Classes and Objects II

LCSCI5202: Object Oriented Design Week 3

Waleed Iqbal – 2025/26

What is List<T>

- A generic collection in C# that stores a dynamically-sized list of elements.
- Part of the System.Collections.Generic namespace.
- The **T** represents the type of elements stored in the list (e.g., List<int> for integers, List<string> for strings).

Key Features

- Dynamic Size: Automatically resizes as elements are added or removed.
- Type Safety: Stores elements of a specific type, providing compile-time type checking.
- Access by Index: Allows fast access to elements using an index (similar to arrays).
  - Zero-based Indexing: The first element is at index 0.
- can store any data type, making it versatile (e.g., List<int>, List<string>, List<MyClass>).

Methods

- Add(item): Adds an item to the end of the list.
- Remove(item): Removes the first occurrence of the item.
- Count: Gets the number of elements in the list.
- Contains(item): Checks if the item exists in the list.
- Clear(): Removes all elements from the list.
- Insert(index, item): Inserts an item at the specified index.

Example

```
using System;
using System.Collections.Generic;

List<string> names = new List<string>();

names.Add("Alice");
names.Add("Bob");
names.Add("Charlie");

Console.WriteLine("First name: " + names[0]);

names.Remove("Bob");

Console.WriteLine("Total names: " + names.Count);
```

Keyword “this”

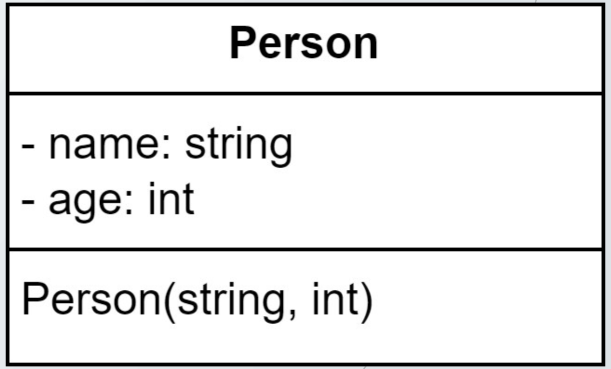
- Refers to the current instance of the class.
- Provides a way to access the class's members (fields, properties, methods) from within that class.

Keyword “this”

```
Person p = new Person("Alice", 25);

public class Person {
    string Name;
    int Age;

    public Person(string name, int age) {
        this.Name = name;
        this.Age = age;
    }
}
```



Passing Current Instance using “this” keyword

```
public class Person
{
                public string Name;
                public int Age;
public Person(string name, int age)
{
                this.Name = name;
                this.Age = age;
}
public void PrintDetails() {
                PersonPrinter printer = new PersonPrinter();
                printer.Print(this);}
}
// Simple class that works with Person objects
public class PersonPrinter {
                public void Print(Person person) {
                 Console.WriteLine($"Name: {person.Name}, Age: {person.Age}");
 }
}
```

“this” Properties

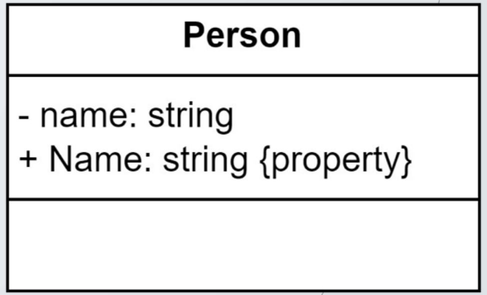
- Provide a flexible mechanism to access and modify private fields.
- Act as intermediaries between private fields and external code, offering controlled access.
- Properties help encapsulate private fields, allowing controlled access.
- Can have get or set accessors or both.
- return is expected in case of get.
- value is provided in case of set.

Properties – set and get

```
Person person = new Person();
person.Name = "Alice";
Console.WriteLine(person.Name);

public class Person {
    private string name;

    public string Name {
        get { return name; }
        set { name = value; }
    }
}
```



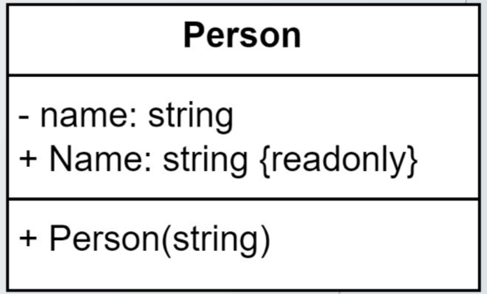
Properties – only get

```
Person person = new Person("Alice");
Console.WriteLine(person.Name);

public class Person {
    private string name;

    public string Name {
        get { return name; }
    }

    public Person(string name) {
        this.name = name;
    }
}
```



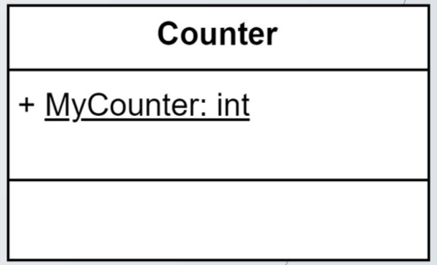
Static variables

- Belongs to the class itself, rather than to any specific instance of the class (object)
- Is shared among all instances of the class, meaning there's only one copy of the static variable for the entire class.
- Declared using the static keyword.
- Memory is allocated only once, regardless of the number of objects created.
- Changes reflect across all instances of the class.

Static variables

```
Counter.MyCounter++;
Counter.MyCounter++;
Console.WriteLine(Counter.MyCounter);


public class Counter {
    public static int MyCounter = 0;
}
```



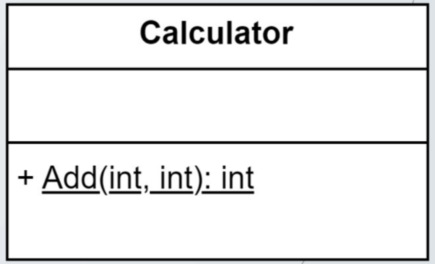
Static methods

- A method that belongs to the class itself, not to any specific instance of the class.
- Can be called without creating an object of the class.
- Declared using the static keyword.
- Cannot access instance variables or instance methods.

Static methods

```
int result = Calculator.Add(3, 7);
Console.WriteLine(result);

public class Calculator {
    public static int Add(int a, int b) {
        return a + b;
    }
}
```



Example

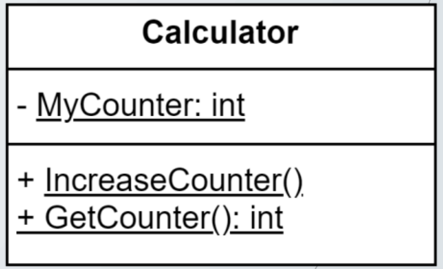
```
Counter.IncreaseCounter();
Counter.IncreaseCounter();

Console.WriteLine(Counter.GetCounter());

public class Counter {
    static int MyCounter = 0;

    public static void IncreaseCounter() {
        MyCounter++;
    }

    public static int GetCounter() {
        return MyCounter;
    }
}
```



Static classes

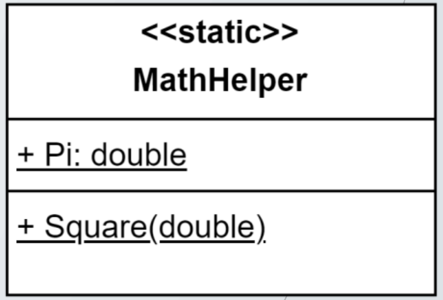
- A special class that cannot be instantiated, meaning you can't create objects of a static class.
  - You cannot use new to create an instance of a static class.
- All members (fields, methods, properties, etc.) of a static class must be declared as static.
- Single Copy in Memory.
- Used for utility or helper classes.
  - Mathematical calculations, conversion utilities, or constants.

Example

```
double area = MathHelper.Pi * MathHelper.Square(5);
Console.WriteLine(area);  // Output: 78.53975


public static class MathHelper {
    public static double Pi = 3.14159;

    public static double Square(double number) {
        return number * number;
    }
}
```



Summary

- **List<T>:** Dynamic, type-safe collections that automatically resize and provide methods like Add(), Remove(), Contains(), and Count.
- **"this" Keyword:** Refers to the current instance of a class, used in constructors to distinguish between parameters and class fields.
- **Properties:** Provide controlled access to private fields through get and set accessors, enabling data encapsulation and validation.
- **Static Variables:** Belong to the class itself rather than instances, shared across all objects with memory allocated only once.
- **Static Methods:** Class-level methods that can be called without creating objects but cannot access instance members.
- **Static Classes:** Special classes that cannot be instantiated, used for utility functions with all members required to be static.

Activity

- Create a comprehensive Math helper class
- It should have all basic functions such as Add, Subtract, Multiply, Divide, Power. It should also contain at least three algebraic functions such as Area of the Circle, circumference of the circle.
- This Math Helper class should be defined in a different file and should be accessible through that file’s workspace.