**History of C#**

C# is one of the most popular programming languages ​​developed by Microsoft, along with the dot net framework development, which has been approved by ECMA-334 and ISO. C# was designed to be general-purpose, high-level, fully object-oriented, as well as a component-oriented programming language. **Mr. Anders Hejlsberg** is the creator of this widely used programming language. Anders Hejlsberg, aimed to fix the flaws in the popular programming languages such as Java, C++, Delphi, and other prominent programming languages. He cited that the flaws enabled him to work on Common Language Runtime(CLR). According to him, CLR is responsible for the C# language design. The language was based on C++ and Java, with additional extensions, libraries, and concepts for implementing different OOPS and component-oriented programming concepts.

**Object Oriented programming**

Object Oriented programming (OOP) is a programming paradigm that relies on the concept of **classes** and **objects**. It is used to structure a software program into simple, reusable pieces of code blueprints (usually called classes), which are used to create individual instances of objects. There are many object-oriented programming languages including JavaScript, [C++](https://www.educative.io/blog/how-to-learn-cpp-the-guide-for-beginners), [Java](https://www.educative.io/blog/object-oriented-programming-concepts-java), and [Python](https://www.educative.io/blog/how-to-use-oop-in-python).

**C# versions**

C# version 1.0 was a viable alternative to Java on the Windows platform.

The major features of C# 1.0 included:

* [Classes](https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/types/classes)
* [Structs](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/struct)
* [Interfaces](https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/types/interfaces)
* [Events](https://docs.microsoft.com/en-us/dotnet/csharp/events-overview)
* [Properties](https://docs.microsoft.com/en-us/dotnet/csharp/properties)
* [Delegates](https://docs.microsoft.com/en-us/dotnet/csharp/delegates-overview)
* [Operators and expressions](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/)
* [Statements](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/statements)
* [Attributes](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/attributes/)

C# version 1.2 shipped with Visual Studio .NET 2003. It contained a few small enhancements to the language.

While C# may have started as a generic Object-Oriented (OO) language, C# version 2.0 changed that in a hurry. Once they had their feet under them, they went after some serious developer pain points. And they went after them in a significant way.C# version 2.0 brought iterators. To put it succinctly, iterators let you examine all the items in a List (or other Enumerable types) with a foreach loop. Having iterators as a first-class part of the language dramatically enhanced readability of the language and people's ability to reason about the code.

C# version 3.0 came in late 2007, along with Visual Studio 2008, though the full boat of language features would actually come with .NET Framework version 3.5. This version marked a major change in the growth of C#. It established C# as a truly formidable programming language.

C# version 4.0, released with Visual Studio 2010, would have had a difficult time living up to the groundbreaking status of version 3.0. With version 3.0, C# had moved the language firmly out from the shadow of Java and into prominence. The language was quickly becoming elegant.

C# version 5.0, released with Visual Studio 2012, was a focused version of the language. Nearly all of the effort for that version went into another groundbreaking language concept: the async and await model for asynchronous programming.

With versions 3.0 and 5.0, C# had added major new features in an object-oriented language. With version 6.0, released with Visual Studio 2015, it would go away from doing a dominant killer feature and instead release many smaller features that made C# programming more productive.

C# version 7.0 was released with Visual Studio 2017. This version has some evolutionary and cool stuff in the vein of C# 6.0, but without the compiler as a service.

C# 8.0 is the first major C# release that specifically targets .NET Core. Some features rely on new CLR capabilities, others on library types added only in .NET Core.

C# 9.0 was released with .NET 5. It's the default language version for any assembly that targets the .NET 5 release.

**Principi objektno – orijentiranog programiranja**

Klasa

Class - osnovni tvorbeni blok objektno orijentiranog jezika kao što je C, je uzorak koji opisuje podatke i ponašanje povezano s instancama (instances) te klase. Kad stvorite instancu klase, stvorili ste objekt koji izgleda i ponaša se kao i druge instance te iste klase. Podaci vezani uz klasu ili objekt su spremljeni u varijablama, a ponašanje vezano uz klase ili objekte se opisuje pomoću metoda. Metode su slične funkcijama ili procedurama u proceduralnim jezicima.

Primjer definiranja klase Stanovi:

class Stanovi

{

public int broj\_stana;

public int broj\_stanara;

public string ime\_stanara;

}

Objekti

U OOP metafori osnovna jedinica je [objekt](https://hr.wikipedia.org/wiki/Objekt), što je konkretna realizacija [klase](https://hr.wikipedia.org/w/index.php?title=Klasa&action=edit&redlink=1). U objektno orijentiranom svijetu izbjegava se uporaba globalnih varijabli, nego svaki objekt ima svoje varijable, koje još zovemo i [podatkovni članovi](https://hr.wikipedia.org/w/index.php?title=Podatkovni_%C4%8Dlna&action=edit&redlink=1) klase. Isto tako nema više ni samostojećih funkcija, nego funkcije pripadaju klasi, bilo da vanjskom svijetu nešto kazuju o stanju objekta, ili mijenjaju stanje objekta, a zovemo ih [funkcijski članovi](https://hr.wikipedia.org/w/index.php?title=Funkcijski_%C4%8Dlan&action=edit&redlink=1) klase ili metode.

Primjer objekta klase Stanovi :

Stanovi stan1 = new Stanovi(1, 5, "Obitelj Stark");

Konstruktori

Specijalna metoda koja se poziva kod stvaranja objekta. U C# upotrebom ključne riječi „new“ koju slijedi ime klase.

Enkapsulacija

Koncept skupljanja metoda, svojstava, događaja i atributa u povezanu klasu i micanja detalja od korisnika. Povezuje podatake i funkcije u objekte. Klasa – opisuje način na koji su povezani podaci i funkcije te pokazuje temeljna svojstva svojih objekata i istovremeno skriva detalje implementacije. Objekti međusobno mogu biti u interakciji samo preko javnih (public) atributa-svojstava i metoda objekta

Nasljeđivanje

Objektno-orijentirani koncept koji definira da jedna klasa može biti izvedena iz drugog ili drugih (bazne klase) i nasljeđuje njihovo sučelje i kod (derivirana ili klasa dijete). Omogućuje gradnju hijerarhija koje izražavaju međusobne odnose, klase mogu nasljeđivati mogućnosti od klasa koje su više u hijerarhiji, nove klase mogu nasljeđivati funkcionalnost postojećih klasa i prema potrebi modificirati ili proširiti tu funkcionalnost. Roditeljska klasa od koje se funkcionalnost nasljeđuje zove se bazna klasa derivirane (izvedene) klase.

Polimorfizam

Objektno-orijentirani koncept koji kaže da različiti objekti mogu imati različite implementacije iste funkcije (npr. metoda Zbroji).

Method overriding

Objektno-orijentirani koncept koji definira da izvedena klasa može stvoriti različitu implementaciju metode bazne klase – u potpunosti „gazi“ preko ponašanja bazne klase.

Method overloading

Sučelje - dogovor koji definira strukturu metoda, svojstava, događaja (i indeksiranja), ne može se stvoriti objekt direktno iz sučelja, prvo treba stvoriti klasu koja implementira mogućnosti sučelja.

Objektno-orijentirani koncept koji postavlja da neka metoda može imati više različitih sučelja uz zadržavanje istog imena.