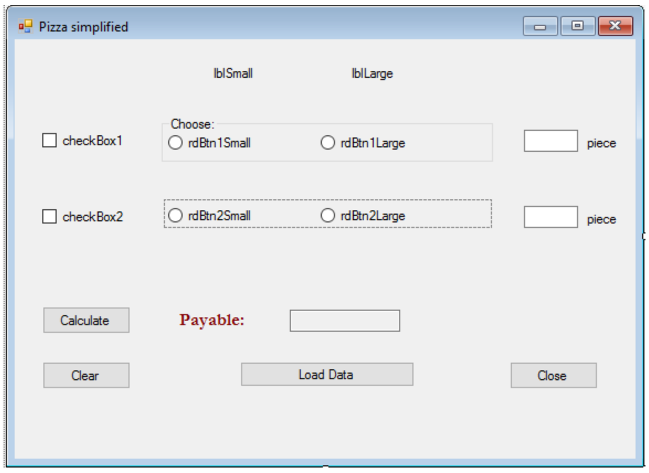
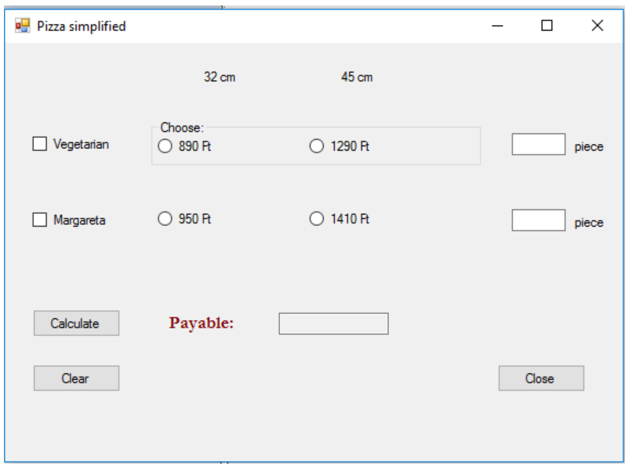
Practice 08

Exercise 1. Pizza Order program

Create an application to order different kind of pizza. Every kind of pizza can order in two sizes. After the ordering the program calculates the amount to be paid. Since the prices may often change, we store the data of pizzas in a file. When the application is started the program loads the data of pizzas from the file. The starting form contains a pushbutton to load the data from the file. After loading the data from the file the selection of pizzas becomes visible. Click the Calculate pushbutton to get the sum to be paid. Each record of the file contains three fields, name, price of small size pizza and price of large size pizza.





Before you solve this problem, create a **simplified version of it**.

The task: Now the data file stores only two kinds of pizza which we use to create a form with controls placed in design time on it.

The next figure shows the starting form and the form after reading data from the file.

Click the **Load Data** button to read data from the file selected by the user with the **OpenFileDialog** box.

Click the **Calculate** button to display the payable sum into the read only textbox.

During the calculation the next **Exception objects** should use to check the result: (Only data for selected checkbox should check.)

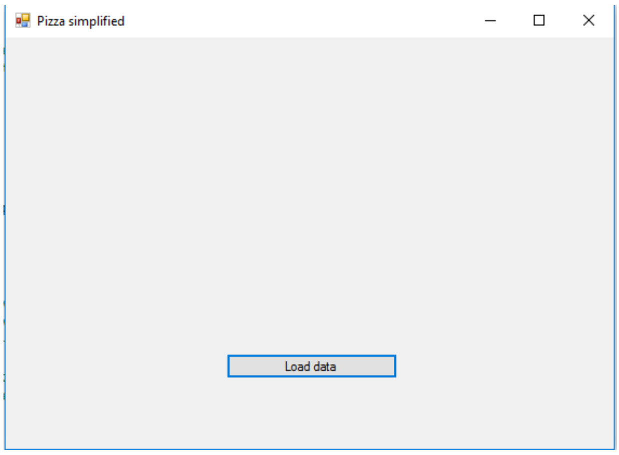
- If an error occurs using the file: **FileNotFoundException**.
- The number of pieces of pizza is missing or is in a wrong form: **FormatException**.
- Logically wrong piece value occurs (negative value): **ArgumentOutOfRangeException**.
- If a checkbox is selected, but not selected any radio button for it: **MissingFieldException**.

The file handling errors can be managed easier using properties of OpenFileDialog component.

The **Clear** button clear the content of all textboxes and set the value of the checkboxes and the radio buttons to *Unchecked*.

First variation of the solution:

a.) The starting form will be the same as you can see in the figure.



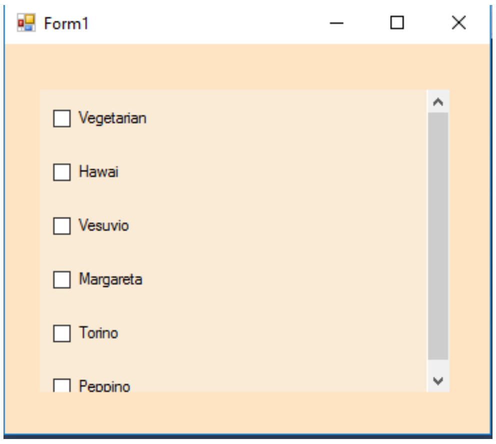
Help:

The visible property of the controls should modify.

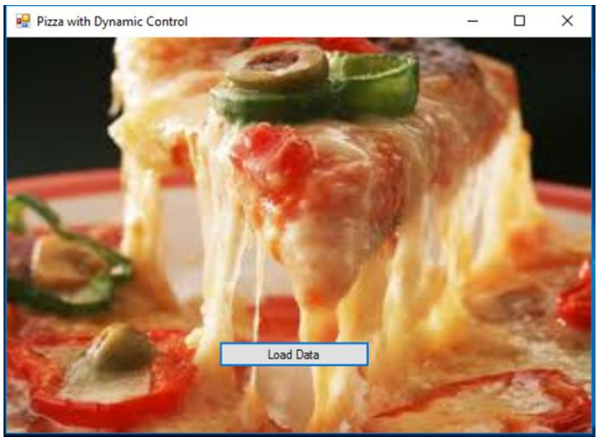
b.) Another small task:

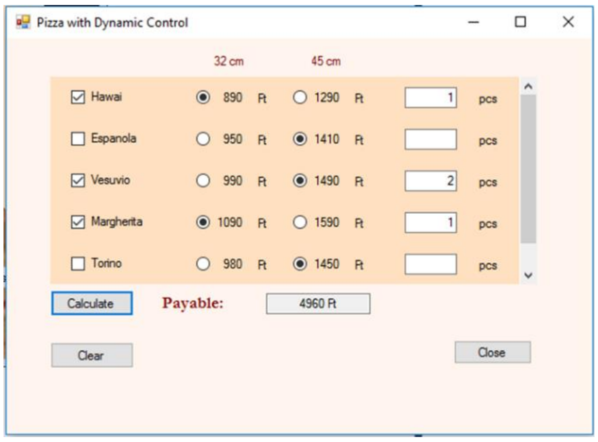
According to the content of *pizzas.txt* create the same form as you can see in the next figure:

The control elements on the central panel are placed from the code, programmatically.



c.) Let's solve the original problem:





	32 cm		45 cm			
✓ Hawai	● 890	Ft	○ 1290	Ft	1	pcs
☐ Espanola	○ 950	Ft	● 1410	Ft		pcs
✓ Vesuvio	○ 990	Ft	● 1490	Ft	2	pcs
✓ Margherita	● 1090	Ft	○ 1590	Ft	1	pcs
☐ Torino	○ 980	Ft	● 1450	Ft		pcs

Payable: 4960 Ft

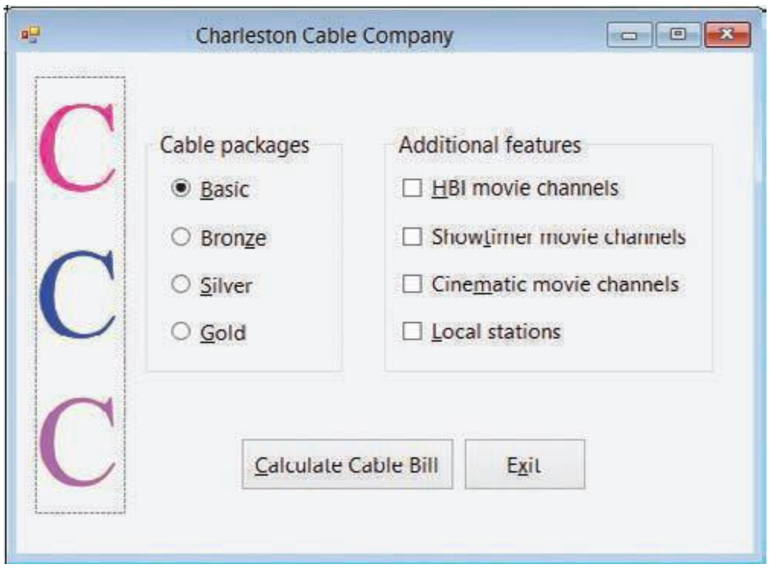
In the data file there are more data now. If there are many data in a file, we read the data until the end of stream. We cannot place the controls in design time in this case, therefore we should place controls dynamically.

When the program starts a background image and a button with Load Data capture is visible in the form. After reading the data the form will contains the controls which you can see in the second figure.

The controls needed place to a central panel which is scrollable. Place the controls to the panel programmatically.

Exercise 2 Cable Company

Create an application for the Charleston Cable Company. The application calculates and displays a customer's monthly cable bill, which is based on the information shown in the table below. The application's starting Form is shown in the Figure. The Form provides radio buttons and check boxes for selecting the cable package and additional features, respectively.



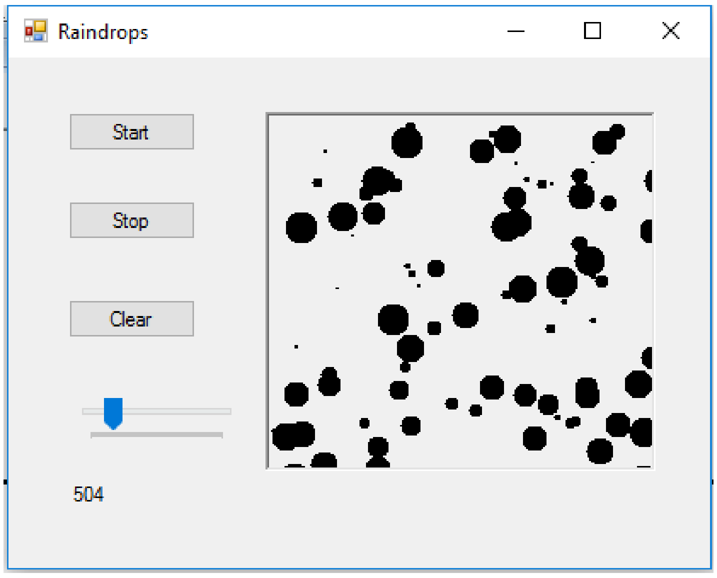
After the user chosen the cable package and additional feature and clicked the Calculate Cable Bill button a read only textbox becomes visible and shows the calculated bill value.

Packages/Additional features	**Charge**
Basic	39.99
Bronze	44.99
Silver	59.99
Gold	74.99
HBI movie channels	10.00
Showtimer movie channels	11.50
Cinematic movie channels	12.00
Local stations	5.00

The data should store in a text file and should read when the program is starts.

Exercise 3 Raindrops

Create a program (Raindrops) which simulates a sheet of paper left out in the rain. It shows random-sized drops falling, at random intervals. The random intervals can be changed via a track bar.



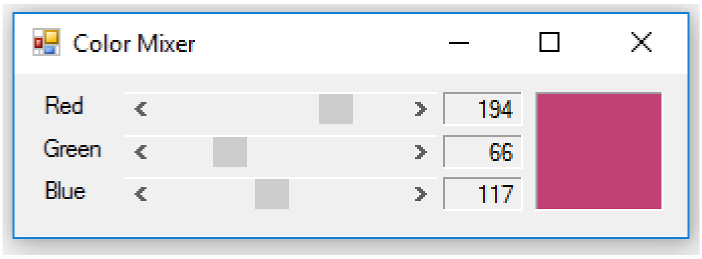
Help to the solution:

The program works by drawing a randomly sized filled circle at a random position at every tick. We also reset the timer interval to a random value controlled by the track bar. (This necessitates stopping and starting the timer.) Each time the track bar is moved, we display its current value in a label. The *Minimum* and *Maximum* track bar values were chosen by experimentation, and are 200 and 2000, set at design-time. We also made use of the *Clear* method of the picture box, which sets all the box to a specified color.

Exercise 5- Color Mixer

The scrollbar control represents a slider in which the user can move a nob to a desired position. The position of the nob on a slider determines the selected value. The Figure below shows a form with three scrollbars. This is a classic example of sliders where the user moves the three nobs to set the red, green, blue (RGB) value in selecting a color. Values for the R, G, and B range from 0 to 255, inclusive. Some of the properties we can set for a scrollbar object are the minimum and maximum range of values, small change and large change value.

The Figure shows the program in action. When you change a scrollbar's value, the label to the right shows the color component's new numeric value, and the large label on the far right shows a sample of the color with the selected red, green, and blue color components.

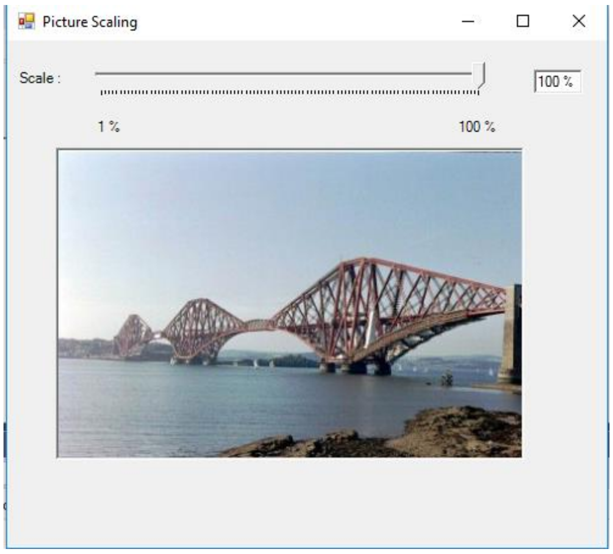


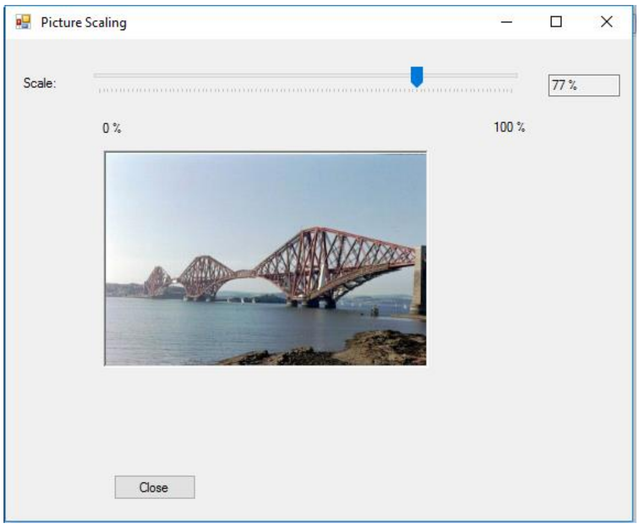
Red	194
Green	66
Blue	117

NOTE For some bizarre reason, the largest value that a user can select with a scrollbar is *Maximum - LargeChange + 1*. *If* `Maximum = 264` and `LargeChange = 10`, the **largest selectable value** is *264 – 10 + 1 = 255*, so these properties let the user select values between 0 and 255.

Exercise 6- Image Scaler

Create a Windows Form application in which the user can resize a picture dynamically. The image is in a PictureBox control. We shall do the scaling as a percentage of this original size. The user can use a trackbar to change the scaling value. The scaling value that a user can select with the trackbar is between 1 and 100 %. The next figure shows the application in action.

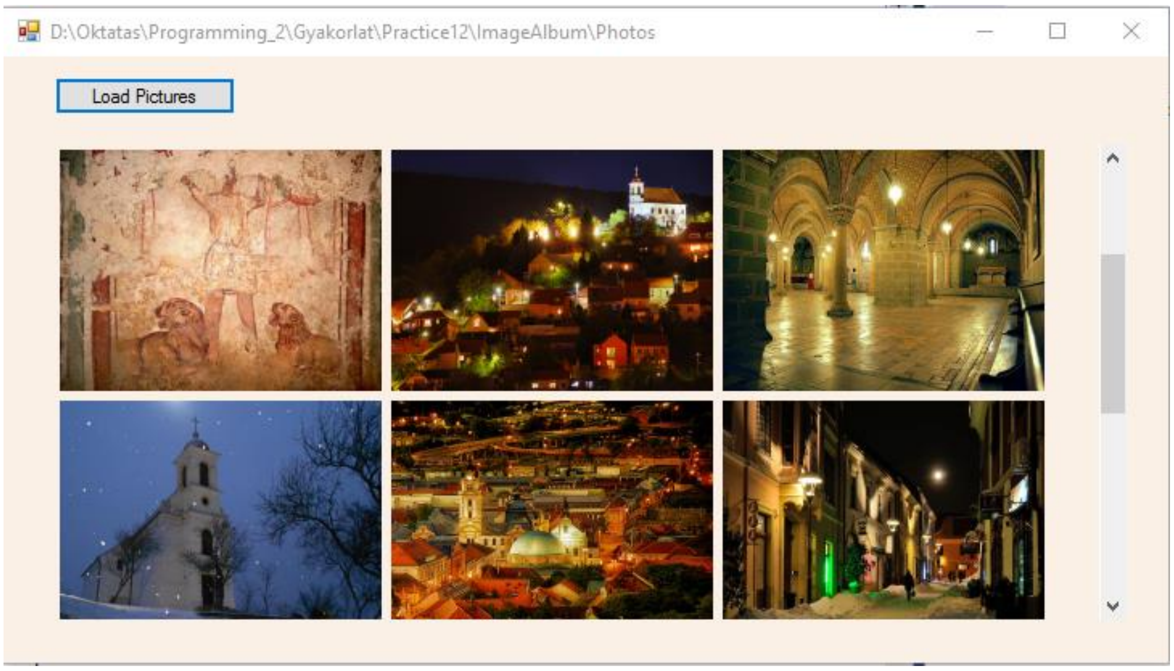




Exercise 7. Creating a simple image album (FlowLayoutPanel Control)

When you insert controls in a FlowLayoutPanel control, you do not position the controls with x and y coordinates. Instead, you treat the panel like a document that flows from one end to the other. When items reach the end of a line, they wrap around to the next line.

Create and display an image album by loading all images from a directory that is selected by the user. Because the number of images is not known at compile time, we will use a FlowLayoutPanel to show the images. The next figure shows a sample of the application while running.


D:\Oktatas\Programming_2\Gyakorlat\Practice12\ImageAlbum\Photos
Load Pictures

Hint: We use the FolderBrowserDialog control. It displays a list of folders and lets the user select one. Similar to the FileOpenDialog control, it does not appear until you call its *ShowDialog*() method. This method returns an enumerated type that lets you know which button was clicked by the user. If returns DialogResult.Cancel, the *Cancel* button was clicked. DialogResult.OK, the *Open* button was clicked.