Lab Assignment 10

Course: CS202 Software Tools and Techniques for CSE
Lab Topic: Object Lifecycle, Inheritance, and Event-Driven Programming in C#
Date: 27th October 2025

Objective

This lab extends the object-oriented programming concepts introduced in lectures through hands-on tasks using classes, inheritance, and event-driven programming in C#. It focuses on building modular console applications that simulate real-world behaviours, combining delegates, events, and inheritance hierarchy control.

Learning Outcomes

By the end of this lab, students will be able to:

- ✓ Create and manage classes with encapsulation, constructors, and destructors.
- ✓ Use inheritance effectively to create hierarchical relationships and method overriding.
- ✓ Design event-driven systems using publisher-subscriber mechanisms.

Pre-Lab Requirements

- Operating System: Windows
- Software: Visual Studio 2022 (Community Edition), Visual Studio with .NET SDK
- Programming Language: C# (.NET or higher)

Lab Activities

- **Constructors and Data Control**:
  - Define a class named **Program** with the following members:
    - A private integer field data.
    - A constructor that initializes data and prints the message **"Constructor Called"**. A destructor that prints the message **"Object Destroyed"** when the object is cleaned up.[1]
    - A method `set_data(int x)` that assigns a value to data. A method `show_data()` that prints the value of data to the console.
  - In the **Main()** method:
    - Dynamically create three objects of the **Program** class.
    - Assign the values 10, 20, and 30 using `set_data()`.
    - Display the values in sequential order using `show_data()`.

[1] How to do this is left as an exercise.

Note: Please reach out to the TAs for any queries/issues.

        - When the program finishes execution, carefully observe the destructor's messages to understand when each object is destroyed.
    - ❑ Add a static counter variable in the class that tracks the number of currently **active (alive)** objects. Increment the counter in the constructor and decrement it in the destructor.
    - ❑ Display the counter value in the constructor and destructor each time they are invoked.

- ❖ **Inheritance and Method Overriding:**
    - Define a base class **Vehicle** with the following members:
        - Protected fields speed and fuel.
        - A virtual[2] method `ShowInfo()` that prints both speed and fuel.
        - A virtual method `Drive()` that decreases fuel by 5 and prints "**Vehicle is moving...**".
    - Define a derived class **Car**: Vehicle with the following members:
        - A new field **passengers.**
        - Override[3] `Drive()` to display "**Car is moving with passenger**" and decrease fuel by 10.
        - Override `ShowInfo()` to include passenger count.
    - Define another derived class **Truck: Vehicle** with the following members:
        - A new field `cargoWeight.`
        - Override `Drive()` to print "**Truck is moving with cargo**" and decrease fuel by 15.
        - Override `ShowInfo()` to display cargo weight.
    - In the `Main()` method:
        - Create one object of each **class (Vehicle, Car, Truck)** and store them in a `Vehicle[]` array[4].
        - Loop through the array, calling `Drive()` and `ShowInfo()` for each – note the behaviour changes due to overriding.
        - Use base[5]-class references to invoke the methods and observe runtime polymorphism[6].
    - In the report, explain how overriding changes the method behavior and how base-class references call the correct derived-class methods at runtime.

- ❖ **Output Reasoning (Level 0)**

---

[2] https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/virtual
[3] https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/override
[4] https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/arrays
[5] https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/base
[6] https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/polymorphism

Note: Please reach out to the TAs for any queries/issues.

- **What will be the output of the following C# code? Why?**

```
using System;
int a = 3;
int b = a++;
Console.WriteLine($"a is {+a++}, b is {-++b}");

int c = 3;
int d = ++c;
Console.WriteLine($"c is {-c--}, d is {~d}");
```

- **What will be the output of the following C# code? Why?**

```
using System;
class Program
{
    int age;
    Program() => age=age==0?age+1:age-1;
    static void Main()
    {
        int k = "010%".Replace('0','%').Length;
        Console.Write("[" + (k<<++new Program().age).ToString() + "]");
        Console.Write("[" + "010%".Split('1')[1][0] + "]");
        Console.Write("[" + "010%".Split('0')[1][0] + "]");
        Console.Write("[" + int.Parse(Convert.ToString("123".ToCharArray()[~-1])) + "]");
    }
}
```

- **What will be the output of the following C# code? Why?**

```
using System;

class Program
{
    static void Main()
    {
        int[] nums = {0, 1, 0, 3, 12};
        int pos = 0;

        for (int i = 0; i < nums.Length; i++)
        {
            if (nums[i] != 0)
            {
                int temp = nums[pos];
                nums[pos] = nums[i];
                nums[i] = temp;
                pos++;
            }
        }

        Console.WriteLine(string.Join(", ", nums));
    }
}
```

❖ Output Reasoning (Level 1)

- **What will be the output of the following C# code? Why?**

```
using System;
class Program
{
    int age;
    Program() =>[7] age=age==0?age+1:age-1;
    static void Main()
```

[7] Recall from the lecture what '=>' is.

Note: Please reach out to the TAs for any queries/issues.

```
    {
        int k = "010%".Replace('0','%').Length;
        Console.Write("[" + (k<<++new Program().age).ToString() + "]");
        Console.Write("[" + "010%".Split('1')[1][0] + "]");
        Console.Write("[" + "010%".Split('0')[1][0] + "]");
        Console.Write("[" + int.Parse(Convert.ToString("123".ToCharArray()[~-1])) + "]");
    }
}
```

➢ **What will be the output of the following C# code? Why?**

```
using System;
class Program
{
    int f;
    public static void Main(string[] args)
    {
        Console.WriteLine("run 1");
        Program p = new Program(new int()+"0".Length);
        for (int i = 0, _ = i; i < 5 && ++p.f >= 0; i++, Console.WriteLine(p.f++));
        {
            for (;p.f == 0;);
            {
                Console.WriteLine(p.f);
            }
        }

        Console.WriteLine("\nrun 2");
        p = new Program(p.f);
        Console.WriteLine(p.f);

        Console.WriteLine("\nrun 3");
        p = new Program();
        Console.WriteLine(p.f);
    }
    Program() => f = 0;
    Program(int x) => f=x;
}
```

➢ **What will be the output of the following C# code? Why?**

```
public class A
{
    public virtual void f1()
    {
        Console.WriteLine("f1");
    }
}
public class B:A
{
    public override void f1() => Console.WriteLine("f2");
}


class Program
{
    static int i=0;
    public event funcPtr handler;
    public delegate void funcPtr();
    public void destroy()
    {
        if (i == 6)
            return;
        else
        {
            Console.WriteLine(i++);
            destroy();
        }
    }
    public static void Main(string[] args)
    {
        Program p = new Program();
        p.handler += new funcPtr((new A()).f1);
        p.handler += new funcPtr((new B()).f1);
        p.handler();
```

Note: Please reach out to the TAs for any queries/issues.

```
        p.handler -= new funcPtr((new B()).f1);
        p.handler -= new funcPtr((new A()).f1);
        p?.destroy(); //check here[8] about ?. operator
        p = null;
        i = -6;
        p?.destroy();
        (new Program())?.destroy();
    }
}
```

❖ **Output Reasoning (Level 2)**

➢ **What will be the output of the following C# code? Why?**

```
public class Institute
{
    internal int i = 7;
    public Institute()
    {
        Console.Write("1");
    }
    public virtual void info()
    {
        Console.Write("2");
    }
}
public class IITGN:Institute
{
    public int i = 8;
    public IITGN()
    {
        Console.Write("3");
    }
    public IITGN(int i)
    {
        Console.Write("4");
    }
    public override void info()
    {
        Console.Write("5");
    }
}
class Program
{
    public static void Main(string[] args)
    {
        Console.Write("6");
        Institute ins1 = new Institute();
        ins1.info();
        IITGN ab101 = new IITGN(3);
        ab101 = new IITGN();
        ab101.info();
        Console.WriteLine();
        Console.WriteLine(ab101.i);
        Console.WriteLine(~(((Institute)ab101).i));
    }
}
```

➢ **What will be the output of the following C# code? Why?**

```
using System;
public class Program
{
    public delegate void mydel();
    public void fun1()
    {
        Console.WriteLine("fun1()");
    }
    public  void fun2()
```

[8] https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/member-access-operators

Note: Please reach out to the TAs for any queries/issues.

```
    {
        Console.WriteLine("fun2()");
    }
    public static void Main(string[] args)
    {
        Program p = new Program();

        mydel obj1 = new mydel(p.fun1);
        obj1 += new mydel(p.fun2);
        Console.WriteLine("run 1");
        obj1();

        mydel obj2 = new mydel(p.fun2);
        obj2 += new mydel(p.fun1);
        Console.WriteLine("run 2");
        obj2();

        obj2 -= p.fun2;
        Console.WriteLine("run 3");
        obj2();
    }
}
```

- **What will be the output of the following C# code? Why?**

```
using System;
using System.Collections[9];
public class Program
{
    int x;
    public static void Main(string[] args)
    {
        ArrayList[10] L=new ArrayList();
        L.Add(new Program());
        L.Add(new Program());
        for (int i=0;i<L.Count;i++)
            Console.WriteLine(++((Program)L[i]).x);

        L[0]=L[1];
        ((Program)L[0]).x = 202;

        for (int i=0;i<L.Count;i++)
            Console.WriteLine(((Program)L[i]).x);

        ((Program)L[0]).x = 111;
        L.RemoveAt(0);
        Console.WriteLine(L.Count);
        Console.WriteLine(((Program)L[0]).x);
    }
}
```

Resources

- Constructors in C#
- Publisher-Subscriber Pattern
- Inheritance in C# and .NET
- Lecture 10
- Lecture 11

[9] https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/collections
[10] https://learn.microsoft.com/en-us/dotnet/api/system.collections.arraylist?view=net-9.0

Note: Please reach out to the TAs for any queries/issues.