Sofia University
Department of Mathematics and Informatics

Course : OO Programming C#.NET
Date:
Student Name:

Lab No. 9a

Submit the all C# .NET files developed to solve the problems listed below. Use comments and Modified-Hungarian notation.

Problem No. 1

Дадени са `interface ISelectable`

```
public  interface ISelectable
{

    // Indexer declaration:
    int  this[int index]
    {
        get;
        set;
    }
}
```

и class Vector със следната декларация:

```
public class Vector
        {
        // reference to an array of int
        private int[] p;
        // value of the first index of a Vector element
        private int i1;

                public Vector (){
                                // default constructor}
                public Vector (int firstIndex, int[] array ){
                                // general purpose constructor}
                public Vector (Vector  v){
                                //copy constuctor}
        }
```

class Vector трябва да представя масив (p[]), чиито елементи са от тип int. За разлика от обикновени масиви, индексирането не винаги е задължително да започва от нула. Вместо това, може да се избира стойност (i1) за индекса на първия елемент. Индексирането трябва да е безопасно, тъй че да не е възможно да се посочва елемент извън масива.

Задача:

1. Имплементирайте свойствата,конструкторите и ToString() метода на class Vector
2. Имплементирайте **`interface ISelectable`** с явно цитиране на името му **в** class Vector по отношение (**`set`** и **`get`**) на данните на class Vector

3. Дефинирайте ***exceptions*** при всеки опит за неправилен достъп до данните **`class Vector`** чрез indexer -a
4. Имплементирайте ***`IComparable`*** и операторите за сравнение ***`(==, !=)`***. (предефинирайте също методите ***`Equals(), GetHashCode()`***)
5. Предефинирайте онаследеният метод ToString() за **class Vector**
6. Предефинирайте **explicit** преобразуване на ***`int[]`*** в ***`Vector`***, както и, оператора **implicit** за преобразуване от ***`Vector в int[]`***
7. Създайте проект Class Library с **`interface ISelectable и class Vector`**
8. Създайте нов проект с конзолно приложение и напишете разширяващи методи:

   a) Filter(Predicate<int> test) за class Vector. Методът връща ArrayList<int> от елементите на масива p, които удовлетворяват предиката test.
   b) Map(Func<int, int> map) за class Vector. Методът връща Vector с елементи, които са получени след изпълнението на map към всеки от елементите на масива рна текущия обект.

9. В конзолното приложение тествайте 1- 6 на class Vector и методите Filter и Map (8a-8b) като преди създадете обект от клас Vector с генератор на случайни числа. Дефинирайте подходящи Ламбда изрази за тестване на методите Filter и Map. Напишете филтър за извеждане на елементите на обект Vector, които са по- малки от съответния им индекс в масива, или тези, които са по- малки от средната стойност на елементите в масива.

Problem No. 2

Write a class ***`CDelegateBubbleSort`*** allowing you to sort an array of objects of any kind that are ***`Comparable`***.

```
public abstract class Comparable
{
        public bool Greater(Comparable obj);
        // this function
        // compares the this reference in the implementation class
        // with the obj reference, according to the class definition
        // of the meaning of the relation Greater
        // Use operator "is " to check for the obj reference type and
        // make an explicit type conversion (or use the operator as)
}
```

For instance, class ***`CDelegateBubbleSort`*** should be able to sort arrays of ***`Vehicles`*** or ***`Shapes`*** that are ***`Comparable`*** i.e. ***`Vehicles`*** and ***`Shapes`*** **are** ***`Comparable`***. **Write a non- static method** public ***`Comparable[] Sort (Comparable[])`***

it takes as argument an array to sort and returns the sorted array- **implement** a fast sorting method like **quicksort** or **merge** sort.

Create a Namespace (a class library) with class ***CDelegateBubbleSort.***

Test the class ***CDelegateBubbleSort*** by employing an **inheritance hierarchy** of classes **-Point- Circle- Cylinder** where the inheritance is defined as follows:

- A Point has integer coordinates x and y
- A Circle is a Point and has a radius (int)
- A Cylinder is a Circle and has height (int)

Additionally classes **Point- Circle- Cylinder** should **additionally extend class *Comparable*** as follows:

a) a **Point** object P1 is greater than another **Point** object P2, if P1.mX > P2.mX and P1.mY > P2.mY, when P1.mX = P2.mX. (for instance point (1,2) is greater than point (1,1)

b) a **Circle** object C1 is greater than another **Circle** object C2, if the center point of C1 (which is a point object) is greater than the center point of C2 (which is also a point object) and C1.mRadius > C2.mRadius, when the center point of C1 is not greater than the center point of C2

c) a **Cylinder** object C1 is greater than another **Cylinder** object C2, if the circle of C1 (which is a Circle object) is greater than the circle of C2 (which is also a Circle object) and C1.mHeight > C2.mHeight, when the circle of C1 is not greater than the circle of C2

Write a C#.NET Windows application. The application class must have a class member- a reference to an array ***arrComparable*** of type ***Comparable*** with **3** elements. **Employ *arrComparable*** to compare:

a) three **Point**s (the coordinates should be entered in text fields of a form) by assigning the **Point**s to ***arrComparable***.

b) three **Circle**s (the centers of the circles should be the three points, add three more text fields to get the user input for the radiuses of the circles) by assigning the **Circle**s to ***arrComparable***.

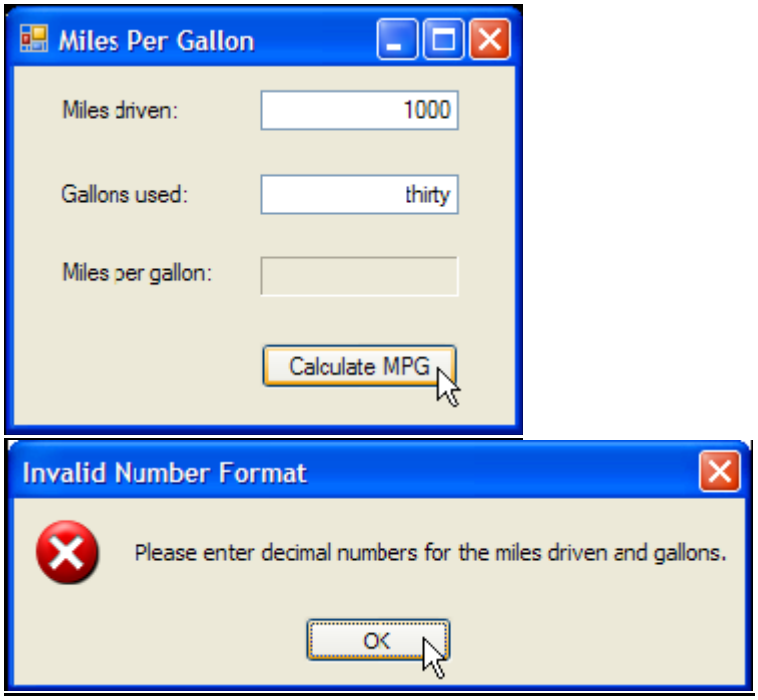
c) three **Cylinder**s (the circles of the circles should be the above defined Circle objects, add three more text fields to get the user input for the heights of the circles) by assigning the **Cylinder**s to ***arrComparable***.

d) add three buttons (***Sort Point***, ***Sort Circle, Sort Cylinder***) to **sort the above arrays- one button for each array**. By clicking on these buttons a message box should display the objects in the respective array sorted by the ***SortArray()*** member function of class ***CDelegateBubbleSort*** in ascending. You **must** use **overriding of function** *ToString()* and late binding (**polymorphism**) to display each one object in the message box to get the whole number of marks allocated for this part of the problem.

Problem No. 3

Create a GUI application that inputs miles driven and gallons used, and calculates miles per gallon. The example should use exception handling to process the ***FormatExceptions*** that occur when

converting the ***strings*** in the ***TextBoxes*** to ***doubles***. If invalid data is entered, a ***MessageBox*** should be displayed informing the user.

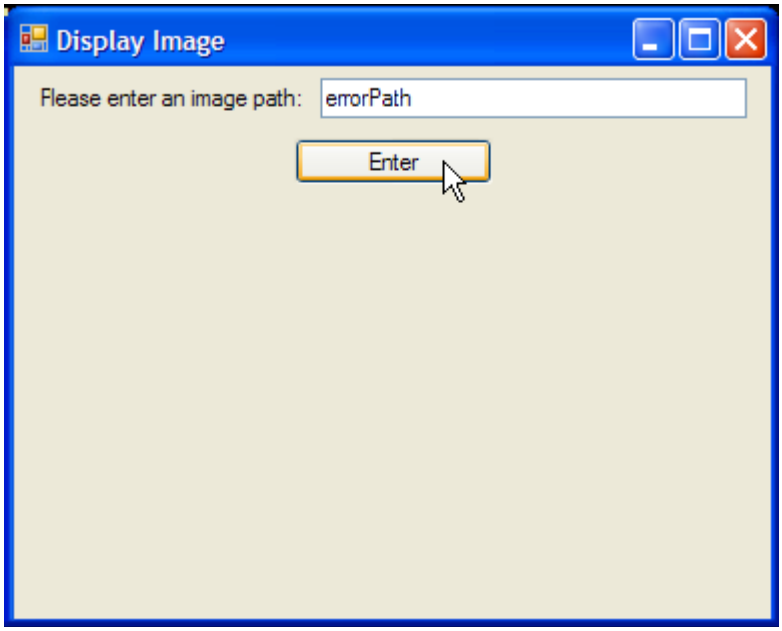


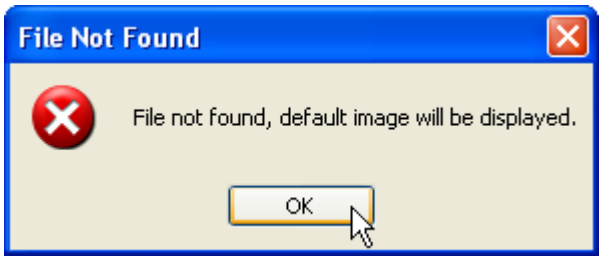
Problem No. 4

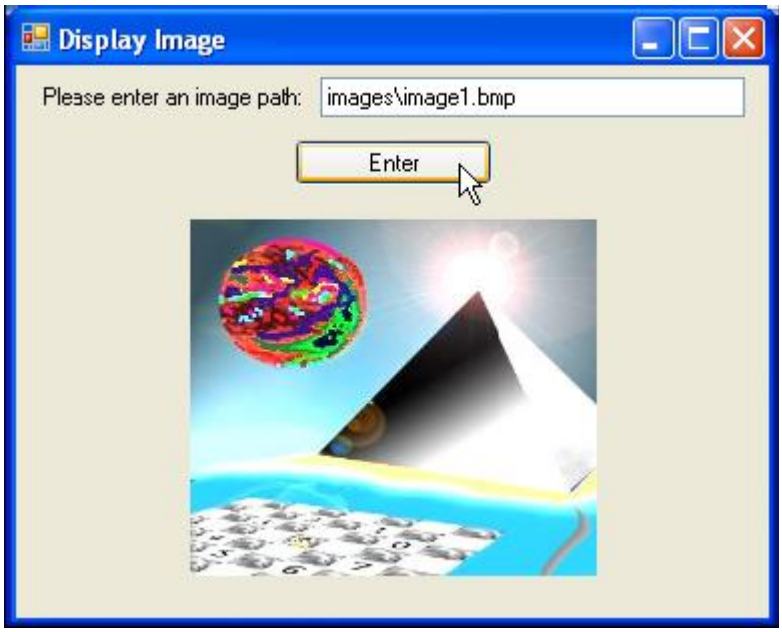
Create a Vending Machine application (see the figure below) that displays images for ***four*** snacks and corresponding ***Labels*** that indicate numbers for each snack (the snacks should be ***numbered 0–3)***. Use a ***string array*** that ***contains the names of each snack.*** The GUI should contain a ***TextBox*** in which the user specifies the number of the desired snack. When the ***Dispense Snack*** Button is clicked, the name of the selected snack (retrieved from the array) should be displayed. If the user enters a snack value not in the ***range 0–3***, an ***IndexOutOfRangeException*** will occur. Use exception handling so that whenever an ***IndexOutOfRangeException*** occurs, a ***MessageBox*** is displayed indicating the proper range of values. Also handle any possible ***FormatExceptions*** that may occur. The ***images*** used in this application are here attached. ***Problem 3 images*** directory.

Problem No. 4

Create a GUI application that displays images in a ***PictureBox.*** Allow the user to enter the path of the image in a ***TextBox*** and click a ***Button*** to display the image. If the user enters an invalid file path, a ***FileNotFoundException*** will occur. Use exception handling so that a default image will be displayed if an invalid path is entered. Whether a valid path is entered or not, clear the ***TextBox*** where the user enters input and set that ***TextBox*** to have the focus. Three images have been provided in the examples folder for this chapter in the ***Problem 4 images***\ directory. Use the image named ***image0.bmp*** as the default image. You can use the other two images to test entering a valid path. [Note: You will need to specify that you are using the ***System.IO*** namespace for this exercise. Use **method *Focus*** inherited from ***class Control*** to set the focus. To set a ***PictureBox*** to display a particular image, assign ***Image.FromFile(path)*** to its ***Image property***, where path is a string variable that specifies the image's filename relative to the current directory.]






Display Image
Please enter an image path:
images\image1.bmp
Enter