PROGRAMMING IN C#

Module 6: Classes and Methods
Module 7: Inheritance and Polymorphism
Lab Guide for Lab3

Session Objectives

In this session, you will be practicing with

- Classes and Methods
- Inheritance and Polymorphism

Part 1 – Getting started (30 minutes)

1. Creating class, object, invoking fields and methods of object

Following an application has two class Car and Program.
Class Car has 4 fields:

- *string make*
- *string model*
- *string color*
- *int yearBuilt*

and 2 methods:

- *void Start(): just prints its informations and string "Start".*
- *void Stop():just prints its informations and string "Start".*

Class Program in Main method creates some its objects and use its fields and methods.

Scan the code first, type the code, compile, run and observe the result.

1. Create class Car

```
using System;
class Car
{
      // declare the fields
      public string make;
      public string model;
      public string color;
      public int yearBuilt;
      // define the methods
      public void Start()
      {
            System.Console.WriteLine(model + " started");
      }
      public void Stop()
      {
            System.Console.WriteLine(model + " stopped");
      }
}
```

2. Create class Program

```
class Program
{
      public static void Main()
      {
            // declare a Car object reference named myCar
            Car myCar;
            // create a Car object, and assign its address to myCar
            System.Console.WriteLine("Creating a Car object and assigning "
            + "its memory location to myCar");
            myCar = new Car();

            // assign values to the Car object's fields using myCar
            myCar.make = "Toyota";
            myCar.model = "MR2";
            myCar.color = "black";
            myCar.yearBuilt = 1995;

            // display the field values using myCar
            System.Console.WriteLine("myCar details:");
            System.Console.WriteLine("myCar.make = "+ myCar.make);
            System.Console.WriteLine("myCar.model= "+myCar.model);
            System.Console.WriteLine("myCar.color = "+myCar.color);
            System.Console.WriteLine("myCar.yearBuilt=" +myCar.yearBuilt);

            // call the methods using myCar
            myCar.Start();
            myCar.Stop();

            // declare another Car object reference and
            // create another Car object
            System.Console.WriteLine("Creating another Car object
            and"+"assigning its memory location to redPorsche");
            Car redPorsche = new Car();
            redPorsche.make = "Porsche";
            redPorsche.model = "Boxster";
            redPorsche.color = "red";
            redPorsche.yearBuilt = 2000;
            System.Console.WriteLine("redPorsche is a " + redPorsche.model);
            //change the object referenced by the myCar object //reference
            to the object referenced by redPorshe
            System.Console.WriteLine("Assigning redPorsche to  myCar");
            myCar = redPorsche;
            System.Console.WriteLine("myCar details:");
            System.Console.WriteLine("myCar.make = " + myCar.make);
            System.Console.WriteLine("myCar.model = " +myCar.model);
            System.Console.WriteLine("myCar.color = " +myCar.color);
            System.Console.WriteLine("myCar.yearBuilt = "+myCar.yearBuilt);
            // assign null to myCar (myCar will no longer reference
            //an object)
            myCar = null;
            Console.ReadLine();
      }
}
```

2. Create subclass and using override method.

This application create 3 class: Window, ListBox, Button and Polimorphism. ListBox and Button are subclasses of Window. Class Window has method DrawWindow and its two subclases override it. Class Polimorphism will create some their objects and use its methods to test the polimorphism.

1. Create class Window

```
class Window
{
      // constructor takes two integers to
      // fix location on the console
      public Window(int top, int left)
      {
            this.top = top;
            this.left = left;
      }
      // simulates drawing the window
      public virtual void DrawWindow()
      {
            Console.WriteLine("Window: drawing Window at {0}, {1}",
                  top, left);
      }
      // these members are protected and thus visible
      // to derived class methods. We'll examine this
      // later in the chapter
      protected int top;
      protected int left;
}
```

2. Create class ListBox

```
class ListBox : Window
{
      // constructor adds a parameter
      public ListBox(int top, int left,string contents)
         :base(top, left) // call base constructor
      {
            listBoxContents = contents;
      }
      // an overridden version (note keyword) because in the
      // derived method we change the behavior
      public override void DrawWindow()
      {
            base.DrawWindow(); // invoke the base method
            Console.WriteLine("Writing string to the listbox:{0}",
                listBoxContents);
      }
      private string listBoxContents; // new member variable
}
```

3. Create class Button

```
class Button : Window
{
      public Button(int top, int left): base(top, left)
      {
      }
      // an overridden version (note keyword) because in the
      // derived method we change the behavior
      public override void DrawWindow()
      {
            Console.WriteLine("Drawing a button at {0}, {1}\n", top,left);
      }
}
```

4. Create class Polimorphism

```
class Polymorphism
{
      public static void Main(string[] args)
      {
            Window win = new Window(1, 2);
            ListBox lb = new ListBox(3, 4, "Stand alone list box");
            Button b = new Button(5, 6);
            win.DrawWindow();
            lb.DrawWindow();
            b.DrawWindow();
            Window[] winArray = new Window[3];
            winArray[0] = new Window(1, 2);
            winArray[1] = new ListBox(3, 4, "List box in array");
            winArray[2] = new Button(5, 6);
            for (int i = 0; i < 3; i++)
            {
                  winArray[i].DrawWindow();
            }
            Console.ReadLine();
      }
}
```

Part 2 – Workshops (30 minutes)

- Quickly look at workshops of Module 5 and Module 6.
- Try to compile, run and observe the output of sample code provided for related workshop. Discuss with your class-mate and your instructor if needed.

Part 3 – Lab Assignment (60 minutes)

Do the assignment for Module 6 and Module 7 carefully. Discuss with your class-mates and your instructor if needed.

Part 4 – Do it your self

Exercise 1:

Design and code a class named Atom that holds information about a single atom. Place your class definition in a file named Atom.cs. Include the following member functions in your design:

- boolean accept() - prompts for and accepts from standard input
  - an integer holding the atomic number,
  - a string holding the atomic symbol,
  - a string holding the full name of the atom and
  - a floating-point value holding the atomic weight.

  If any input is invalid, your function rejects that input and requests fresh data.
- void display() - displays the atomic information on standard output.

Design and code a main program that accepts information for up to 10 atomic elements and displays the atomic information in tabular format.
The program output might look something like:

```
Atomic Information
==================
Enter atomic number : 3
Enter symbol : Li
Enter full name : lithium
Enter atomic weight : 6.941
Enter atomic number : 20
Enter symbol : Ca
Enter full name : calcium
Enter atomic weight : 40.078
Enter atomic number : 30
Enter symbol : Zn
Enter full name : zinc
Enter atomic weight : 65.409
Enter atomic number : 0
No Sym Name Weight
---------------------------------
3 Li lithium 6.941
20 Ca calcium 40.078
30 Zn zinc 65.409
```

Exercise 2:

Write an ***Employee*** class to record the following attributes and behaviors for an Employee

- Declare the following instance variables
  - string firstName
  - string lastName
  - string address
  - long sin;
  - double salary
- Implement a ***constructor*** to initialize all the member variables from given parameters
- Override the ***ToString*** method to print the employee info in a good presentable format
- Define a method to calculate the bonus (salary * percentage where

percentage is given as parameter)

Write a Test program to test all the behaviors of above Employee class

note: Employer information - Social Insurance Number (SIN)