شی گرایی در C#

کلاس

کلاس روشی برای بسته بندی نوع داده مجرد است . در کلاس امکان بسته بندی و محصور کردن (Encapsulation) مجموعه ای از داده ها است . روال های پردازش کننده این داده ها را به صورت یک بسته فراهم می کند.

داده های داخل یک کلاس به وسیله کلاس محافظت می گردد. به گونه ای که پردازش داده های خصوصی یک کلاس از طریق روال های دخلی آن امکان پذیر است . داده های یک کلاس را متغیرهای کلاس و روال های آن را روش نامیده اند.

برای مثال کلاس انسان ها یک کلاس قابل تعریف است. در این کلاس خصوصیات مشترک انسانها تعریف می گردد و هیچ انسان خاصی را نشان نمی دهد. کلاس یک نوع است. همانگونه که مثلا int یک نوع است.

عملیات محاسباتی (یا غیر محاسباتی) بر روی نوع داده انجام نمی شود. بلکه این عملیات بر روی متغیرهایی که از این نوع داده تعریف می گردد انجام می شود. به طور مشابه عملیات محاسباتی (یا غیر محاسباتی) روی کلاس انجام نمی شود.

شی

دیدگاه شی گرایی (Object Oriented) از اواسط دهه ۱۹۷۰ تا اواخر ۱۹۸۰ در حال مطرح شدن بود.

در این دوران تلاشهای زیادی برای ایجاد روشهای تحلیل و طراحی شی گرا صورت پذیرفت. در نتیجه این تلاشها بود که در طول ۵ سال یعنی ۱۹۸۹تا ۱۹۹۴ ، تعداد متدلوژیهای شی گرا از کمتر از ۱۰ متدلوژی به بیش از ۵۰ متدلوژی رسید. تکثیر متدلوژیها و زبانهای شی گرایی و رقابت بین اینها به حدی بود که این دوران به عنوان "جنگ متدلوژیها " لقب گرفت. از جمله متدلوژیهای پر کاربرد آن زمان می توان از Shlayer- Mellor، Coad-

yourdan، Fusion، OMT، OOSE، Booch و غیره نام برد. فروانی و اشباع متدلوژیها و روشهای شی گرایی و نیز نبودن یک زبان مدلسازی استاندارد ، باعث مشکلات فروانی شده بود . از یک طرف کاربران از متدلوژیهای موجود خسته شده بودند ، زیرا مجبور بودند از میان روشهای مختلف شبیه به هم که تفاوت کمی در قدرت و قابلیت داشتند یکی را انتخاب کنند . بسیاری از این روشها ، مفاهیم مشترک شی گرایی را در قالبهای مختلف بیان می کردند که این واگرایی و نبودن توافق میان این زبانها ، کاربران تازه کار را از دنیای شی گرایی زده می کرد و آنها را از این حیطه دور می ساخت . عدم وجود یک زبان استاندارد ، برای فروشندگان محصولات نرم افزاری نیز مشکلات زیادی ایجاد کرده بود.

پیدایش شی گرایی

برنامه نویسی شی گرا در اوایل دهه ۱۹۷۰ توسط آلن کی Alan Kay طراحی شده یعنی اولین قدمهای این سبک برنامه نویسی توسط آلن کی برداشته شده است. اولین زبان شی گرا توسط آلن کی طراحی شد. اسم این زبان Small Talk است.

آلن کی گفته بود که : آن چیزی که باعث شد این فکر به ذهنم برسد نحوه عملکرد سلولهای زیست محیطی بود. یعنی این سبک برنامه نویسی از روی سلولهای جاندارها الگو برداری شده است.

آن چیزی که باعث شد که آلن کی از روی سلول های جانداران الگو برداری کند نحوه زندگی سلولها بود:

- هر سلول نمونه ای از اصل است و هر خصوصیاتی که دارد از اصل خود به ارث برده.(ژنتیک سلول). همچنین هر سلول رفتارهایی دارد که از اصل خود به ارث برده.
- سلولها همگی مستقل از هم زندگی می کنند و براساس ارسال پیام های شیمیایی با یکدیگر ارتباط برقرار می کنند. ارسال پیام به این صورت است که پیام از پوسته یکی خارج و به پوسته دیگری وارد می شود.
- سلولها می توانند از یکدیگر متمایز شوند.

با توجه به گفته های بالا

می توان متوجه شد که همان مشخصه کلاسها رو بیان می کند یعنی هر شی از یک کلاسی تشکیل شده که ویژگی های آن کلاس رو با خودش به ارث برده است.

همانطور که می دانیم اشیا با یکدیگر ارتباط برقرار می کنند. نحوه ارتباط یا فرستادن پیام در اشیا هنگام فراخوانی رفتارها در یک رویداد است.
هر شی خودش یک شناسنامه یا Identifier دارد که ویژگی های آن شی را بیان می کند.

Small Talk مانند سلولهای جانداری عمل می کند. یعنی آلن کی در تمامی قسمتهای این زبان تعیین کرده بود که اشیا با هم ارتباط برقرار می کنند و دارای شناسنامه ای هستند و همچنین مستقل از همدیگر کار می کنند.

اصول اولیه ای که آلن کی برای برنامه نویسی شی گرا تعیین کرده بود اینها هستن:

- هر چیزی یک شی است.
- هر برنامه ای شامل اشیا هست که اشیا با ارسال پیام به یکدیگر تعیین می کنند که چه کاری باید الان انجام بشود.
- هر شی یک حافظه Memory برای خودش دارد که بتوان به وسیله آن اشیای دیگر را ساخت.
- هر شی خودش از یک کلاس Class هست.

برنامه نویسی شی گرا

برنامه نویسی شی گرا شیوه نوینی است که در آن می توان قطعاتی را ایجاد کرد و در برنامه های مختلف مورد استفاده قرار داد.قابلیت خوانایی برنامه هایی که در این روش نوشته می شوند بالا بوده ، تست ، عیب یابی و اصلاح آن ها آسان است . شی گرایی ، بر اشیا تاکید دارد. در برنامه نویسی شی گرا اشیا به صورت انتزاع مطرح می شوند.

انتزاع: به آن چیزی می گویند که شما در مورد آن فکر می کنید و در یک دید کلی مطرح می کنید. مثلاً وقتی به یک دانه شن فکر می کنید ناخوداگاه فکرتان به سمت ساحل می رود یا وقتی به یک درخت فکر می کنید ذهنتان به سمت جنگل متمرکز می شود. به این انتزاع می گویند که در این سبک برنامه نویسی مطرح می شود که اشیا با توجه به کلاس های خودشان ساخته می شوند که خود کلاس ها ممکن است که از کلاسهای دیگری مشتق شده باشند.

در این تحقیق شی گرایی را در دو زبان برنامه نویسی (C,C#,++) مورد بررسی قرار می دهم و شی گرایی و موارد آن را در هر یک به اختصار توضیح می دهم.

در ابتدا نگاهی می اندازیم به این خاصیت در زبان برنامه نویسی C++ و در ادامه شی گرایی را در زبان C# مورد بررسی قرار می دهیم .

شئی گرایی در زبان C++

برنامه نویسی شی گرا چیست؟

از آنجا که اساس تکوین C++ برنامه نویسی شی گرا است. پس مهم است که تعریف دقیقی را از برنامه نویسی شی گرا ارائه دهیم . شی گرا از بهترین مفاهیم برنامه سازی ساخت یافته بوجود آمده است و با چندین مفهوم قوی ترکیب شده که امکان سازماندهی برنامه ها را به طور کارآمد را فراهم می کند.

به طور کلی هنگامی که در حالتی شی گرا نیز برنامه می نویسید مساله را به بخش های تشکیل دهنده آن تجزیه می کنید. هر مولفه ای یک شی خود ظرف می شود که شامل دستورالعملهای خودش و داده های مرتبط با آن شی است . از طریق این عملیات پیچیدگی کاهش یافته و می توان برنامه های بزرگ را مدیریت کرد . همه زبان های برنامه نویسی شی گرا در سه چیز مشترک هستند : کپسوله سازی ، چند ریختی و وراثت. که در زیر اشاره مختصری به هریک از آنها می کنیم.

کپسوله سازی

همانطور که می دانید تمام برنامه ها از دو عنصر اصلی تشکیل می شوند : عبارت برنامه (کد) و داده ها . کد بخشی از برنامه است کد عملیات را اجرا می کند و داده ها اطلاعاتی است که توسط این عملیات تحت تاثیر قرار گرفته . کپسوله سازی یک مکانیزم برنامه نویسی است که کد و داده ها را با هم در یک جا قرار داده و هر دو را از استفاده نادرست و تداخل خارجی ایمن نگه می دارد .

در یک زبان شی گرا ، کد و داده ممکن است با هم در چنین روش محدود شوند که یک جعبه سیاه خود ظرف را ایجاد می کند . درون جعبه تمام داده های مورد نیاز و کد است . هنگامی که در این روش کد و داده ها با هم پیوند برقرار می کنند یک شی به وجود می آید . به عبارت دیگر یک شی ابزاری است که ازکپسوله سازی پشتیبانی می کند.

درون یک شی (کد وداده ها) یا هردو ممکن است برای آن شی محلی (خصوصی/Private) یا عمومی (Public) باشند.کد یا داده های محلی فقط توسط بخش دیگری از شی شناخته شده و قابل دست یابی هستند.

به همین دلیل کد یا داده محلی برای قطعه ای از برنامه که خارج از شی است قابل دسترس نمی باشد . هنگامی که کد یا داده ها عمومی هستند بخش ها دیگری از برنامه ممکن است به آنها دسترسی داشته باشند حتی اگر درون شی تعریف شده باشند بخش های عمومی یک شی برای ارائه یک ارتباط کنترل شده با عناصر محلی شی مورد استفاده قرار می گیرند.

چند ریختی

چند ریختی کمیتی است که به یک رابط امکان می دهد تا برای یک کلاس عمومی از عملیات مورد استفاده قرار می گیرد . عمل خاص توسط ذات حقیقی شی تعیین می شود.به عنوان مثال یک پشته را در نظر بگیرید . ممکن است برنامه ای داشته باشید که نیاز به سه نوع مختلف پشته داشته باشید یک پشته برای مقادیر صحیح ،یک پشته برای اعداد اعشاری و یک پشته برای کاراکترها مورد استفاده قرار می گیرد.در این صورت الگوریتمی که هر سه پشته را پیاده می کند یکسان است حتی اگر داده هایی که در آنها ذخیره می شود متفاوت باشند . در یک زبان غیر شی گرا نیاز خواهید داشت تا سه نوع مختلف از روال های پشته را ایجاد کرده . به هر کدام نام متفاوتی قرار داده و برای هر کدام از روابط خاص خودش استفاده کنید . به دلیل وجود چند ریختی در C++ می توانید یک مجموعه روال عمومی از پشته ایجاد کرده وآن را برای هر سه نوع به کار ببرید.

به طور کلی مفهوم چندریختی اغلب توسط عبارت "یک رابط چندین روشی" بیان می شود . این بدین معنی است که امکان طراحی یک رابط عمومی برای گروهی از عملیات مرتبط وجود دارد.

چند ریختی با اعمال رابط یکسانی که برای تعیین یک کلاس عمومی مورد استفاده قرار می گیرد . به کاهش پیچیدگی کمک می کند . این است وظیفه کامپایلر تا فعالیت خاصی (مثل متد) را برای اعمال روی آن انتخاب کند . شما به عنوان برنامه نویس نیاز ندارید تا این انتخاب را انجام دهید شما فقط نیاز دارید روابط عمومی را به خاطر سپرده و استفاده کنید.

زبان های برنامه نویسی شی گرای اولیه چون به صورت مفسری بودند از چند ریختی در زمان اجرا پشتیبانی می کردند. ولی چون C++ کامپایلری است پس هم در زمان اجرا و هم در زمان کامپایل از چند ریختی پشتیبانی می کند.

وراثت

وراثت عملی است که یک شی می تواند مشخصه های شی دیگری را به دست آورد. به همین دلیل از مفهوم دسته بندی سلسه مراتبی پشتیبانی می کند.

بدون استفاده از وراثت هر شی به طور مجزا بایستی تمام مشخصه های خودش را تعریف کند . با استفاده از وراثت شی فقط نیاز به تعریف مشخصه هایی دارد که در داخل آن کلاس منحصر به فرد هستند . این سبب می شود که صفات عمومی را از پدرشان به ارث ببرند.

بنابراین مکانیزم وراثت به یک شی امکان می دهد تا نمونه خاص از یک حالت عمومی تر باشد.

شی گرایی در زبان C#

شی گرایی (oop) در C# بر چند پایه استوار است که به قرار زیرند:

- Inheritance
- Encapsulation
- Polymorphism
- Abstraction
- Interface

اکنون به توضیح مختصر هر یک می پردازیم:

Inheritance ارث بری

پدر و فرزندی را در نظر بگیرید . هر پدری مشخصات فردی به خصوصی دارد . فرزند وی می تواند همه خصوصیات او را به ارث برد و خصوصیتهای دیگری نیز داشته باشد که پدرش ندارد . این یعنی ارث بری !

در برنامه نویسی شی گرا از مفهوم ارث بری استفاده های زیادی می شود . قابلیت استفاده دوباره از کد (Reusability) یکی از مزیات اصلی ارث بری است.

Encapsulation

همانطور که از اسمش پیداست ، به قرار دادن پیاده سازی در یک کپسول اشاره می کند ، به طوری که کاربر بیرونی از نحوه پیاده سازی مطلع نباشد و فقط بداند که این کپسول کار خاصی را انجام می دهد.

هدف Encapsulationاین است که ما را از پرداختن به ریز موضوعات رها کند و اشیا را به صورت یک جعبه سیاهی بدانیم که به ازای یک ورودی خاص خروجی خاصی می دهند .

در C# برای کپسوله کردن از Access Modifier های Private، Protected، Public استفاده شود.

Polymorphism

فرض کنید پدر شما کار خاصی را به طریق خاصی انجام می دهد . مثلا برای پختن غذا اول ظرفهای دیشب را شسته و بعد گاز را روشن می کند و بعد غذا را می پزد! شما که خصوصیات پدر و کارهای او را به ارث می برید برای مثال برای پختن غذا ابتدا گاز را روشن می کنید بعد کبریت می کشید،غذا را می پزید و بعد ظرفهای دیشب را می شویید! پختن غذا کاری است که شما از پدر خود به ارث می برید ، ولی آن را به طریق دیگری انجام می دهید . یعنی یک کار توسط فرزندان مختلف یک پدر به طرق مختلفی انجام می شود . این دقیقا همان چیزی است که به آن چند شکلی یا Polymorphismمی گویند.

Abstraction

تجرید یا مجرد سازی ! به کلاسی مجرد گفته می شود که پیاده سازی متدها در آن انجام نمی شود. فرض می کنیم که شما رئیس یک شرکت بزرگ برنامه نویسی هستید و می خواهید پروژه بزرگی را انجام دهید. برای اجرای پروژه از برنامه نویسان مختلفی استفاده می کنید که ممکن است همه آنها هموطن نباشند! مثلا هندی ، ایرانی یا آلمانی باشند! اگر قرار باشد هر برنامه نویسی در نامگذاری متدها و کلاسهایش آزاد باشد ، در کد نویسی هرج ومرج به وجود می آید . شما به عنوان مدیر پروژه ، کلاسی تعریف می کنید که در آن تمام متدها با ورودی و خروجی هایشان مشخص با شند . ولی این متدها را پیاده سازی نمی کنید و کار پیاده سازی را به برنامه نویسان می دهید و از آنها می خواهید که همه کلاسهایی که می نویسند از این کلاس شما به ارث ببرند و متدها را به طور دلخواه پیاده سازی کنند . این باعث می شود که با داشتن یک کلاس ، ورودی و خروجی های مورد نظر خود را داشته باشید و دیگر نگران برنامه نویسان نباشید. کلاسی که شما تعریف می کنید یک کلاس مجرد نامیده می شود.

برای تعریف یک کلاس مجرد از کلمه کلیدی abstract استفاده می کنیم. فیلدهایی که می خواهیم در کلاس های مشتق شده از این کلاس پیاده سازی شوند حتما باید با abstract تعریف شوند . یک کلاس مجرد می تواند فیلدها و متدهای نا مجرد داشته باشد.اگر متد نا مجردی در یک کلاس مجرد تعریف کردید، حتما باید آن را پیاده سازی کنید و نمی توانید پیاده سازی آن را به کلاسهای مشتق شده بسپارید.

Interface

اینترفیس در برنامه نویسی همانند همان کلاس است تنها با این تفاوت که هیچکدام از اعضای آن پیاده سازی نمی شوند . در واقع یک اینترفیس گروهی از متدها ، خصوصیات ، رویدادها و Indexer ها هستند که در کنار هم جمع شده اند. اینترفیس ها را نمی توان) Instantiate وهله سازی) کرد (یعنی نمی توان وهله ای از یک اینترفیس ایجاد کرد !) . تناه چیزی که اینترفیس دارا می باشد امضای (signature) تمامی اعضای آن می باشد. به ای معنی که ورودی و خروجی متدها ، نوع property ها و ... در آن تعریف می شوند ولی چیزی پیاده سازی نمی شود. اینترفیس ها سازنده و فیلد ندارد. یک اینترفیس نمی تواند Operator Overload داشته باشد و دلیل آن این است که در صورت و جود ویژگی ، احتمال بروز مشکلاتی از قبیل ناسازگازی با دیگر زبانهای Net.مانند VB.Net که از این قابلیت پشتیبانی نمی کند وجود داشت. نحوه تعریف اینترفیس بسیار شبیه تعریف کلاس است تنها با این تفاوت که در اینترفیس پیاده **سازی وجود ندارد.**