**What is a Constructor in C#?**

A **constructor** is a special method in a class that **initializes** an object when it is created. It has the same name as the class and **does not have a return type (not even void)**.

**1️ Types of Constructors in C#**

**1. Default Constructor (Parameterless)**

A constructor with no parameters. It is **automatically provided by C#** if no constructor is defined.

**✅ Example:**

class Person

{

public string Name;

public int Age;

// Default constructor

public Person()

{

Name = "Unknown";

Age = 0;

Console.WriteLine("Default Constructor Called!");

}

}

class Program

{

static void Main()

{

Person p = new Person();

Console.WriteLine($"Name: {p.Name}, Age: {p.Age}");

}

}

**2. Parameterized Constructor**

A constructor that **accepts parameters** to initialize an object with custom values.

**✅ Example:**

class Person

{

public string Name;

public int Age;

// Parameterized constructor

public Person(string name, int age)

{

Name = name;

Age = age;

}

}

class Program

{

static void Main()

{

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

Console.WriteLine($"Name: {p.Name}, Age: {p.Age}");

}

}

**3. Copy Constructor**

A constructor that **copies values from another object** of the same class.

**✅ Example:**

class Person

{

public string Name;

public int Age;

public Person(string name, int age)

{

Name = name;

Age = age;

}

// Copy Constructor

public Person(Person other)

{

Name = other.Name;

Age = other.Age;

}

}

class Program

{

static void Main()

{

Person p1 = new Person("Alice", 25);

Person p2 = new Person(p1); // Copying values

Console.WriteLine($"Name: {p2.Name}, Age: {p2.Age}");

}

}

**4. Static Constructor**

A constructor that **runs only once** when the class is first used. It is used to **initialize static members**.

**✅ Example:**

class Example

{

public static int Value;

// Static Constructor

static Example()

{

Value = 10;

Console.WriteLine("Static Constructor Called!");

}

}

class Program

{

static void Main()

{

Console.WriteLine($"Value: {Example.Value}");

}

}

📌 **Key Points:**

* No parameters are allowed.
* Runs only once per class.
* Cannot be called manually.

**5. Private Constructor**

A constructor that **prevents object instantiation from outside the class**. Useful in **Singleton Patterns**.

**✅ Example:**

class Singleton

{

private static Singleton obj;

// Private Constructor

private Singleton() { }

public static Singleton GetInstance()

{

if (obj == null)

obj =new Singleton();

return obj;

}

}

class Program

{

static void Main()

{

Singleton obj1 = Singleton.GetInstance();

Singleton obj2 = Singleton.GetInstance();

Console.WriteLine(obj1 == obj2); // True, both are the same instance

}

}

**2️⃣ Constructor Overloading**

You can define **multiple constructors** with different parameters.

**✅ Example:**

csharp

CopyEdit

class Person

{

public string Name;

public int Age;

public Person() // Default

{

Name = "Unknown";

Age = 0;

}

public Person(string name) // Overloaded constructor

{

Name = name;

Age = 18; // Default age

}

public Person(string name, int age) // Overloaded constructor

{

Name = name;

Age = age;

}

}

class Program

{

static void Main()

{

Person p1 = new Person();

Person p2 = new Person("Alice");

Person p3 = new Person("Bob", 30);

Console.WriteLine($"{p1.Name}, {p1.Age}");

Console.WriteLine($"{p2.Name}, {p2.Age}");

Console.WriteLine($"{p3.Name}, {p3.Age}");

}

}

**3️⃣ Constructor Chaining**

One constructor can **call another constructor** using this.

**✅ Example:**

csharp

CopyEdit

class Person

{

public string Name;

public int Age;

// Calls the parameterized constructor

public Person() : this("Unknown", 0) { }

public Person(string name) : this(name, 18) { }

public Person(string name, int age)

{

Name = name;

Age = age;

}

}

class Program

{

static void Main()

{

Person p1 = new Person();

Person p2 = new Person("Alice");

Person p3 = new Person("Bob", 30);

Console.WriteLine($"{p1.Name}, {p1.Age}");

Console.WriteLine($"{p2.Name}, {p2.Age}");

Console.WriteLine($"{p3.Name}, {p3.Age}");

}

}

**🔹 Summary**

| **Type** | **Purpose** |
| --- | --- |
| **Default Constructor** | Initializes with default values |
| **Parameterized Constructor** | Initializes with custom values |
| **Copy Constructor** | Creates a new object by copying another object |
| **Static Constructor** | Initializes static members, runs once |
| **Private Constructor** | Restricts instantiation, used in Singleton Pattern |
| **Constructor Overloading** | Multiple constructors with different parameters |
| **Constructor Chaining** | Calls another constructor using this |