Date	Dec 11 2024	Unit	2

Introduction to C#, OOPS with C#

Introduction to C#

C# is a modern, object-oriented programming language developed by Microsoft. It is part of the .NET platform and is widely used for building Windows applications, web services, and enterprise-level solutions. Its syntax is similar to C++ and Java, making it beginner-friendly for those familiar with programming.

Reference for code editor - https://onecompiler.com/csharp

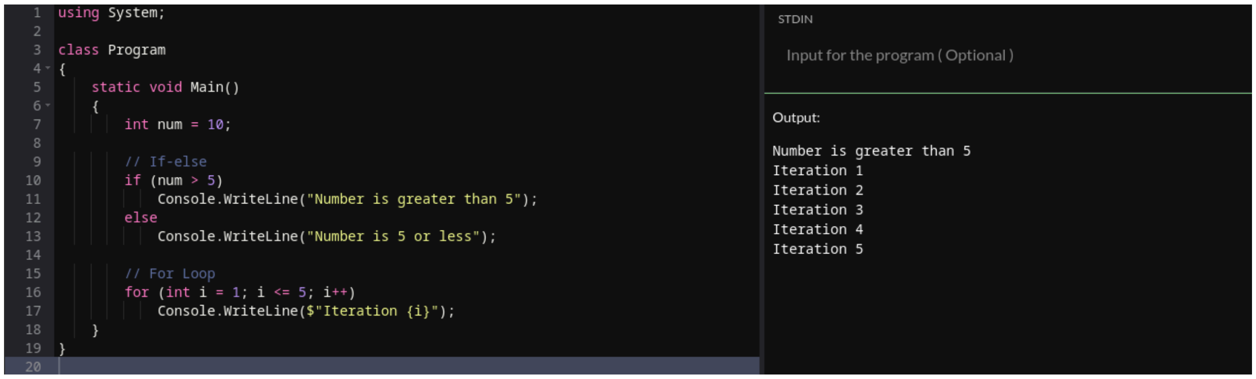
Examples

1. Literals, Variables, and Data Types

Literals are fixed values like numbers or strings. Variables store these values and are declared with specific data types.

3. Control Structures

C# supports loops and conditional statements.

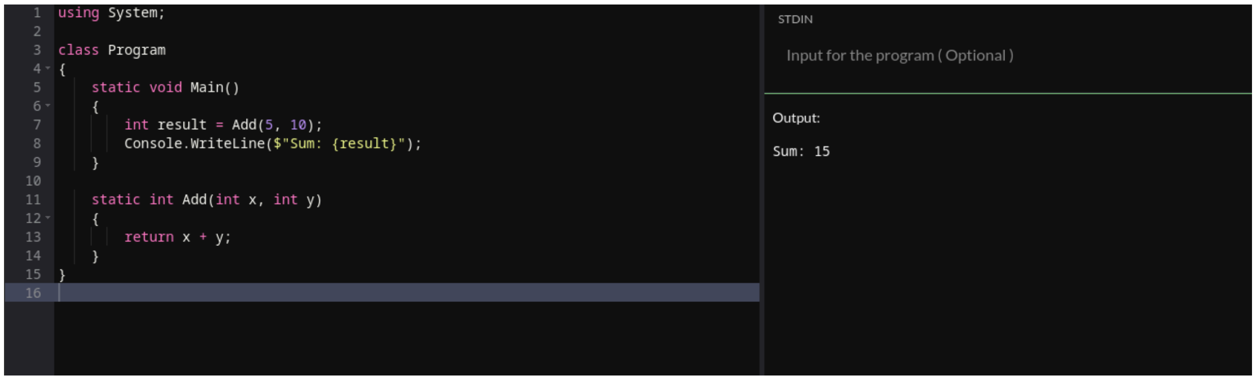


```csharp
using System;

class Program
{
    static void Main()
    {
        int num = 10;

        // If-else
        if (num > 5)
            Console.WriteLine("Number is greater than 5");
        else
            Console.WriteLine("Number is 5 or less");

        // For Loop
        for (int i = 1; i <= 5; i++)
            Console.WriteLine($"Iteration {i}");
    }
}
```

```
Number is greater than 5
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
```

4. Methods

Methods are blocks of code that perform specific tasks.

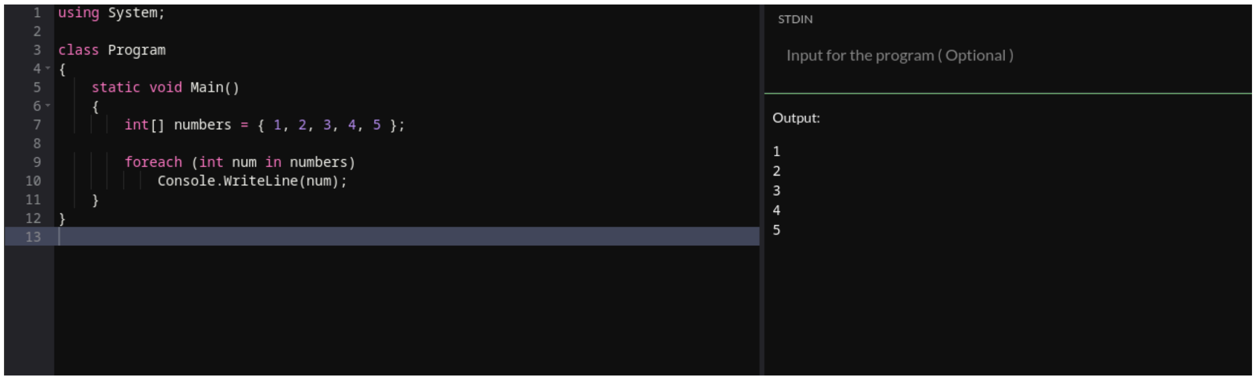


```csharp
using System;

class Program
{
    static void Main()
    {
        int result = Add(5, 10);
        Console.WriteLine($"Sum: {result}");
    }

    static int Add(int x, int y)
    {
        return x + y;
    }
}
```

```
Sum: 15
```

5. Arrays

Arrays store multiple values of the same type.

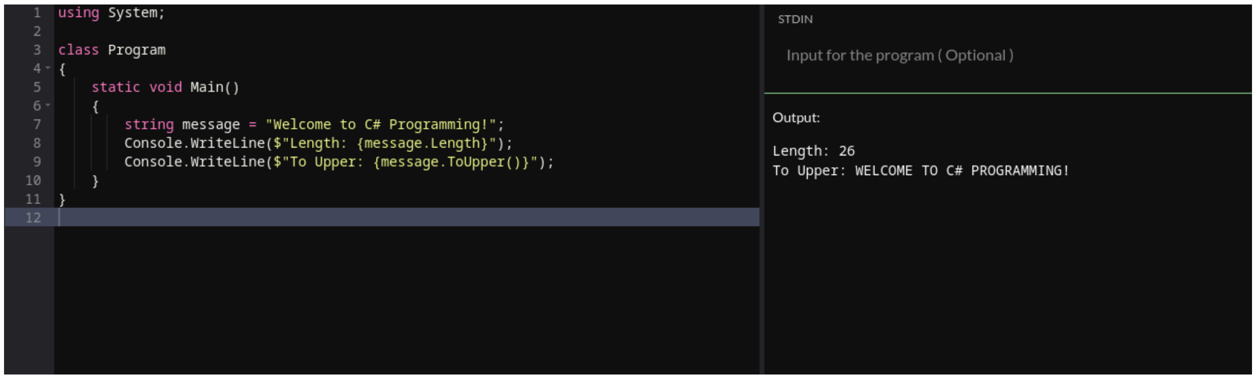


```csharp
using System;

class Program
{
    static void Main()
    {
        int[] numbers = { 1, 2, 3, 4, 5 };

        foreach (int num in numbers)
            Console.WriteLine(num);
    }
}
```

```
1
2
3
4
5
```

6. Strings

Strings are sequences of characters.



```
using System;

class Program
{
    static void Main()
    {
        string message = "Welcome to C# Programming!";
        Console.WriteLine($"Length: {message.Length}");
        Console.WriteLine($"To Upper: {message.ToUpper()}");
    }
}
```

STDIN

Input for the program (Optional)

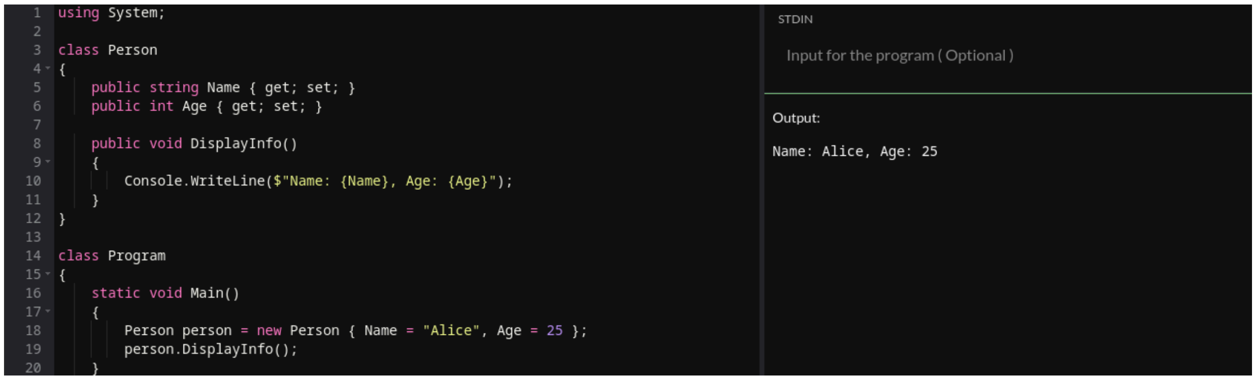
Output:

```
Length: 26
To Upper: WELCOME TO C# PROGRAMMING!
```

OOPs with C#

1. Classes and Objects

A class is a blueprint for objects, and objects are instances of classes.



```
using System;

class Person
{
    public string Name { get; set; }
    public int Age { get; set; }

    public void DisplayInfo()
    {
        Console.WriteLine($"Name: {Name}, Age: {Age}");
    }
}

class Program
{
    static void Main()
    {
        Person person = new Person { Name = "Alice", Age = 25 };
        person.DisplayInfo();
    }
```

STDIN

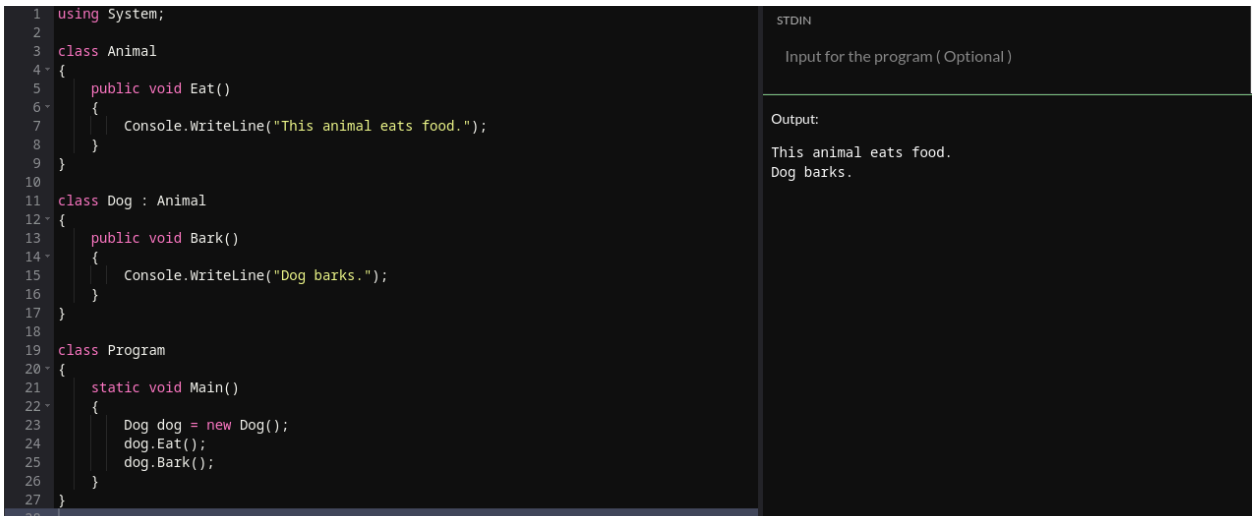
Input for the program (Optional)

Output:

```
Name: Alice, Age: 25
```

2. Inheritance

Inheritance allows a class to inherit properties and methods from another class.



```
using System;

class Animal
{
    public void Eat()
    {
        Console.WriteLine("This animal eats food.");
    }
}

class Dog : Animal
{
    public void Bark()
    {
        Console.WriteLine("Dog barks.");
    }
}

class Program
{
    static void Main()
    {
        Dog dog = new Dog();
        dog.Eat();
        dog.Bark();
    }
}
```

STDIN

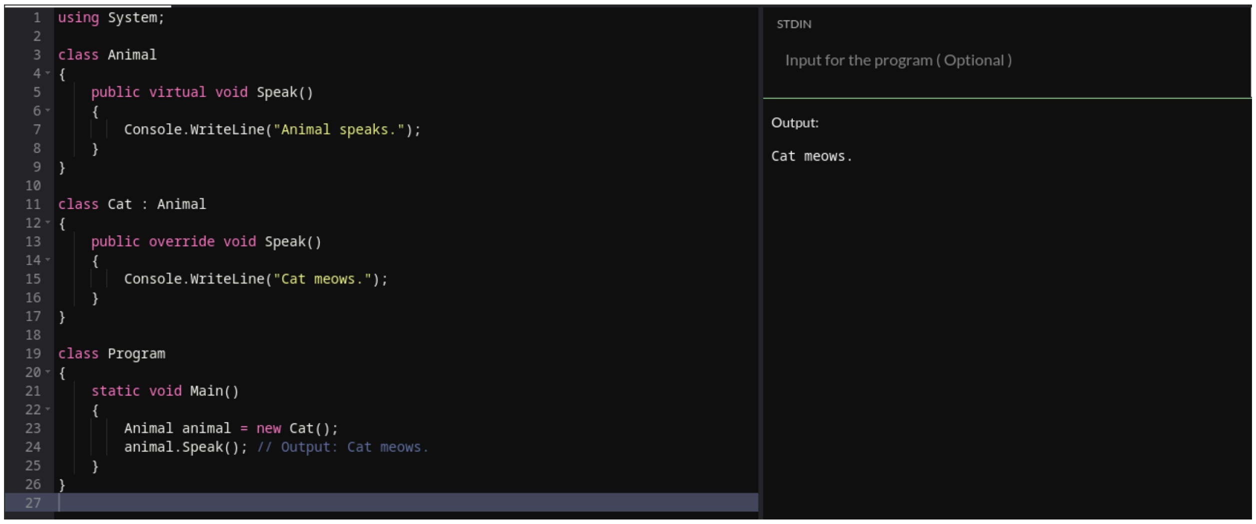
Input for the program (Optional)

Output:

```
This animal eats food.
Dog barks.
```

3. Polymorphism

Polymorphism enables methods to behave differently based on the context.



```
using System;

class Animal
{
    public virtual void Speak()
    {
        Console.WriteLine("Animal speaks.");
    }
}

class Cat : Animal
{
    public override void Speak()
    {
        Console.WriteLine("Cat meows.");
    }
}

class Program
{
    static void Main()
    {
        Animal animal = new Cat();
        animal.Speak(); // Output: Cat meows.
    }
}
```

STDIN

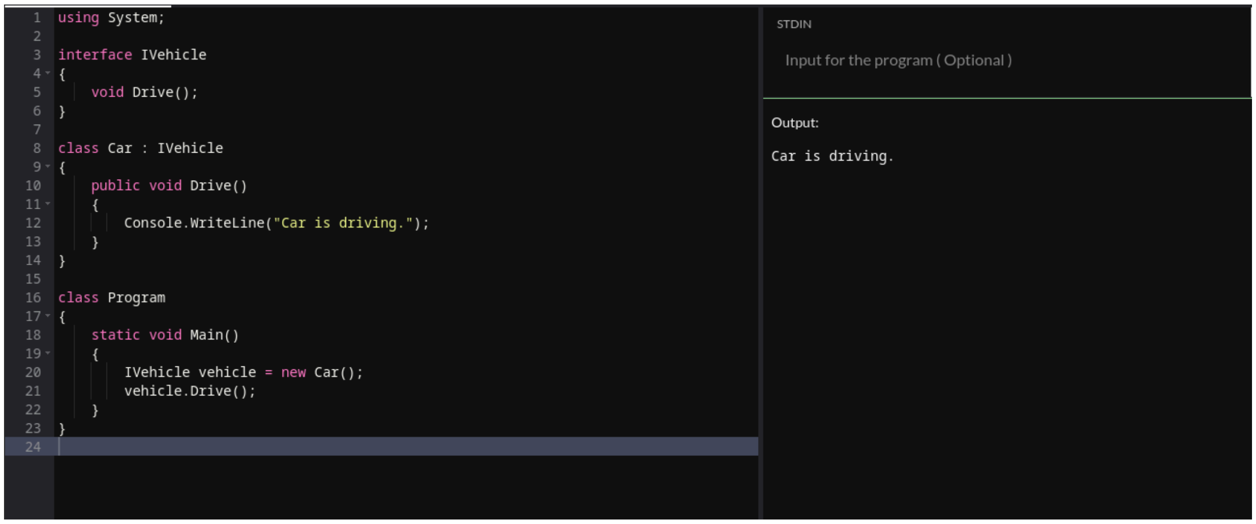
Input for the program (Optional)

Output:

```
Cat meows.
```

4. Interfaces

Interfaces define contracts that classes must implement.



STDIN

Input for the program (Optional)

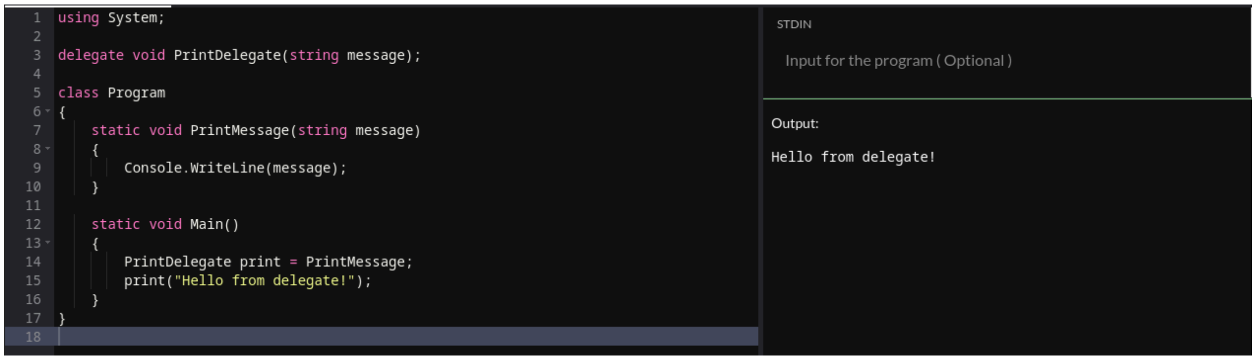
```csharp
using System;

interface IVehicle
{
    void Drive();
}

class Car : IVehicle
{
    public void Drive()
    {
        Console.WriteLine("Car is driving.");
    }
}

class Program
{
    static void Main()
    {
        IVehicle vehicle = new Car();
        vehicle.Drive();
    }
}
```

Output:

```
Car is driving.
```

5. Delegates and Events

Delegates point to methods, and events notify when something occurs.



STDIN

Input for the program (Optional)

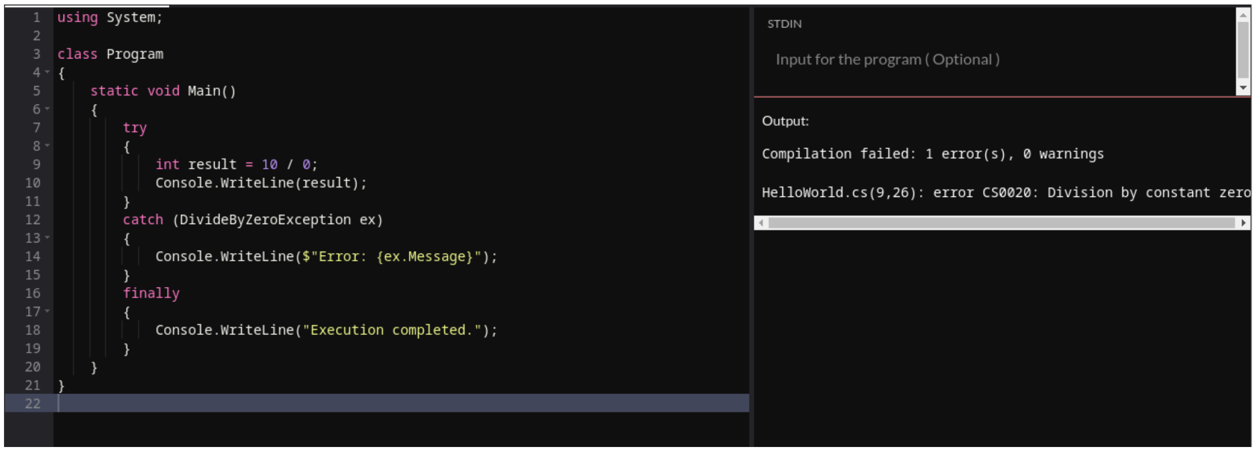
```csharp
using System;

delegate void PrintDelegate(string message);

class Program
{
    static void PrintMessage(string message)
    {
        Console.WriteLine(message);
    }

    static void Main()
    {
        PrintDelegate print = PrintMessage;
        print("Hello from delegate!");
    }
}
```

Output:

```
Hello from delegate!
```

6. Errors and Exceptions

C# uses `try-catch-finally` for error handling.



```csharp
using System;

class Program
{
    static void Main()
    {
        try
        {
            int result = 10 / 0;
            Console.WriteLine(result);
        }
        catch (DivideByZeroException ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
        finally
        {
            Console.WriteLine("Execution completed.");
        }
    }
}
```

```
Compilation failed: 1 error(s), 0 warnings

HelloWorld.cs(9,26): error CS0020: Division by constant zero
```

END