2nd International Conference on Management & Economics IN 21st CENTURY
20th December 2016 - TEHRAN - IRAN
www.icme21.com
دومین کنفرانس بین المللی مدیریت و اقتصاد در قرن ۲۱
۳۰ آذر ۱۳۹۵ - تهران - ایران

بررسی اجزای برنامه نویسی شی گرا

نیما ثابت فر – کارشناس ارشد MBA Marketing – دانشگاه تهران - nimainkish@gmail.com

چکیده

برنامه‌نویسی شیءگرا (به انگلیسی Object Oriented Programming مخفف OOP) یک شیوه برنامه نویسی است که ساختار یا بلوک اصلی اجزای آن اشیا می باشد. در واقع دراین نوع پارادیام برنامه نویسی برنامه به شیء گرایش پیدا می کند و در ضمن مزیت هایی از جمله "پیچیدگی کم" و "هزینه ی کم" نسبت به سایر پارادیام های برنامه نویسی دارد. زبان‌های برنامه نویسی شیءگرا، زبان‌هایی هستند که در آن برنامه نویس می‌تواند اشیاء مختلفی را تعریف نماید و از اشیاء تولید شده استفاده نماید. امروزه اکثر زبان‌های دستوری برنامه نویسی از فنون شیءگرایی پشتیبانی می‌کنند. زبانهایی مانند جاوا، سی++، سی شارپ، دلفی از جمله زبانهای شیءگرا هستند. حتی بسیاری از زبانهای روال گونه که ساختار برنامه‌ها در آنها بلوک‌هایی با نام پروسیجر است امروزه از فنون شیءگرایی نیز پشتبانی می‌کنند. زبانهای سی++ و پی‌اچ‌پی از این جمله هستند. هر شیء یک سری خصوصیت و قابلیت دارد، که اصطلاحاً Properties و Operation خوانده می‌شوند. مفهوم کلاس در C++ مهمترین مفهوم محسوب می شود. که در این مبحث ما به بررسی دقیق تر این مفهوم ونیز مفاهیم مرتبط با آن می پردازیم . یعنی ابتدا به معرفی کلاسها پرداخته و سپس نگاهی دقیق تر به بنیادهای کلاس خواهیم داشت . همچنین درباره روشها methods و سازنده ها construction و واژه کلیدی this مطالبی می آموزید. برنامه نویسی شی گرا هسته اصلی C++ است.

کلمات کلیدی: شی گرایی، C++، OOP، پروسیجر

مقدمه

شی گرایی به پیشرفت و تکامل سیستمهای خبره و هوش مصنوعی سرعت بخشید چرا که از ابتدا بر اساس الگوی حیات سلولی بنا شده بود. در یک برنامه شی گرا، یک نمونه از کلاسی معین، متولد (Instantiate) می شود. در طول حیاط خود از طریق شناسایی الگوهای فراخوانی رفتارهای خود توسط محیط پرامون و پاسخهایی که از دیگر اشیا دریافت می کند، حالت (State) خود را بهینه می نماید، (در صورتی که برنامه نویس تصمیم به پیاده سازی یادگیری ماشینی هرچند بسیار ساده در کلاس شی مورد نظر، داشته باشد،) چیزهایی را از پیرامون درمی یابد و رفتارهای بالغ تری از خود نشان می دهد، و نهایتا در زمان مناسب می میرد(آهوجا و کارلی،1999).

یکی از مهمترین و اصلی ترین مشکلات برای افرادی که به تازگی با محیطهای برنامه نویسی شی‌گرا آشنا می شوند، درک مفاهیم شیءگرائی است. در حقیقت درک مفاهیمی چون شیء و مولفه (Component) بسیار دشوار نیست، کافیست کمی به اطراف خود با دقت نگاه کنیم. ما در دنیایی از اشیاء مختلف زندگی میکنیم. تلویزیون، رادیو در و پنجره، همه و همه نمونه هایی از اشیاء مختلفی هستند که در اطراف ما وجود دارند. اما درک و پیاده‌سازی این مفهوم در یک زبان برنامه‌سازی اندکی متفاوت است. شی گرایی یک مفهومه اما پیاده سازی اون در زبانهای مختلف متفاوت هست(الکادی و همکاران،2003).

شیء چیست؟

با یک دید تصویری به سادگی می توانید اشیاء مختلفی را در اطراف خود بیابید. تمامی این اشیاء دارای سطوح و درجه پیچیدگی متفاوتی هستند. پیچیدگی آنها به شکل ظاهری و نوع رفتار آنها بستگی دارد.

در شیء گرایی به "شکل ظاهر" در اصطلاح، صفت یا Attribute و به عملی که شیء انجام می دهد، رفتار یا Behavior می گویند. برای مثال یک صندلی را در نظر بگیرید. صندلی صفات مختلفی دارد ولی رفتار خاصی ندارد. مثلاً پایه های صندلی جزو صفات آن بشمار می روند. با کمی دقیق تر شدن می توان از این صفات برای توصیف صندلی استفاده کرد. بعنوان مثال تعداد پایه های صندلی می تواند عددی بین 3 تا 5 باشد. محل نشستن صندلی می تواند جمله‌ای در وصف جنس آن و مقدار مصرف ماده سازنده آن باشد. پشتی صندلی را نیز می توان بعنوان متغییری boolean در نظر گرفت چراکه برخی از صندلی ها فاقد پشتی هستند. با استفاده از این سه صفت ساده، به راحتی می توان صندلی را توصیف نمود و با همین سه صفت

میتوان گونه‌های مختلفی از صندلی را نیز توصیف کرد. منظور از رفتار، عملی است که یک شیء انجام می دهد. از اینرو برای صندلی نمی توان به سادگی صفات آن، رفتاری را متصور شد. مثلاً می‌توانیم بگوئیم تاشو بودن صندلی یکی از رفتارهای آن می تواند باشد، چراکه عملی است که می تواند یک صندلی آنرا انجام هد.(کوپر، مونچ،2000).

حال شیء دیگری مانند تلویزیون را در نظر بگیرید. صفاتی که می توان برای تلویزیون در نظر گرفت عبارتند از : صفحه نمایش، سازنده آن و ... برای تلویزیون به راحتی می توان رفتار در نظر گرفت : خاموش و روشن شدن، تغییر کانال و کم و زیاد کردن صدا. این رفتارها بر اثر درخواست یک انسان یا همان کاربر اتفاق می افتند. شیء تلویزیون را می‌توان بصورت زیر نمایش داد.

بطور کلی، یک شیء را میتوان بوسیله صفات و رفتارهای آن بطور کامل توصیف نمود. یک شیء حتماً جسمی فیزیکی نیست، بلکه هر چیز قابل تصوری است که دارای صفت و رفتار است. در حقیقت میتوان گفت یک شیء شبیه به یک اسم است. اگر بتوان برای این اسم، صفت و رفتاری تعریف کرد، آن وقت تبدیل به شیء می شود(هکس،2003).

یک شیء می‌تواند هر یک از نیازمندی‌های یک پروژه باشد. طراحی و اعلان صحیح اشیاء و مشخص کردن صفات و رفتار آنها یکی از مقوله‌های مهم در مهندسی نرم افزار بر پایه شیءگرائی است، چراکه همگی تراکنشها بین اشیاء صورت می پذیرند. برنامه نویسی شی گراء، بدون شک مهمترین تحول در دنیای نرم افزار طی سالیان گذشته بوده که بیشترین تاثیر را در پیشبرد نرم افزار بدنبال داشته و ما امروز در صنعت گسترده نرم افزار از دستاوردهای وسیع آن بهره مند هستیم . در این مقاله قصد داریم ، به بررسی برنامه نویسی شی گراء پرداخته و از این زاویه نگاهی به دات نت داشته باشیم. از آنجایی 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 برمانه ها به طور کارامد 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 دستور العملهای خودش و داده های مرتبط به 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 .(ادواردسن و کامکار، 1998).

برنامه نویسی شی گرا

در زبان های برنامه نویسی اصول های وجود دارد، که یکی از جدید ترین و پرکاربرد ترین اصول های آن شی گرایی می‌باشد. حتما تابه حال نام این اصل را شنیده اید و مثل و یا صد هزار کدنویس دیگه بی تفاوت از آن رد شدید. شی گرایی دارای چند نوع اصل می‌باشد. مانند وراثت و کپسوله سازی و نهان سازی .(کربلایی،1383).

این اصول رو در اکثر زبان های امروزی مانند سی++ و یا دلفی و.... وجود دارد . اما به آن ها در زبان هایی مثل وی بی بسیارز کم توجه شده به حدی که برنامه نویسان ویژوال بیسیک وجود چنین چیزی را به یاد نمی آورند. متاسفانه یکی از دلایل پر طرفدار بودن وی بی(ویژوال بیسیک) همین موضوع می باشد. یکی از کم ترین کار برد هایی که ما روزانه داریم صد ها هزار بار از آن سود می بریم وراثت می باشد. تا به حال فکر کردید اگر وراثت وجود نداشت مجبور به نوشن تک تک کنترل ها در برنامه تان می شدید. مثلا برای گذاشتن 3 عدد تکست باکس باید سه بار نوشتن کلاس تکست را انجام می دادید واین کار علاوه بر زیاد کردن حجم فایل اجرایی و پیچیدگی کد ها حتی موجب اشغال شدن فضای زیادی از سی پی یو و پایین آمدن سرعت کار سیستم عامل می شد که برنامه نویس را از نوشتن برنامه متنفر می‌ساخت اما الان مثلا دلفی کاران به لطف گذاتن نام کلاس پایه در بین دو پرانتز در جلوی نام کلاس پایه می‌توانند به سادگی یک کلاس را مشتق کنند و دیگر هیچ یک از مشکلات بالا را نداشته باشند.و یا به طور مثال وقتی می خواهید یک اکتیوایس بنویسید دیگر لازم نیست کل کلاس را خودتان تعریف کنید بلکه به سادگی با یه وراثت می توانید یک کنترل ساخته و خواص آن راتغییر دهید و کنترل مورد نظر را بسازید (الراد و همکاران،2001).

تاریخچه شی گرایی :اول بار در سال 1966 در قالب زبانی به نام simula که به منظور پیاده سازی شبیه سازی کامپیوتری تهیه شده بود، ارائه شد.در سال 1970 ، مفهومی تحت عنوان Abstract Data Type(ADT) بوجود آمد . در دهه 70 زبان small talkبوجود آمد . ویژگی آن , این بود که اینترفیسش بصورت منویی طراحی شد و این ویژگی بدون شی گرایی ممکن نیست . در دهه 80 زمینه های کاربردی آن در هوش مصنوعی هم توسعه پیدا کرد . از دهه90 تا حالا بشدت در حوزه زبانهای برنامه نویسی رشد کرد . بطور مثال corba روشی شی گرا برای مدیریت شبکه (الراد و همکاران،2001).

مزایایی شی گرایی

1- مدل سازی دنیای واقعی را تسهیل می کند

2- شی گرایی امکان Reuse کردن برنامه های کامپیوتری را فراهم می کند

3- قابلیت استفاده مجدد.

Reuseability

یعنی یک قطعه برنامه ای را یکبار نوشتن و به دفعات متعدد در برنامه های مختلف از آن استفاده کردن .

مزایای Reuseability :

1- صرفه جویی در زمان

2 – صرفه جویی د رهزینه

3_ شی گرایی امکانات مناسبی برای ساخت سریع GUI دقیق فراهم می کند .

4_ امکان اجرای موازی (Parallel) نرم افزار را روی چندین CPU فراهم می کند .

5_ امکان Proto typing سریع فراهم می کند .

6_امکان توسعه کم هزینه تر و سریع تر و راحت تر نرم افزار را فراهم می کند (چن و همکاران،2000).

تئوری شی گرایی چیست ؟

این تئوری سه رکن دارد:

1- شی

2- کلاس

3- وراثت

1-اشیاء(Objects) و کلاسها (Classes):

2-شی کلیدی ترین مفهوم برنامه نویسی شی گرا است هر شی ء یک حالت و رفتار دارد و برنامه عبارت است از بر هم کنش

بین اشیاء . حالت یک شی ء عبارت است از متغییرهای عناصر داده ای شیيء و مقدار آنها . رفتار یک شی ء را متدهای آن تعیین می کند . شیء در واقع مقداری کد است که کار خاصیتی انجام می دهد (هاتچینسون،1996).

3- هر شیء کپسولی است از مقداری متغییر و کد که کار نگهداری و به روز در آوردن آنها را ساده می کند . معمولاً درون یک شیء از دسترس دنیای خارج به دور است و برای کار با آن باید از ارسال پیام استفاده کرد . مزیتش آن است که کاربر برای کار با آن هیچ نیازی به دانستن مکانیزم های درونی آن ندارد و فقط باید ساختار پیام ها را بداند . معمولاً ساختار پیام ها در اشیاء مختلف یکسان است . در دنیای واقعی هم وضع به همین منوال است مثلاً برای استفاده از یک تلویزیون شما هیچ نیازی به آشنایی با پیچیدگیهای درون آن ندارید فقط کافی است بدانید که باید آن را به برق زده و روشن کنید و یک کانال را انتخاب کنید . کلاس در واقع الگویی است برای ایجاد شیء . در واسط برنامه نویسی جاوا (java API) چندین کلاس مختلف وجود دارد . چندین کلاس با هم یک کتابخانه کلاس Class Library می سازند . برنامه نویسی جاوا اصولاً چیزی نیست جز طراحی و پیاده سازی کلاس ها . برنامه نویسی شیء (OOP) یکی از بزرگترین ایده های برنامه نویسی در دو دهه اخیر است ، اما تسلط کامل بر این ایده به سالها زمان و ممارست نیاز دارد . برنامه نویسی شیء در واقع پیوند دنیای واقعی با برنامه نویسی کامپیوتر است (هاتچینسون،1996).

4- به موازات رشد برنامه نویسی مدولار ، تکنولوژی دیگری در زمینه برنامه نویسی اختراع شد و توسعه یافت . برنامه نویسی شی گرا یا OOP -Object – Oriented Programming محرک توسعه این تکنولوژی رشد و پیچیدگی روز افزون برنامه ها و مشکلاتی بود که به تیغ آن برنامه نویسان را درگیر خود کرده بود . مهمترین منبع این مشکلات بر هم کنش غیر قابل پیش بینی قسمتهای مختلف یک برنامه با یکدیگر بود . چون این قسمتها مانند دانه های یک زنجیر در هم بافته شده بوند و هر تغییری در یک قسمت به راحتی سایر قسمتها را متاثر می کرد(فرایکین و لونهاردت،2002).

را ه حل این مشکل آن بود که هر قسمت برنامه در یک بسته بنام شی Object ، کپسوله یا Encapsulation شود . ساز و کار درونی هر شی مطلقا از دید دنیای خارج مخفی است و آنها نمی توانند تاثیری بر عملکرد وی بگذارند . البته یک شی نمی تواند بکلی از دنیای اطراف خود ایزوله شود چون بدین ترتیب دیگر چیز بی فایده ای بیش نخواهد بود، به همین دلیل برای ارتباط با دیگر قسمتهای برنامه ، هر شی از وسیله ای بنام واسط یا Interface استفاده می کند . واسط هر شی دو بخش دارد : خواص (داده ها) و متدها (کدها) ی آن . شیء گرایی (Object-Oriented) لغتی است که امروزه در صنعت نرم افزار باب شده است. شرکتها به سرعت حرکت می کنند تا خود را با این تکنولوژی سازگار کنند و آن را در برنامه های موجود خود وارد نمایند. در حقیقت بیشتر برنامه ها امروزه با شی گرایی توسعه می یابند(آفوت و همکاران،2003).

متد شیء گرایی یک راه متفاوت مشاهده برنامه هاست. با متد شیء گرایی شما یک برنامه را به قطعات بسیار کوچک یا آبجکتهایی تقسیم می‌کنید که تا اندازه ای مستقل از یکدیگر می باشند. به آن مانند ساختمانی از بلوکها نگاه کنید. به محض اینکه تعدادی آبجکتهای اساسی را در دنیای کامپیوتر ساختید یا بدست آوردید می‌توانید به سادگی آنها را کنار هم بگذارید تا برنامه های جدید را ایجاد نمائید. یکی از امتیازات اساسی متد شیء گرایی این است که می توانید یکبار اجزاء را ساخته و بارها و بارها استفاده کنید. یک بلاک ساختمان را می توانید در یک خانه یا یک قصر یا یک سفینه فضایی دوباره استفاده کنید. همچنین می توانید از یک قطعه طرح یا کد شیء گرایی در یک سیستم حاسبداری - یک سیستم بازرگانی یا یک سیستم پردازش سفارش استفاده مجدد نمائید. تفاوت متد شیء گرایی با روش سنتی توسعه: در روش سنتی روش توسعه به همراه اطلاعاتی که سیستم نگهداری خواهد کرد به خودمان وابسته است. در این روش ما از کاربران می رسیم که چه اطلاعاتی را نیاز دارند. پایگاه داده ای را طراحی می کنیم که اطلاعات را نگه دارد. صفحاتی را تهیه می کنیم که اطلاعات را بگیرد و گزارشاتی را چاپ می کنیم تا اطلاعات را برای کاربران نمایش دهد. به عبارت دیگر ما بر روی اطلاعات متمرکز می شویم و کمتر توجه می کنیم که چه کاری با اطلاعات انجام شده است یا رفتار سیستم چگونه است (نمودارهای ERD را در روش تحلیل سیستم ساخت‌یافته ببینید). این روش Data Senteric یا مبتنی بر داده نامیده شده است و برای ایجاد سیستمهای زیادی تا کنون استفاده شده است(آنتوی و هاملت،2000).

یک چالش بسیار بزرگ که روش مبتنی بر داده با آن روبرو می شود این است که درخواستهای سیستمها معمولا چندین بار تغییر می‌نماید. این سیستمها تغییرات در پایگاه داده را به آسانی پوشش می‌دهند ولی تغییرات در رفتار سیستم را به آسانی نمی توانند پوشش دهند. متد شیء گرایی در پاسخ به این مشکل ایجاد شده است. با متد شیء گرایی هم بر اطلاعات و هم بر رفتار متمرکز می شویم. در نتیجه اکنون می توانیم سیستم هایی را ایجاد نماییم که انعطاف پذیر شده اند تا اطلاعات یا رفتار را تغییر دهند. شی گرایی یک نوع نگرش است که در آن به اجزا سیستم به عنوان اشیا دارای ماهیت مستقل از هم نگاه می-شود، در مقابل تاکید بر روی روالهای و داده های داخلی سیستم تاکید بر روی اشیا موجود در سیستم است. این نگرش در تحلیل سیستم اولین تاثیر خود را می‌گذارد، سپس موجب می‌شود در طراحی سیستم از ابزارهای خاص مربوطه استفاده شود و در نهایت با ابزارهای شیگرایی سیستم پیاده سازی می‌شود. سیستم شی‌گرا سیستمی است که از مجموعه ای از اشیا که در راستای هدف مشترکی با یکدیگر تعامل دارند ساخته می‌شود.

شی (موجود) (Object) : هر چیزی را که بتوانید در نظر بگیرید، یک شی است. البته منظور صرفاً اشیا فیزیکی نیست مثل یک ایده، یا یک مفهوم.

هر شی دو بعد مهم دارد :

1. وجود : هر شیی وجود دارد.

یک شی در یک مقطع از زمان بوجود می آید(توسط اشیا دیگر) و مدتی وجود دارد و در یک مقطع زمانی (توسط خودش و یا اشیا دیگر) نابود می‌شود.

2. ساختار(structure) : مجموعه ای از ویژگیها و خصائص که موجب تمایز بین یک شی با اشیا دیگر می‌گردد.ساختار برای یک شی حکم یک قالب را دارد و حتی در زبانهای شیگرا برای ایجاد اشیا باید از طریق ساختارش اقدام کرد(مثل استفاده از قالب در ریخته گری).نکته خیلی مهم در مورد تعریف ساختار اینست که ساختار مشخص می کند که شی چه ویژگیهایی دارد، اما مشخص نمی‌کند که مقدارهر ویژگی چیست.هر شی یک ساختار درونی و یک ساختار بیرونی دارند. از دید اشیا دیگر ساختار بیرونی هر شی مهم است و ساختار درونی هر شی فقط برای خود شی اهمیت دارد(میشل و مک‌گراو،2008).

اجزای اصلی شی گرایی در زبان برنامه نویسی

1 _ کلاسه کردن اشیاء(وراثت) (inheritance)

2_ کپسوله کردن (Encapsulation)

3_ چند ریختی (Polymorphism)

4_انتزاع (Abstraction)

5_ میانجی (Interface).(همان منبع).

وراثت

وراثت عملی است که یک شی می تواند مشخصه های شی دیگری را بدست آورد . به همین دلیل از مفهوم دسته بندی سلسله مراتبی پشتیبانی می کند. به عنوان مثال سیب قرمز بخشی از دسته بندی سیب که آن هم بخشی از کلاس میوه هاست که آن هم در کلاس بزرگتری به نام غذا قرار دارد . کلاس غذا دارای مشخصات اصلی (خوراکی . پروتئین و غیره) است که به طور منطقی به زیر کلاس های غذا اعمال می شود . بدون استفاده از وراثت هرشی 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 میدهد تا نمونه خاصی از یک حالت عمومی تر باشد . برای مثال C++ برنامه نویسی شی گرا را پیاده سازی می کند . چندین ویژگی در C++ وجود دارد که از کپسوله سازی . چند ریختی و وراثت پشتیبانی می کنند. به خاطر داشته باشید که می توانید از C++ برای نوشتن هر نوع برنامه ای و با استفاده از هر نوع روشی استفاده کنید . این مطلب کهC++ از OOP پشتیبانی می کند بدین معنی نیست که فقط می توانید برنامه های شی گرا بنویسید . همانند پیشینه خود (یعنیc) یکی از قویترین مزایای C++ قابلیت انعطاف آن است. به ارث بردن مشخصات و صفات یکی از مهمترین راهها برای تولید ویژگی های جدید است. مهمترین ویژگی وراثت امکان استفاده مجدد از کارها ی قبلی و ایجاد هماهنگی بین آنهاست(ناتن و شیلد، ترجمه فرسایی،1376).

2_ کپسوله کردن (Encapsulation)

بطور مستقیم نمی‌توان از ساختار درونی یک شی آگاه شد و یا آن را تغییر داد ولی ساختار بیرونی یک شی این امکان را می-تواند فراهم کند. مفهوم Encapsulation نسبت به مخفی کردن اطلاعات شامل این نکته است که میتوان بعضی اطلاعات مخفی شده را از طریق روشهای کنترل شده و مطمئن در اختیار اشیا دیگر قرار داد. در این تعریف بیشتر بر روی امنیت اطلاعات شی تاکید میشود.برای بخاطرسپاردن این مفهوم یک پرنده را در یک قفس تصور کنید، شما آن را میبینید اما نمی توانید آسیبی به پرنده برسانید. تمام برنامه ها از دو عنصور اصلی تشکیل می شوند : عبارت برنامه (کد) و داده ها . کد بخشی از برنامه است که عملیات را اجرا می کند و داده ها اطلاعاتی است که توسط این عملیات تحت تاثیر قرارگرفته . کپسوله سازی یک مکانیزم برنامه نویسی است که کد و داده را با هم در یک جا قرار داده و هر دو را از استفاده نادرست وتداخل خارجی ایمن نگه می دارد(ناتن و شیلد، ترجمه فرسایی،1376).

در یک زبان شی گرا کد و داده ممکن است باهم در چنین روشی 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 از کپسوله سازی پشتیبانی می کند . درون یک شی (کد و داده ها) یا هر دو 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 بخشهای 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(سنیش، ترجمه ریاضی،1384) .

کپسوله سازی روشی است که یک شئ را مستقل از اینترفیسش پیاده سازی کنیم. یک برنامه با یک شئ بواسطه اینترفیسش تعامل می کند، که شامل خصوصیات عمومی و متدهایش است. تا زمانی که این اینترفیس ثابت باقی می ماند، برنامه می تواند به تعامل با کامپوننت ادامه دهد. حتی اگر پیاده سازی اینترفیس کاملا بین دو نسخه کاملا بازنویسی شده باشد. اشیا فقط از طریق متدها و خصوصیات عمومیشان با دیگر شئ ها تعامل می کنند. داده های داخلی یک شئ، نباید در اینترفیس قرار بگیرد. بنابراین فیلدها به ندرت Public تعریف می شوند. به مثال ماشینمان برگردیم: اگر شئ ماشین با شئ راننده تعامل کند، اینترفیس ماشین شاید شامل متدهایGoForward ، Backwardو متد Stop باشد. این همه‌ی اطلاعاتی است که راننده برای تعامل با ماشین نیاز دارد. ماشین شاید شامل شئ" موتور" نیز باشد، اما راننده نیازی به شناخت شئ موتور ندارد. همه اطلاعتی که راننده درباره این متدها دارد این است که می توانند فراخوانی شوند و آنها مقادیر ویژه‌ای را برمی گردانند. بنابراین اگر شئ موتور تغییری کند، تا زمانیکه اینترفیس به درستی به کار خود ادامه می دهد این امر تفاوتی برای راننده ایجاد نمی کند(ماکرجی،2000).

3_ چند ریختی (Polymorphism)

چند ریختی کمیتی است که به یک رابط امکان می دهد تا برای یک کلاس عمومی از عملیات مورد استفاده قرار گیرد . عمل خاص توسط ذات حقیقی شی تعیین می شود . یک مثال ساده از چند ریختی در فرمان اتومبیل است . فرمان اتومبیل برای تمام اتومبیل ها بدون توجه به مکانیزمی که مورد استفاده قرار می دهند یکسان است . فرممان برای اتومبیلی که به صورت دستی کار میکند یا با نیروی برق یا هر چیز دیگری عمل یکسانی را انجام میدهد . بنابراین بعد از اینکه شما چگونگی عمل کردن فرمان را یاد گرفتید می‌توانید هر نوع اتومبیلی را برانید . همین هدف نیز می تواند در برنامه نویسی اعمال شود . به عنوان مثال یک پشته (لیستی که در آن اولین ورودی آخرین خروجی است مثل تعدادی سکه که روی هم قرار می گیرند) را در نظر بگیرید. ممکن است برنامهای داشته باشید که نیاز به سه نوع مختلف پشته داشته باشد یک پشته برای مقادیر اعداد صحیح . یک پشته برای اعداد اعشاری و یک پشته برای کاراکترها مورد استفاده قرار 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 ها ی 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 و آن را برای هر سه نوع به کار ببرید . با این روش بعد از اینکه استفاده از یک پشته را یاد گرفتید می توانید همه انواع آن را به کار ببرید (بیات‌قلی‌لاله،1388).

به طور کلی مفهوم چند ریختی اغلب توسط عبارت " یک رابط. چندین روش " 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 که 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 رابط 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 . به طور کلی می توان گفت که چند شکلی به معنای یک چیز بودن و چند شکل داشتن است. به طور مثال شما می توانید در را باز کنید ، پنجره را باز کنید ، یا یک حساب در بانک باز کنید. در مدل شیء گرا کلاسهای متفاوتی خواهیم داشت که همگی متدی به نام "بازکردن" دارند، ولی هر کلاس خودش می داند که باید چگونه عملیات "بازکردن" را انجام دهد. چند شکلی به مدل سازان این امکان را می دهد تا با مشتریان با زبان و اصطلاحات خودشان صحبت کنند(شیوانوویچ و همکاران، 1996).

چند شکلی توانایی کلاسهای متفاوت، در پیاده سازی های مختلف از اینترفیسهای عمومی مشابه است. به عبارت دیگر، چندشکلی به متدها و خصوصیات یک شئ اجازه می دهد، بدون توجه به چگونگی پیاده سازی اعضای آنها، فراخوانی شوند. برای مثال شئ Driver می تواند بوسیله اینترفیس عمومی Car با شئ Car تعامل کند.اگر شئ دیگری مانند شئ Truck یا شئ SportCar اینترفیس عمومی مشابهی را داشته باشند، شئ Driver می تواند با آنها بدون توجه به پیاده سازی خاص آن اینترفیس تعامل کند این جا دو راه اصلی برای تامین چندشکلی وجود دارد: چندشکلی اینترفیس interface).

4_انتزاع (Abstraction)

به کلاسی مجرد گفته می شود که پیاده سازی متدها در آن انجام نمی شود! بر خلاف انسانها که مجرد تعریف دیگری برایشان دارد! حال سئوالی پیش می آید که اگر کلاسی داشته باشیم که نخواهیم پیاده سازی متدها را در آن انجام بدیم. برای اجرای پروژه از برنامه نویسان مختلفی استفاده می کنید که ممکن است همه آنها هموطن نباشند، اگر قرار باشد هر برنامه نویسی در نامگذاری متدها و کلاسهایش آزاد باشد، در کد نویسی هرج و مرج به وجود می آید. شما به عنوان مدیر پروژه، کلاسی تعریف

می کنید که در آن تمام متدها با ورودی و خروجی هایشان مشخص باشند. 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 را که می نویسند از این کلاس شما به ارث ببرند و متدها را به طور دلخواه پیاده سازی کنند. این باعث می شود که شما با داشتن یک کلاس، ورودی و خروجی های مد 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(مجرد ، انتزاعی) استفاده می کنیم. فیلدهایی که می خواهیم در کلاسهای مشتق شده از این کلاس پیاده سازی شوند حتما باید با abstract تعریف شوند. یک کلاس مجرد می تواند فیلدها و متدهای نامجرد داشته باشد. اگر متد نامجردی 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. از دید انتزاعی، زمان را نیز می توان بعنوان یک شیء در نظر گرفت. صفات زمان، ساعت، دقیقه و ثانیه هستند و گذشت زمان، رفتار آن است. در ایجاد شیء هیچ محدودیتی وجود ندارد و همه چیز به تخیل شما باز می‌گردد.

5_ میانجی (Interface)

رویه (Interface)

به ساختار بیرونی یک شی رویه(interface) گفته می‌شود. به ساختار دورنی پیاده سازی رویه شی گفته می‌شود.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 index ها هستند که در کنار هم جمع شده اند. اینترفیس ها را نمی توان Instantiate (وهله سازی) کرد . تنها چیزی که یک اینترفیس دارا می باشد امضای (signature) تمامی اعضای آن می باشد. به این معنی که ورودی و خروجی متدها، نوع Property(خاصیت) ها و... در آن تعریف می شوند ولی چیزی پیاده سازی نمی شود. اینترفیس ها سازنده و فیلد ندارند . یک اینترفیس نمی تواند Overload Operator داشته باشد و دلیل آن این است که در صورت وجود این ویژگی، احتمال بروز مشکلاتی از قبیل ناسازگاری 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(وبلیوبینور، ترجمه اسماعیل‌زاده قلزم، 1386).

نتیجه گیری

برنامه نویسی شئ گرا یک روش جدید برنامه نویسی می باشد که در آن علاوه بر ویژگی ساخت یافته بودن برنامه از چند ویژگی قوی جدید استفاده می شود. در برنامه نویسی شی گرا از توانایی عجیب انسان در انتزاع برای مدل سازی اشیای دنیای واقعی در مفاهیم موجود در زبان برنامه نویسی استفاده می شود. همچنین صفات و اطلاعات یکی شی از دید سایر اشیاء و اجزاء مخفی باشد و ارتباط از طریق ارسال پیام صورت می‌گیرد. جمع بندی حاصل از تحقیق نشان می دهد که در حال حاضر برنامه نویسی شی‌گرا یکی از بهترین وسیله ها برای برنامه‌ریزی مدیریت بهینه در زمینه‌های مختلف است. همچنین از برنامه شی‌گرا جهت دستیابی به دقت بالا و دستیابی به اطلاعات دقیق استفاده کرد. برنامه نویسی شی گراء، بدون شک مهمترین تحول در دنیای نرم افزار طی سالیان گذشته بوده که بیشترین تاثیر را در پیشبرد نرم افزار بدنبال داشته و ما امروز در صنعت گسترده نرم افزار از دستاوردهای وسیع آن بهره مند هستیم .

منابع

- شجاعی، رشید، برنامه سازی پیشرفته – تحلیل شی گرا و برنامه نویسی شی گرا، مجله فناوری اطلاعات،1389.

- کربلایی، آرش، آزمون خودکار نرم‌افزارهای شی گرا با استفاده از ضوابط UML ، پایان نامه کارشناسی ارشد، دانشگاه تربیت مدرس،1383.

- ناتن، پاتریک؛ شیلد، هربرت. آموزش زبان برنامه نویسی جاوا، ترجمه داریوش فرسایی، نشر اسحاق، 1376.

- بیات قلی لاله، زهر، معمای شی گرایی در C#، نشر اسحاق، تهران،1388.

- وبلیونور،رابرت. مرجع کامل برنامه نویسی شیء گرا با++C ، ترجمه حسین اسماعیل راده قلزم، نشر سیمای دانش، تهران،1386.

- شنیس، آنتونی. آموزش برنامه نویسی شیء گرا در ۲۱ روز، ترجمه عباس ریاضی، نشر نص، تهران،1384.

- Edvardsson J. M., Kamkar,M.(1998). "Analysis of the Constraint Solver in UNA Based Test Data Generation", 9th ACM SIGSOFT International Symposium on the Foundations of. Software Engineering (FSE-9), pp.237 - 245.

- Elrad. T., Filman R.E., A. Bader,A.(2001). Aspect-oriented programming :Introduction, Communication of the ACM, 44(10), pp.24-32.

-Chen H. Y. T., Tse,H.,Deng,T.. (2000). ROCS: "an object-oriented class-level testing system based on the relevant observable contexts technique, Information and Software Technology", 42(10):PP.677-686.

-Ahuja, M. K., and Carley, K. M. (1999). Network structure in virtual organizations. Organization Science 10 (6): 741-757.

-Alkadi, I., Alkadi, G., and Totaro, M. (2003). Effects of information technology on the business world. Human Systems Management 22: 99-103.

-Cooper, W. W., and Muench, M. L. (2000). Virtual organization: Practice and the literature. Journal of Organizational Computing & Electronic Commerce 10 (3): 89-209.

- Fraikin F. Leonhardt,T.(2002). "SeDiTeC-testing based on sequence diagrams", International Conference on Automated Software Engineering, pp.261-266.

-Hutchinson, M. F. (1996). A locally Adaptive Approach to the Interpolation of Digital Elevation, National Center Information andAnalysis.

- Michael C. C. and McGraw G.(2008). "Automated software test data generation for complex programs"., Proceedings of Automated Software Engine ering , pp.136 -146.

- Antoy S.,. Hamlet.,D.(2000). "Automatically Checking an Implementation against Its Formal Specification", IEEE Transactions on Software Engineering, 26(1),pp.55-69.

- Offutt, J. Abdurazik, A.(2000). "Using UML Collaboration Diagrams for Static Checking and Test Generation". UML2000 – The Unified Modeling Language, Advancing the Standard, Third International Conference, Springer vol. 1939, pp.385-395.

-Sehanovic, J., Zugaj, M.(1996). "mathematical modeling of organization and information technology", Library management , pp.25-30, MCB university press.

-Makhreji, A.(2000). "the evolution of information systems: their impact on organizations and structures", Emerald ,PP:407-507.