How to write a copy constructor (C# Programming Guide)

07/20/2015 • 2 minutes to read • +4

In this article

C# doesn't provide a copy constructor for objects, but you can write one yourself.

Example

In the following example, the `Person` class defines a copy constructor that takes, as its argument, an instance of `Person`. The values of the properties of the argument are assigned to the properties of the new instance of `Person`. The code contains an alternative copy constructor that sends the `Name` and `Age` properties of the instance that you want to copy to the instance constructor of the class.

C# Copy

```
class Person
{
    // Copy constructor.
    public Person(Person previousPerson)
    {
        Name = previousPerson.Name;
        Age = previousPerson.Age;
    }
```

```
    //// Alternate copy constructor calls the instance constructor.
    //public Person(Person previousPerson)
    //    : this(previousPerson.Name, previousPerson.Age)
    //{
    //}

    // Instance constructor.
    public Person(string name, int age)
    {
        Name = name;
        Age = age;
    }

    public int Age { get; set; }

    public string Name { get; set; }

    public string Details()
    {
        return Name + " is " + Age.ToString();
    }
}

class TestPerson
{
    static void Main()
    {
        // Create a Person object by using the instance constructor.
        Person person1 = new Person("George", 40);

        // Create another Person object, copying person1.
        Person person2 = new Person(person1);

        // Change each person's age.
        person1.Age = 39;
        person2.Age = 41;

        // Change person2's name.
```

```
        person2.Name = "Charles";

        // Show details to verify that the name and age fields are distinct.
        Console.WriteLine(person1.Details());
        Console.WriteLine(person2.Details());

        // Keep the console window open in debug mode.
        Console.WriteLine("Press any key to exit.");
        Console.ReadKey();
    }
}
// Output:
// George is 39
// Charles is 41
```

See also

- ICloneable
- C# Programming Guide
- Classes and Structs
- Constructors
- Finalizers

Is this page helpful?

Yes No