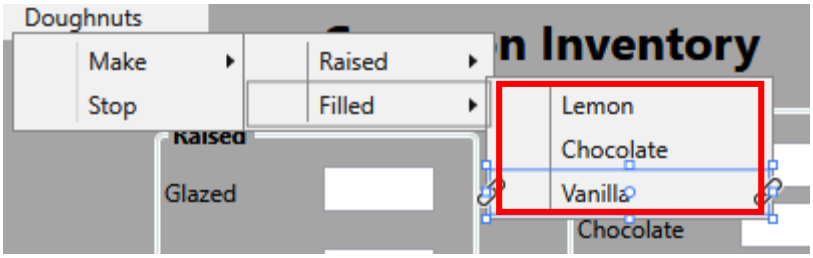
   - Lemon– **Name: lemonFilledMenuItem**
   - Chocolate – **Name: chocolateFilledMenuItem**
   - Vanilla – **Name: vanillaFilledMenuItem**

3. Tot in clasa **MainWindow.xaml.cs,** in metoda **DoughnutCompleteHandler(),** în structura switch, adăugăm următoarele cazuri:

```
//...
case DoughnutType.Lemon:
     mFilledLemon++;
     txtLemonFilled.Text = mFilledLemon.ToString(); break;
case DoughnutType.Chocolate:
     mFilledChocolate++;
     txtChocolateFilled.Text = mFilledChocolate.ToString(); break;
case DoughnutType.Vanilla:
     mFilledVanilla++;
     txtVanillaFilled.Text = mFilledVanilla.ToString(); break;
```

4. Pentru a doua categorie de gogosi (Filled) din meniu nu vom mai creea pentru fiecare MenuItem un handler de eveniment ci vom folosi evenimente rutate pentru a gestiona unitar itemii din submeniul „Filled”. Pentru a creea evenimente rutate în codul XAML aferent obiectului MenuItem pentru submeniului **„Filled”** de pe formular:

- Atasam un event handler pentru evenimentul `Click="FilledItems_Click"` – event handler care va fi apelat pentru oricare din submeniurile Lemon, Chocolate, Vanilla.




```
<MenuItem Header="Filled" Click="FilledItems_Click">
```

5. În fişierul **MainWindow.xaml.cs** se generează handler-ul pentru evenimentul rutat creat mai sus, în care scriem urmatorul cod:

Handler pentru evenimentul *Click* asociat controlului de tip MenuItem cu headerul Filled:

```
private void FilledItems_Click(object sender, RoutedEventArgs e)
  {
  string DoughnutFlavour;

  MenuItem SelectedItem = (MenuItem)e.OriginalSource ;
  DoughnutFlavour = SelectedItem.Header.ToString();

  Enum.TryParse(DoughnutFlavour, out DoughnutType myFlavour);
  myDoughnutMachine.MakeDoughnuts(myFlavour);

        }
```

6. Evenimentul va fi rutat mai apoi la obiectul de tip **Grid,** iar pentru a prinde evenimentul al nivelul Grid-ului vom atasa un event handler `FilledItemsShow_Click` pentru evenimentul `Click`. Deoarece Gridul nu poate declasansa evenimentul `Click` vom folosi un eveniment atasat `MenuItem.Click`

```
<Grid MenuItem.Click="FilledItemsShow_Click">
```

7. Dorim ca titlul formularului să se modifice in functie de tipul de gogoasa care se coace la un moment dat. Astfel în fişierul **MainWindow.xaml.cs** generam un handler `FilledItemsShow_Click`:

```
private void FilledItemsShow_Click (object sender, RoutedEventArgs e)         {
   string mesaj;

   MenuItem SelectedItem = (MenuItem)e.OriginalSource;
   mesaj = SelectedItem.Header.ToString()+" doughnuts are being cooked!";
   this.Title = mesaj;
}
```

8. Rulăm proiectul pentru a vizualiza efectul si observăm declanşarea în cascadă a evenimentelor pe arborele XAML al obiectelor. Selectam pe rand optiunile **Doughnuts >Make > Filled >Lemon** ,

Doughnuts>Make>Filled>Chocolate, Doughnuts>Make>Filled>Vanilla și observam ca acum evenimentul Click este rutat la nivelul MenuItem-ului parinte "Filled", fiind gestionat de handlerul de eveniment de la acest nivel , iar mai apoi este rutat la controlul Grid, iar cu handlerul de eveniment de la nivelul Gridului modificam titlul ferestei in functie de tipul de gogoasa selectat.

9. Pentru ca evenimentul Click al MenuItem-ului Stop sa nu fie rutat vom adauga in handler-ul lui de eveniment

```
private void stopToolStripMenuItem_Click(object sender, RoutedEventArgs e)
{
    myDoughnutMachine.Enabled = false;
    this.Title = " Virtual Doughtnuts Factory ";
    e.Handled=true;
}
```

10. In continuare vom dezvolta funcționalitățile pentru partea a doua a interfetei care va gestiona vanzarile de gogosi.

11. Pentru a asocia fiecarui tip de gogoasa un pret, definim în fișierul **MainWindow.xaml.cs** o colectie de tipul key/value astfel:

```
KeyValuePair<DoughnutType, double>[] PriceList = {
   new KeyValuePair<DoughnutType, double>(DoughnutType.Sugar, 2.5),
   new KeyValuePair<DoughnutType, double>(DoughnutType.Glazed,3),
   new KeyValuePair<DoughnutType, double>(DoughnutType.Chocolate,4.5),
   new KeyValuePair<DoughnutType, double>(DoughnutType.Vanilla,4),
   new KeyValuePair<DoughnutType, double>(DoughnutType.Lemon,3.5)
       };
```

12. Definim în fișierul **MainWindow.xaml.cs** o variabila de tip DoughnutType pentru tipul de gogoasa curent selectat :

```
DoughnutType selectedDoughnut;
```

13. Pentru a vizualiza tipurile de gogosi in obiectul CmbType, vom realiza bindingul la obiectul de tip Combobox (cbmType) cu colectia Pricelist. Astfel adaugam in handlerul de eveniment **frmMain_Loaded** al formularului urmatoarele instructiuni:

```
private void frmMain_Loaded(object sender, RoutedEventArgs e)
        {
            myDoughnutMachine = new DoughnutMachine();
            myDoughnutMachine.DoughnutComplete += new
DoughnutMachine.DoughnutCompleteDelegate(DoughnutCompleteHandler);

            cmbType.ItemsSource = PriceList;
            cmbType.DisplayMemberPath = "Key";
            cmbType.SelectedValuePath = "Value";
        }
```

14. Dorim ca la selectia unui tip de gogosa in Combobox-ul CmbType sa se afiseze automat pretul acesteia asa cum a fost stabilit in colectia PriceList. Astfel pentru evenimenul `SelectionChanged` atasam handlerul de eveniment `"cmbType_SelectionChanged"` in fisierul MainWindow.xaml astfel:

```
<ComboBox x:Name="cmbType" ...    SelectionChanged="cmbType_SelectionChanged"/>
```

15. Creem apoi handlerul de eveniment `"cmbType_SelectionChanged"` in fisierul **MainWindow.xaml.cs** cu urmatorul continut:

```
private void cmbType_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {   txtPrice.Text = cmbType.SelectedValue.ToString();
            KeyValuePair<DoughnutType, double> selectedEntry =
(KeyValuePair<DoughnutType, double>)cmbType.SelectedItem;
            selectedDoughnut = selectedEntry.Key;
}
```

16. Deoarece nu putem vinde mai multe gogosi decat avem in stoc, creem o metoda in Mainwindow.xaml.cs pentru a valida cantitatea dorita a fi cumparata din fiecare gogoasa

```
private int ValidateQuantity(DoughnutType selectedDoughnut)
        {
            int q = int.Parse(txtQuantity.Text);
            int r = 1;

            switch (selectedDoughnut)
            {
                case DoughnutType.Glazed:
                    if (q > mRaisedGlazed)
                        r = 0;
                    break;
                case DoughnutType.Sugar:
                    if (q > mRaisedSugar)
                        r = 0;
                    break;
                case DoughnutType.Chocolate:
                    if (q > mFilledChocolate)
                        r = 0;
                    break;
                case DoughnutType.Lemon:
                    if (q > mFilledLemon)
                        r = 0;
                    break;
                case DoughnutType.Vanilla:
                    if (q > mFilledVanilla)
                        r = 0;
                    break;
            }
```

```
            return r;
        }
```

17. Dorim ca la apasarea butonului AddToSale sa se adauge in listboxul lstSale o inregistrare cu urmatorul format: „Cantitate*spatiu*Tipgogoasa:PretunitarGogoasa*spatiu*Valoare”. Astfel pentru evenimenul `Click` al butonului `btnAdd` atasam handlerul de eveniment `" btnAdd_Click"` in fisierul MainWindow.xaml astfel:

```
<Button x:Name="btnAdd" ... Click="btnAdd_Click"/> >
```

18. Creem apoi handlerul de eveniment `" btnAdd_Click"` in fisierul **MainWindow.xaml.cs** cu urmatorul continut:

```
private void btnAdd_Click(object sender, RoutedEventArgs e)
        {
            if (ValidateQuantity(selectedDoughnut) > 0)
            {
                lstSale.Items.Add(txtQuantity.Text + " " + selectedDoughnut.ToString() +
":" + txtPrice.Text + " " + double.Parse(txtQuantity.Text) *
double.Parse(txtPrice.Text));
            }
            else
            {
                MessageBox.Show("Cantitatea introdusa nu este disponibila in stoc!");
            }
        }
```

19. Dorim ca in cazul in care clientul s-a razgandit la apasarea butonului RemoveItem sa se stearga din listboxul lstSale inregistrarea curenta selectata. Astfel pentru evenimenul `Click` al butonului `btnRemoveItem` atasam handlerul de eveniment `"btnRemoveItem_Click"` in fisierul MainWindow.xaml astfel:

```
<Button x:Name=" btnRemoveItem" ... Click=" btnRemoveItem_Click"/> >
```

20. Creem apoi handlerul de eveniment `"btnRemoveItem_Click"` in fisierul **MainWindow.xaml.cs** cu urmatorul continut:

```
private void btnRemoveItem_Click(object sender, RoutedEventArgs e)
        {
            lstSale.Items.Remove(lstSale.SelectedItem);
        }
```

21. Dorim ca la apasarea butonului Checkout sa se afiseze costul total in textboxul txtTotal si sa se scada din stocul de gogosi, gogosile vandute. Astfel pentru evenimenul `Click` al butonului `btnCheckOut` atasam handlerul de eveniment `" btnCheckOut_Click"` in fisierul MainWindow.xaml astfel:

```
<Button x:Name=" btnCheckOut" ... Click=" btnCheckOut_Click"/> >
```

22. Creem apoi handlerul de eveniment "btnCheckOut_Click" in fisierul **MainWindow.xaml.cs** cu urmatorul continut:

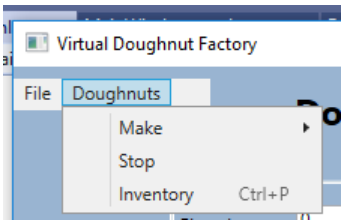
```
private void btnCheckOut_Click(object sender, RoutedEventArgs e)
        {
            txtTotal.Text = (double.Parse(txtTotal.Text) + double.Parse(txtQuantity.Text)
* double.Parse(txtPrice.Text)).ToString();

            foreach (string s in lstSale.Items)
            {
                switch (s.Substring(s.IndexOf(" ") + 1, s.IndexOf(":") - s.IndexOf(" ") -
1))
                {
                    case "Glazed":
                        mRaisedGlazed = mRaisedGlazed - Int32.Parse(s.Substring(0,
s.IndexOf(" ")));
                        txtGlazedRaised.Text = mRaisedGlazed.ToString();
                        break;
                    case "Sugar":
                        mRaisedSugar = mRaisedSugar - Int32.Parse(s.Substring(0,
s.IndexOf(" ")));
                        txtSugarRaised.Text = mRaisedSugar.ToString();
                        break;
                    case "Chocolate":
                        mFilledChocolate = mFilledChocolate - Int32.Parse(s.Substring(0,
s.IndexOf(" ")));
                        txtChocolateFilled.Text = mFilledChocolate.ToString();
                        break;
                    case "Lemon":
                        mFilledLemon = mFilledLemon - Int32.Parse(s.Substring(0,
s.IndexOf(" ")));
                        txtLemonFilled.Text = mFilledLemon.ToString();
                        break;
                    case "Vanilla":
                        mFilledVanilla = mFilledVanilla - Int32.Parse(s.Substring(0,
s.IndexOf(" ")));
                        txtVanillaFilled.Text = mFilledVanilla.ToString();
                        break;
                }
            }
        }
```

II. Comenzi în WPF

i) Comenzi predefinite

1. In MainWindow.xaml adaugam un nou MenuItem la meniul Doughnuts pentru afisarea stocului curent:




```
<MenuItem x:Name="mnuStop" Header="Stop" Click="stopToolStripMenuItem_Click"/>
        <MenuItem Header="Inventory" ></MenuItem>
    </MenuItem>
```

2. Dorim să se invoce comanda **ApplicationCommands.Print** la selectarea optiunii Inventory sau la apasarea combinatiei de taste **Ctrl+P**. Pentru aceasta asociem in codul XAML menuitem-ului Inventory comanda predefinită Print:

```
<MenuItem Header="Inventory" Command="ApplicationCommands.Print"></MenuItem>
```

3. Următorul pas este să realizăm Binding-ul între comanda şi handler-ul care să execute efectiv comanda. Astfel, în clasa de code-behind **MainWindow.xaml.cs, în constructorul clasei MainWindow()**, imediat după *InitializeComponent();* adăugăm următoarele linii de cod (marcate cu galben):

```
public MainWindow()
{
   InitializeComponent();
    //creare obiect binding pentru comanda
    CommandBinding cmd1 = new CommandBinding();
        //asociere comanda
    cmd1.Command = ApplicationCommands.Print;
    //asociem un handler
    cmd1.Executed += new ExecutedRoutedEventHandler(CtrlP_CommandHandler);
    //adaugam la colectia CommandBindings
    this.CommandBindings.Add(cmd1);
}
```

4. Apoi în aceiaşi clasa **MainWindow.xaml.cs** trebuie să implementăm metoda *CtrlP_CommandHandler* ce se va executa la invocarea comenzii ApplicationCommands.Print în aplicaţie:

```
 private void CtrlP_CommandHandler(object sender, ExecutedRoutedEventArgs e)
 {
     MessageBox.Show("You have in stock:"+mRaisedGlazed+" Glazed,"+mRaisedSugar+"
Sugar,"+mFilledLemon+" Lemon,"+mFilledChocolate+" Chocolate,"+mFilledVanilla+" Vanilla"
);
 }
```

Daca vom rula si vom invoca comanda Print fie prin selectarea optiunii din meniu fie prin apasarea combinatiei de taste Ctrl+P(aceasta este combinatia de taste implicita definita pentru

ApplicationCommands.Print), vom observa ca se executa handler-ul de comanda `CtrlP_CommandHandler` si se afiseaza stocul curent de gogosi

ii) Comenzi cu input gestures

1. Daca dorim ca pe langa combinatia de taste implicita Ctrl+P, comanda definita mai sus sa fie invocata si cu combinatia de taste Alt+I definim aceasta noua combinatie cu InputGestures. Adaugam in constructorul clasei MainWindow urmatoarea instructiune:

```
public MainWindow()
        {
            InitializeComponent();
            //creare obiect binding pentru comanda
            CommandBinding cmd1 = new CommandBinding();
            //asociere comanda
            cmd1.Command = ApplicationCommands.Print;
            //input gesture: I + Alt
            ApplicationCommands.Print.InputGestures.Add(
                                new KeyGesture(Key.I, ModifierKeys.Alt));
            //asociem un handler
            cmd1.Executed += new ExecutedRoutedEventHandler(CtrlP_CommandHandler);
            //adaugam la colectia CommandBindings
            this.CommandBindings.Add(cmd1);
        }
```

iii) Comenzi Custom

Introducem o comandă personalizată pe care o vom ataşa Meniului Doughnuts > Stop (Ctrl+S)

1. Adăugăm o nouă clasă în proiect: *StopCommand.cs* (Project>Add class)

2. Clasa o implementam intr-un subnamespace al proiectului: **CustomCommands**

```
using System.Windows.Input;

  namespace NumeStudent_PrenumeStudent_Lab2.CustomCommands
  {
      class StopCommand
      {
          private static RoutedUICommand launch_command;

          static StopCommand()
          {
              InputGestureCollection myInputGestures = new InputGestureCollection();
  myInputGestures.Add(new KeyGesture(Key.S, ModifierKeys.Control));
              launch_command = new RoutedUICommand("Launch", "Launch",
  typeof(StopCommand), myInputGestures);
          }

          public static RoutedUICommand Launch
          {
              get
```

```
            {
                return launch_command;
            }
        }
    }

}
```

3. In codul XAML al formularului, în tag-ul Window, introducem namespace-ul nou creat astfel:

```
<Window x:Name="frmMain" x:Class="NumeStudent_PrenumeStudent_Lab2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace: NumeStudent_PrenumeStudent_Lab2"
xmlns:CustomCommands="clr-namespace:NumeStudent_PrenumeStudent_Lab2.CustomCommands"
        mc:Ignorable="d"
        Title="Virtual Doughnut Factory" Height="420.5" Width="643.5"
ResizeMode="NoResize" Background="{DynamicResource {x:Static
SystemColors.ActiveBorderBrushKey}}" Loaded="frmMain_Loaded" >
```

4. Identificăm meniul Stop şi îi specificăm comanda Ctrl+S:

```
<MenuItem x:Name="mnuStop" Header="Stop" Click="stopToolStripMenuItem_Click"
InputGestureText="Ctrl+S" Command="CustomCommands:StopCommand.Launch" />
```

5. Creem un Binding în constructorul clasei **MainWindow.xaml.cs** si asociem handlerul de comanda:

```
//Doughnuts>Stop
//comanda custom
CommandBinding cmd2 = new CommandBinding();
cmd2.Command = CustomCommands.StopCommand.Launch;
cmd2.Executed += new
ExecutedRoutedEventHandler(CtrlS_CommandHandler);//asociem handler
this.CommandBindings.Add(cmd2);
```

6. Scriem în aceiaşi clasă, handler-ul CtrlS_CommandHandler:

```
private void CtrlS_CommandHandler(object sender, ExecutedRoutedEventArgs e)
{
    //handler pentru comanda Ctrl+S -> se va executa stopToolStripMenuItem_Click
    MessageBox.Show("Ctrl+S was pressed! The doughnut machine will stop!");
    this.stopToolStripMenuItem_Click(sender, e);
}
```