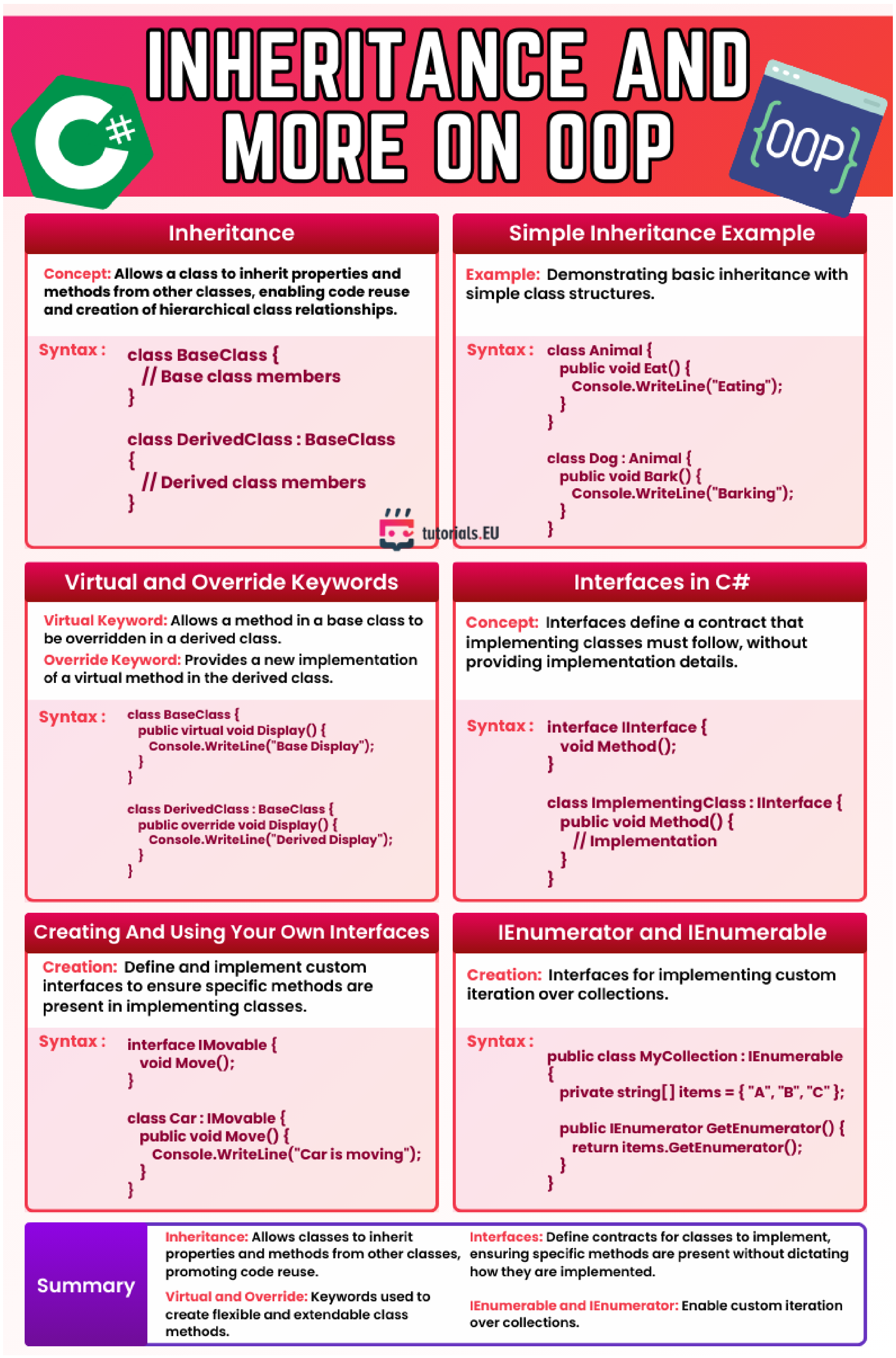
CHEATSHEET - Inheritance and more on OOP



INHERITANCE AND MORE ON OOP

Inheritance

Concept: Allows a class to inherit properties and methods from other classes, enabling code reuse and creation of hierarchical class relationships.

Syntax :

```
class BaseClass {
  // Base class members
}

class DerivedClass : BaseClass
{
  // Derived class members
}
```

Simple Inheritance Example

Example: Demonstrating basic inheritance with simple class structures.

Syntax :

```
class Animal {
  public void Eat() {
    Console.WriteLine("Eating");
  }
}

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

tutorials.EU

Virtual and Override Keywords

Virtual Keyword: Allows a method in a base class to be overridden in a derived class.

Override Keyword: Provides a new implementation of a virtual method in the derived class.

Syntax :

```
class BaseClass {
  public virtual void Display() {
    Console.WriteLine("Base Display");
  }
}

class DerivedClass : BaseClass {
  public override void Display() {
    Console.WriteLine("Derived Display");
  }
}
```

Interfaces in C#

Concept: Interfaces define a contract that implementing classes must follow, without providing implementation details.

Syntax :

```
interface IInterface {
  void Method();
}

class ImplementingClass : IInterface {
  public void Method() {
    // Implementation
  }
}
```

Creating And Using Your Own Interfaces

Creation: Define and implement custom interfaces to ensure specific methods are present in implementing classes.

Syntax :

```
interface IMovable {
  void Move();
}

class Car : IMovable {
  public void Move() {
    Console.WriteLine("Car is moving");
  }
}
```

IEnumerator and IEnumerable

Creation: Interfaces for implementing custom iteration over collections.

Syntax :

```
public class MyCollection : IEnumerable
{
  private string[] items = { "A", "B", "C" };

  public IEnumerator GetEnumerator() {
    return items.GetEnumerator();
  }
}
```

Summary

Inheritance: Allows classes to inherit properties and methods from other classes, promoting code reuse.

Virtual and Override: Keywords used to create flexible and extendable class methods.

Interfaces: Define contracts for classes to implement, ensuring specific methods are present without dictating how they are implemented.

IEnumerable and IEnumerator: Enable custom iteration over collections.