Laborator 3

1. Vom crea o clasa nouă numită DoughnutMachine. Pentru aceasta, in meniul Project selectăm AddClass. Introducem numele noii clase: DoughnutMachine, iar in fisierul DoughnutMachine .cs va fi generat automat urmatorul cod:

```
namespace Nume_Pren_Lab2
{
   class DoughnutMachine
   {

   }
}
```

2. In codul sursă fisierul DoughnutMachine.cs, lângă clasa DoughnutMachine adăugăm un tip enumerare numit **DoughnutType** ca si mai jos:

```
class DoughnutMachine
{

}
public enum DoughnutType
{
    Glazed,
    Sugar,
    Lemon,
    Chocolate,
    Vanilla
}
```

3. In **cadrul fisierului sursă DoughnutMachine.cs** creem o clasă numită Doughnut care reprezintă o gogoașă. Această clasă trebuie să aibă o proprietate numită flavor, una price si o proprietate time (de tip read-only) care reprezintă momentul la care a fost creat obiectul de tip Doughnut. Deoarece proprietătile de tip readonly pot să fie initializate doar in constructor, pentru această clasă va trebui să scriem un constructor care să seteze proprietățile obiectului de tip Doughnut creat: **(Atentie!!! Clasa Doughnut se pune in fisierul DoughnutMachine.cs dupa structura enum DoughnutType)**

```
class Doughnut       {
   private DoughnutType mFlavor;

   public DoughnutType Flavor
   {

        get
        {
             return mFlavor;
        }
       set
       {
```

```
            mFlavor = value;
        }
    }
```

```
    private readonly DateTime mTimeOfCreation;
    public DateTime TimeOfCreation
    {
        get
        {
            return mTimeOfCreation;
        }

    }
    public Doughnut(DoughnutType aFlavor) // constructor
    {
        mTimeOfCreation = DateTime.Now;
        mFlavor = aFlavor;
    }
}
```

4. In clasa DoughnutMachine adăugăm o proprietate care să indice tipul de gogoașă generat de instanta curentă din clasă. Proprietatea trebuie să fie de tipul enumerare DoughnutType creat la pct. 2:

```
private DoughnutType mFlavor;

public DoughnutType Flavor
{
   get
  {
      return mFlavor;
   }
    set
   {
      mFlavor = value;
    }
 }
```

5. Creem un eveniment care se va genera atunci cand crearea unei gogoși este finalizată. Pentru acest eveniment trebuie să creem si un Delegate public in clasa **DoughnutMachine**:

```
public delegate void DoughnutCompleteDelegate();
public event DoughnutCompleteDelegate DoughnutComplete;
```

6. In clasa **DoughnutMachine** creem un obiect de tipul DispatcherTimer pentru a monitoriza timpul de coacere pentru fiecare gogoașă și includem spțiul de nume **`System.Windows.Threading`**

```
DispatcherTimer doughnutTimer;
```

7. Pentru a initializa obiectul doughnutTimer adaugam metoda InitializeComponent() in clasa **DoughnutMachine**

```
private void InitializeComponent()
        {
           this.doughnutTimer = new DispatcherTimer();

           this.doughnutTimer.Tick += new System.EventHandler(this.doughnutTimer_Tick);

        }
```

8. Trebuie să ne asigurăm că această metodă este apelată la creerea unei instanțe din componenta DoughnutMachine. Deci, in componenta DoughnutMachine trebuie să existe un constructor care să apeleze metoda InitializeComponent. Adaugam in clasa DoughnutMachine urmatorul constructor:

```
public DoughnutMachine()
{
    InitializeComponent();
}
```

9. Scriem un handler de evenimente pentru **evenimentul Tick** denumit `doughnutTimer_Tick`. In clasa DoughnutMachine in care adaugam codul:

```
private void doughnutTimer_Tick(object sender, EventArgs e)
{
    Doughnut aDoughnut = new Doughnut(this.Flavor);
    DoughnutComplete();
}
```

Astfel, la fiecare tick a timerului, se va crea un nou obiect de tip Doughnut, obiect care va fi salvat in colecția **mDoughnuts** a componentei de tip DoughnutMachine. După ce obiectul de tip Doughnut este creat si adăugat in colecția mDoughnuts, se va lansa un eveniment de tip **DoughnutComplete**.

10. Adăugăm incă 2 proprietăți in clasa DoughnutMachine numite Enabled and Interval. Aceste 2 proprietăți sunt asociate timerului doughnutTimer și servesc pentru a seta proprietățile interne ale timerului.

```
public bool Enabled
{
    set
    {
        doughnutTimer.IsEnabled = value;
    }
}
public int Interval
{
   set
   {
        doughnutTimer.Interval = new TimeSpan(0, 0, value);
   }
}
```

11. Adăugăm o metodă in clasa DoughnutMachine care are rolul de a seta masina să creeze tipul corect de gogoașă, să seteze in mod corespunzător intervalul și să pornească masina. Această metodă ia ca și parametru de intrare un obiect de tipul DoughnutType:

```
public void MakeDoughnuts(DoughnutType dFlavor)
        {

            Flavor = dFlavor;
            switch (Flavor)
            {
                case DoughnutType.Glazed:Interval = 3;break;
                case DoughnutType.Sugar: Interval = 2; break;
                case DoughnutType.Lemon: Interval = 5; break;
                case DoughnutType.Chocolate: Interval = 7; break;
                case DoughnutType.Vanilla: Interval = 4; break;
            }
            doughnutTimer.Start();
        }
```

12. In cele ce urmează, vom adăuga o instanță din componenta DoughnutMachine in fisierul de codebehind al formularului nostru, in clasa **MainWindow** din fisierul **MainWindow.xaml.cs**.

In interfața utilizator vom adăuga cod care să seteze tipul de gogoașă care se crează și vom adăuga o metodă care să trateze evenimentul DoughnutComplete aruncat de către DoughnutMachine atunci când crearea unei gogoși este finalizată. De asemenea, vom echipa interfața utilizator cu o metodă care să oprească producția de gogoși. Mai intâi in clasa **MainWindow** creem o variabilă privată de tip DoughnutMachine:

```
private DoughnutMachine myDoughnutMachine;
```

13. Instanța myDoughnutMachine dorim să fie creată la incărcarea interfeței utilizator. Pentru aceasta, creem din Properties un handler de evenimente pentru evenimentul **Loaded** al formularului. Se genereaza event handle-rul in clasa de code behind a formularului in care adaugam urmatorul cod:

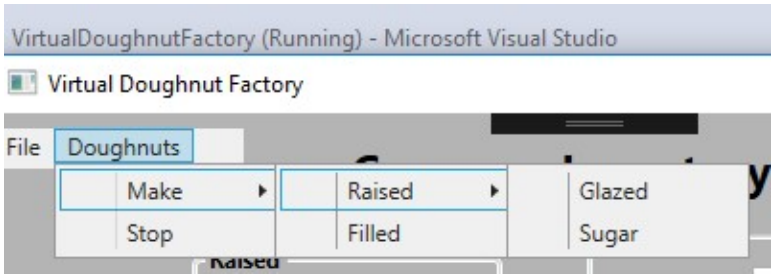
```
private void frmMain_Loaded(object sender, RoutedEventArgs e)
{
    //Tema de laborator – instantiem un obiect din clasa DougnutMachine
```

```
}
```

14. Creem variabile private in clasa de code behind a formularului în care să memorăm valori corespunzătoare numărului de gogoși create din fiecare tip de gogoașă.

```
private int mRaisedGlazed;
private int mRaisedSugar;
private int mFilledLemon;
private int mFilledChocolate;
private int mFilledVanilla;
```

15. La submeniul Make al ferestrei MainWindow adăugăm intrări in meniu pentru a permite crearea fiecărui tip de gogoașă, ca in figura de mai jos:



Codul XAML pentru meniu devine:

```
<Menu x:Name="mnuMain" HorizontalAlignment="Left" Height="21" VerticalAlignment="Top"
Width="131" Margin="0,26,0,0">
            <MenuItem Header="File">
                <MenuItem Header="Exit" Click="exitToolStripMenuItem_Click"/>
            </MenuItem>
            <MenuItem Header="_Doughnuts">
                <MenuItem Header="Make">
                    <MenuItem Header="Raised">
                        <MenuItem x:Name="glazedToolStripMenuItem" Header="Glazed"
Click="glazedToolStripMenuItem_Click" IsCheckable="True"/>
                        <MenuItem x:Name="sugarToolStripMenuItem" Header="Sugar"
IsCheckable="True" Click="sugarToolStripMenuItem_Click"/>
                    </MenuItem>
                    <MenuItem Header="Filled"/>
                </MenuItem>
                <MenuItem x:Name="mnuStop" Header="Stop"
Click="stopToolStripMenuItem_Click"/>
            </MenuItem>

        </Menu>
```

Totodata pentru fiecare meniu nou adaugat setam proprietatea **IsCheckable:True** si proprietatea **Name** pentru fiecare astfel:

- Meniul Glazed: `glazedToolStripMenuItem`
- Meniul Sugar: `sugarToolStripMenuItem`

16. Creem handlere pentru fiecare intrare de meniu creată in pasul anterior. Selectam fiecare meniu, prin punerea cursorului in linia de cod XAML aferenta acestuia. Din Properties, modul de configurare evenimente se creaza cate un handler pentru evenimentul **Click** asociat fiecarui item din meniu. Aceste handlere vor specifica obiectului de tip DoughnutMachine ce tip de gogoașă să inceapă să creeze. Pentru tipul de gogoașă selectată, intrarea respectivă de meniu va fi marcată ca si checked. In clasa **MainWindow.xaml.cs** se creaza urmatoarele event handler-e in care se scrie codul:

```
private void glazedToolStripMenuItem_Click(object sender, RoutedEventArgs e)
{
    glazedToolStripMenuItem.IsChecked = true;
   sugarToolStripMenuItem.IsChecked = false;
   myDoughnutMachine.MakeDoughnuts(DoughnutType.Glazed);
}
private void sugarToolStripMenuItem_Click(object sender, RoutedEventArgs e)
{

       // Tema de laborator – completam cu instructiunile necesare
}
```

17. Tot in clasa **MainWindow.xaml.cs** creem o metodă care să răspundă la evenimentul DoughnutComplete aruncat de către instanta de tip DoughnutMachine. Metoda setează in mod corespunzator textul intrărilor de tip TextBox din interfața grafică, precum in exemplul de mai jos:

```
private void DoughnutCompleteHandler()
{
    switch (myDoughnutMachine.Flavor)
    {
       case DoughnutType.Glazed:
            mRaisedGlazed++;
            txtGlazedRaised.Text = mRaisedGlazed.ToString();
            break;


        case DoughnutType.Sugar:
             mRaisedSugar++;
            txtSugarRaised.Text = mRaisedSugar.ToString();
            break;
           //...
    // Tema de laborator – completam cu instructiunile necesare
      }
}
```

18. In **handlerul** pentru evenimentul **Loaded** al formularului MainWindow.xaml.cs adăugăm cod pentru a lega apariția evenimentului DoughnutComplete de handlerul de evenimente prin intermediul delegate-ului:

```
myDoughnutMachine.DoughnutComplete += new
DoughnutMachine.DoughnutCompleteDelegate(DoughnutCompleteHandler);
```

19. In **meniul Doughnuts** adaugăm intrarea de meniu **Stop.**

```
<Menu x:Name="mnuMain" HorizontalAlignment="Left" Height="21" VerticalAlignment="Top"
Width="230">
            <MenuItem Header="File">
                <MenuItem Header="Exit" Click="exitToolStripMenuItem_Click"/>
            </MenuItem>
            <MenuItem Header="_Doughnuts">
<MenuItem Header="Make">
                <MenuItem Header="Raised">
                    <MenuItem x:Name="glazedToolStripMenuItem" Header="Glazed"
Click="glazedToolStripMenuItem_Click" IsCheckable="True"/>
                    <MenuItem x:Name="sugarToolStripMenuItem" Header="Sugar"
IsCheckable="True" Click="sugarToolStripMenuItem_Click"/>
                </MenuItem>
                <MenuItem Header="Filled"/>
                </MenuItem>
                <MenuItem x:Name="mnuStop" Header="Stop"/>
</MenuItem>
        </Menu>
```

20. **Apoi** in handlerul evenimentului click asociat itemului de meniu Stop 'oprim' mașina de gogoși. Adaugam astfel la evenimentul Click al noului meniu item, handler-ul de evenimente *stopToolStripMenuItem_Click* iar in clasa **MainWindow.xaml.cs**. in metoda generata, scriem urmatorul cod:

```
private void stopToolStripMenuItem_Click(object sender, RoutedEventArgs e)
{
    myDoughnutMachine.Enabled = false;
}
```

21. **Creem** un **handler de evenimente** pentru intrarea de meniu Exit. Pentru aceasta vom selecta meniul Exit utilizand fereastra XAML ca si mai sus iar in fereastra Properties, vom intra in modul de configurare a evenimentelor. La evenimentul Click vom introduce denumirea handlerului: *exitToolStripMenuItem_Click*. Se genereaza metoda in fisierul de cod behind unde adaugam urmatorul cod:

```
private void exitToolStripMenuItem_Click(object sender, RoutedEventArgs e)
{
   this.Close();
}
```

22. Vom crea handlere pentru validare a cantității introduse in obiectul de tip TextBox txtQuantity: Pentru obiectul txtQuantity cream handler-ul pentru *txtQuantity_KeyPress* evenimentul **KeyUp**:

```
private void txtQuantity_KeyPress(object sender, KeyEventArgs e)
{
   if (!(e.Key>=Key.D0 && e.Key<=Key.D9))
    {
```

```
      MessageBox.Show("Numai cifre se pot introduce!", "Input Error",MessageBoxButton.OK,
MessageBoxImage.Error);
    }
}
```