1. C# və .NET-ə Giriş

C# ümumi məqsədli (general-purpose), tip-təhlükəsiz (type-safe), obyekt-yönümlü (object-oriented) proqramlaşdırma dilidir. Dilin məqsədi proqramçının məhsuldarlığını (programmer productivity) artırmaqdır. Bu məqsədlə C# sadəlik, ifadəlilik və performans arasında balans yaradır. Dilin ilk versiyasından bəri baş memarı (chief architect) Anders Hejlsberg-dir (Turbo Pascal-ın yaradıcısı və Delphi-nin memarı). C# dili platformadan asılı deyil (platform neutral) və müxtəlif platform-specific runtime-lar ilə işləyir.

Object Orientation

C# object-orientation paradiqmasının zəngin tətbiqidir və özündə *encapsulation*, *inheritance* və *polymorphism* daxildir. Encapsulation obyektin ətrafında sərhəd yaratmaq deməkdir ki, bu da onun public davranışını private implementasiya detallarından ayırır. Aşağıda object-oriented perspektivdən C#-ın fərqləndirici xüsusiyyətləri göstərilir:

Unified Type System

C#-da əsas tikinti bloku type adlanan encapsulated data və function-lar vahididir. C#-da unified type system var ki, burada bütün type-lar sonda ümumi base type-ı paylaşırlar. Bu o deməkdir ki, bütün type-lar - istər biznes obyektlərini təmsil etsinlər, istərsə də rəqəmlər kimi primitive type-lar olsunlar - eyni əsas funksionallığı paylaşırlar. Məsələn, istənilən type-ın instance-ı onun `ToString` metodu çağırılaraq string-ə çevrilə bilər.

Class və Interface-lər

Ənənəvi object-oriented paradiqmada yeganə type növü class-dır. C#-da bir neçə digər type növü var ki, onlardan biri də *interface*-dir. Interface data saxlaya bilməyən class kimidir. Bu o deməkdir ki, o yalnız *behavior* təyin edə bilər (*state* yox), bu da multiple inheritance-ə və specification ilə implementation arasında ayrılığa imkan verir.

Property-lər, Method-lar və Event-lər

Pure object-oriented paradiqmada bütün function-lar *method*-dur. C#-da method-lar *function member*-lərin yalnız bir növüdür, bunlara həmçinin *property*-lər və *event*-lər daxildir (başqaları da var). Property-lər obyektin state-inin bir hissəsini encapsulate edən function member-lərdir, məsələn button-un rəngi və ya label-in mətni kimi. Event-lər obyektin state dəyişikliklərində hərəkət etməyi sadələşdirən function member-lərdir.

C# əsasən object-oriented dil olsa da, *functional programming paradiqmasından* da bəzi elementlər götürür, xüsusən:

Function-lar value kimi istifadə oluna bilər

Delegate-lərdən istifadə edərək, C# function-ların digər function-lara value olaraq ötürülməsinə imkan verir.

C# təmizlik (purity) üçün pattern-ləri dəstəkləyir

Functional programming-in əsası dəyişən dəyərlərə malik variable-lardan qaçmaq və declarative pattern-lərə üstünlük verməkdir. C#-ın bu pattern-lərə kömək edən əsas xüsusiyyətləri var, o cümlədən dərhal variable-ları "capture" edən adsız function-lar yazmaq imkanı (*lambda expression*-lar), və *query expression*-lar vasitəsilə list və ya reactive proqramlaşdırma aparmaq bacarığı. C# həmçinin *immutable* (read-only) type-lar yazmağı asanlaşdıran *record*-lar təqdim edir.

Type Safety

C# əsasən *type-safe* dildir, yəni type-ların instance-ları yalnız onların təyin etdiyi protokollar vasitəsilə qarşılıqlı əlaqədə ola bilər, beləliklə hər type-ın daxili uyğunluğunu (internal consistency) təmin edir. Məsələn, C# sənin *string* type ilə sanki *integer* type imiş kimi qarşılıqlı əlaqədə olmağına mane olur.

Daha dəqiq desək, C# *static typing*-i dəstəkləyir, yəni dil *compile time*-da type safety-ni tətbiq edir. Bu, *runtime*-da tətbiq edilən type safety-yə əlavədir.

Static typing proqram işə salınmazdan əvvəl böyük bir xəta sinfini aradan qaldırır. Bu, yükü runtime unit testlərdən compiler-ə köçürür ki, proqramdakı bütün type-ların düzgün uyğunlaşdığını yoxlasın. Bu, böyük proqramların idarə olunmasını daha asan, daha proqnozlaşdırıla bilən və daha möhkəm edir.

Bundan əlavə, static typing Visual Studio-da IntelliSense kimi alətlərə proqram yazmaqda kömək etməyə imkan verir, çünki o, verilmiş variable-ın hansı type olduğunu bilir və beləliklə həmin variable üzərində hansı method-ları çağıra biləcəyini göstərir. Belə alətlər həmçinin proqramda variable, type və ya method-un harada istifadə edildiyini müəyyən edə bilir ki, bu da etibarlı refactoring-ə imkan verir.

C# həmçinin *dynamic keyword* vasitəsilə kodunuzun hissələrinin dynamically typed olmasına imkan verir. Bununla belə, C# əsasən statically typed dil olaraq qalır.

C# həmçinin *strongly typed* dil adlanır, çünki onun type qaydaları ciddi şəkildə tətbiq edilir (istər statik olaraq, istərsə də runtime-da). Məsələn, siz integer qəbul etmək üçün nəzərdə tutulmuş function-ı floating-point number ilə çağıra bilməzsiniz, əvvəlcə floating-point number-ı açıq şəkildə (*explicitly*) integer-ə convert etməli olursunuz. Bu, səhvlərin qarşısını almağa kömək edir.