C# Cheat Sheet

1. Basic Syntax

- **Main Function**:

  ```
  csharp
  Copy code
  using System;

  class Program
  {
      static void Main(string[] args)
      {
          Console.WriteLine("Hello, World!");
      }
  }
  ```

- **Comments**:

  - Single-line comment: `// This is a comment`
  - Multi-line comment:

    ```
    csharp
    Copy code
    /* This is a
    multi-line comment */
    ```

- **Variables and Data Types**:

  ```
  csharp
  Copy code
  int x = 10;         // Integer
  double y = 5.5;     // Double (Floating-point number)
  char z = 'A';       // Character
  string name = "John";  // String
  bool isTrue = true;    // Boolean
  ```

2. Control Structures

- **If-Else**:

  ```
  csharp
  Copy code
  if (x > 10)
  {
      Console.WriteLine("x is greater than 10");
  }
  else
  {
      Console.WriteLine("x is less than or equal to 10");
  }
  ```

- **Switch-Case**:

  ```
  csharp
  Copy code
  switch (x)
  {
      case 1:
  ```

```
        Console.WriteLine("x is 1");
        break;
    case 2:
        Console.WriteLine("x is 2");
        break;
    default:
        Console.WriteLine("x is neither 1 nor 2");
        break;
}
```

- **Loops**:
  - **For loop**:

    ```
    csharp
    Copy code
    for (int i = 0; i < 5; i++)
    {
        Console.WriteLine(i);
    }
    ```

  - **While loop**:

    ```
    csharp
    Copy code
    int i = 0;
    while (i < 5)
    {
        Console.WriteLine(i);
        i++;
    }
    ```

  - **Do-While loop**:

    ```
    csharp
    Copy code
    int i = 0;
    do
    {
        Console.WriteLine(i);
        i++;
    } while (i < 5);
    ```

3. Functions (Methods)

- **Method Declaration and Definition**:

  ```
  csharp
  Copy code
  int Add(int a, int b)  // Method declaration
  {
      return a + b;       // Method definition
  }
  ```

- **Calling a Method**:

  ```
  csharp
  Copy code
  int result = Add(5, 3);  // Calls the Add method and stores the result
  Console.WriteLine(result);  // Outputs 8
  ```

- **Method Overloading**:

```
csharp
Copy code
int Add(int a, int b)
{
    return a + b;
}

double Add(double a, double b)
{
    return a + b;
}
```

- **Lambda Functions**:

```
csharp
Copy code
Func<int, int, int> add = (a, b) => a + b;
Console.WriteLine(add(5, 3));  // Outputs 8
```

4. Arrays

- **Declaration and Initialization**:

```
csharp
Copy code
int[] arr = { 1, 2, 3, 4, 5 };
```

- **Accessing Elements**:

```
csharp
Copy code
Console.WriteLine(arr[0]);  // Outputs 1
```

- **Multidimensional Arrays**:

```
csharp
Copy code
int[,] arr = { { 1, 2 }, { 3, 4 } };
Console.WriteLine(arr[1, 1]);  // Outputs 4
```

5. Strings

- **String Concatenation**:

```
csharp
Copy code
string firstName = "John";
string lastName = "Doe";
string fullName = firstName + " " + lastName;
Console.WriteLine(fullName);  // Outputs "John Doe"
```

- **String Interpolation**:

```
csharp
Copy code
string fullName = $"{firstName} {lastName}";
Console.WriteLine(fullName);  // Outputs "John Doe"
```

- **String Methods**:

```
csharp
Copy code
```

```
string text = "Hello, World!";
Console.WriteLine(text.Length);       // Length of the string
Console.WriteLine(text.ToUpper());    // Converts to uppercase
Console.WriteLine(text.Substring(7));  // Outputs "World!"
```

6. Object-Oriented Programming (OOP)

- **Class Declaration and Definition**:

```
csharp
Copy code
class MyClass
{
    public int x;

    public MyClass(int val)
    {
        x = val;
    }

    public void Display()
    {
        Console.WriteLine(x);
    }
}
```

- **Creating Objects**:

```
csharp
Copy code
MyClass obj = new MyClass(10);
obj.Display();  // Outputs 10
```

- **Inheritance**:

```
csharp
Copy code
class Animal
{
    public void Speak()
    {
        Console.WriteLine("Animal speaks");
    }
}

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

Dog d = new Dog();
d.Speak();  // Outputs "Dog barks"
```

- **Polymorphism**:

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

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

Animal animal = new Dog();
animal.Speak();  // Outputs "Dog barks"
```

- **Encapsulation**:

```
csharp
Copy code
class MyClass
{
    private int x;

    public void SetX(int val)
    {
        x = val;
    }

    public int GetX()
    {
        return x;
    }
}
```

- **Abstraction**:

```
csharp
Copy code
abstract class Animal
{
    public abstract void Speak();
}

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

7. Collections

- **List**:

```
csharp
Copy code
using System.Collections.Generic;

List<int> list = new List<int> { 1, 2, 3, 4 };
list.Add(5);
```

```
Console.WriteLine(list[2]);  // Outputs 3
```

- **Dictionary**:

```
csharp
Copy code
Dictionary<int, string> dict = new Dictionary<int, string>();
dict.Add(1, "One");
dict[2] = "Two";
Console.WriteLine(dict[1]);  // Outputs "One"
```

- **Queue**:

```
csharp
Copy code
Queue<int> queue = new Queue<int>();
queue.Enqueue(10);
queue.Enqueue(20);
Console.WriteLine(queue.Dequeue());  // Outputs 10
```

- **Stack**:

```
csharp
Copy code
Stack<int> stack = new Stack<int>();
stack.Push(10);
stack.Push(20);
Console.WriteLine(stack.Pop());  // Outputs 20
```

8. LINQ (Language Integrated Query)

- **Basic LINQ Query**:

```
csharp
Copy code
using System.Linq;

List<int> numbers = new List<int> { 1, 2, 3, 4, 5 };
var evenNumbers = from num in numbers
                  where num % 2 == 0
                  select num;

foreach (var num in evenNumbers)
{
    Console.WriteLine(num);  // Outputs 2, 4
}
```

9. Exception Handling

- **Try-Catch Block**:

```
csharp
Copy code
try
{
    int x = 10 / 0;
}
catch (DivideByZeroException ex)
{
    Console.WriteLine("Error: " + ex.Message);
}
```

- **Finally Block**:

```
csharp
Copy code
try
{
    int x = 10 / 0;
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
finally
{
    Console.WriteLine("This is always executed.");
}
```

---

10. Async and Await

- **Asynchronous Programming**:

```
csharp
Copy code
using System.Threading.Tasks;

async Task<int> FetchDataAsync()
{
    await Task.Delay(2000);  // Simulate a delay
    return 42;
}

static async Task Main(string[] args)
{
    int result = await FetchDataAsync();
    Console.WriteLine(result);  // Outputs 42
}
```