بخش چهارم: مهندسی نرم افزار

فصل ۱۷ اصول و مفاهیم تحلیل شیء گرا
فصل ۱۸ مدلسازی تحلیل شیء گرا
فصل ۱۹ طراحی شیء گرا
فصل ۲۰ آزمون نرم افزار و راهکارها

فصل ۱۷: اصول و مفاهیم تحلیل مهندسی نرم افزار شیء گرا

ما در جهانی از اشیاء زندگی می‌کنیم. این اشیاء در طبیعت، در نهادهای ساخت دست بشر، در تجارت و در محصولاتی که استفاده می‌کنیم، وجود دارند. آنها را می‌توان بسته‌بندی کرد، توصیف نمود، سازماندهی کرد، ترکیب کرد، دستکاری کرد و ایجاد نمود. بنابراین، تعجبی ندارد که برای ایجاد نرم‌افزارهای کامپیوتری نیز دیدگاهی شیءگرا پیشنهاد شود- این شکل انتزاعی ما را قادر می‌سازد تا جهان را به شیوه‌ای مدلسازی کنیم که بهتر قابل درک و کاوش باشد.

روش شیءگرا در توسعهٔ نرم‌افزار اولین بار در اواخر دههٔ ۱۹۶۰ برای توسعهٔ نرم‌افزار به کار گرفته شد. ولی ۲۰ سال طول کشید تا فنآوری شیءگرا به طور گسترده مورد استفاده قرار گیرد. در سرتاسر دههٔ ۱۹۹۰، مهندسی نرم‌افزار شیءگرا الگوی انتخابی بسیاری از نرم‌افزار نویسان شد و تعداد فزاینده‌ای از سیستم های اطلاعاتی و مهندسان حرفه‌ای به آن روی آوردند. به مرور زمان، فنآوری های شیءگرا جایگزین روش های کلاسیک توسعهٔ نرم‌افزار می‌شوند. سوال مهم این است: چرا؟

پاسخ این سؤال (همانند پاسخ بسیاری از سوالات دیگر در مهندسی نرم‌افزار) پاسخ ساده‌ای نیست. برخی استدلال می‌کنند که نرم‌افزارنویسان حرفه‌ای صرفاً به دنبال یک روش جدید بودند، ولی این دیدگاه بیش از حد ساده‌نگرانه است. فنآوری های شیءگرا به چندین مزیت ذاتی منجر می‌شوند که هم در سطح مدیریتی و هم فنی مزایایی به همراه دارد.

فنآوری های شیءگرا منجر به استفاده مجدد می‌شود و استفاده مجدد (از مؤلفه‌های برنامه) منجر به توسعه سریعتر نرم‌افزارها و برنامه‌هایی با کیفیت بالاتر می‌شود. نگهدرای نرم‌افزارهای شیءگرا آسانتر است زیرا ساختار آن ذاتاً فاقد پیوستگی است. این موضوع، به هنگام اعمال تغییرات، اثرات جانبی کمتری به وجود می‌آورد و برای مهندس نرم‌افزار و مشتری دردسر کمتری ایجاد می‌کند. به علاوه، تطبیق دادن و تغییر دادن اندازهٔ سیستم های شیءگرا آسانتر است (یعنی سیستم های بزرگ را می‌توان با مونتاژ کردن زیرسیستم های قابل استفاده مجدد ایجاد کرد).

سال ها بود که اصطلاح شیءگرا (OO) برای مشخص کردن روشی به کار می‌رفت که در آن از زبانهای برنامه‌نویسی شیءگرا (مثل اِدا ۹۵، جاوا، C++، ایفل و اسمالتاک) استفاده می‌شود. امروزه الگوی OO شامل دیدگاهی کامل از مهندسی نرم‌افزار می‌شود. Berar به این نکته چندین اشاره دارد [BER93]:

مزایای فنآوری شیءگرا اگر به طور زودهنگام و در سرتاسر فرآیند نرم‌افزار به آن پرداخته شود. بهبود می‌یابد. آنها که به فنآوری شیءگرا روی می‌آورند، باید تأثیر آن را بر کل فرآیند مهندسی نرم‌افزار مورد سنجش قرار دهند. فقط استفاده از برنامه‌نویسی شیءگرا (OOP) نیست که بهترین نتایج را ببار دهد. مهندسان نرم‌افزار و مدیران آنها باید چنین عناصری را به عنوان تحلیل نیازهای شیءگرا (OORA)، طراحی شیءگرا (OOD)، تحلیل دامنه شیءگرا (OODA)، سیستم های بانک اطلاعاتی شیءگرا (OODBMS) و مهندسی نرم‌افزار شیءگرا به کمک کامپیوتر (OOCASE) در نظر بگیرند.

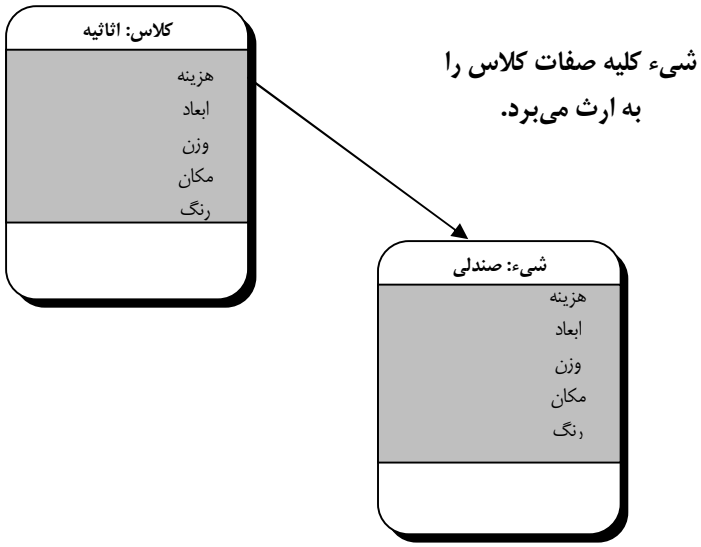
مفاهیم و قواعد کلی شیء گرایی

برای درک موضوع از دیدگاه شیء گرایی ، مثالی از یکی از اشیای دنیای واقعی – یعنی صندلی – را در نظر بگیرید.

- صندلی یک **عضو**(member) یا یک **نمونه**(instance) از یک کلاس بزرگتر از اشیاء بنام اثاث خانه است.
- در کلاس اثاث خانه مجموعه ای از **صفات**(attributes) کلی و مشترک به تمام اشیای این کلاس نسبت داده می شود. بعنوان مثال تمام اثاث خانه قیمت، ابعاد، وزن، محل، رنگ و خیلی صفات احتمالی دیگر دارند.
- چون صندلی عضوی از اثاث خانه است، تمام صفاتی را که برای کلاس اثاث خانه تعریف شده است به **ارث** می برد(inherits).

کوشش کرده‌ایم تا تعریفی حکایت‌وار از کلاس را با توصیف صفات آن ارایه کنیم. ولی چیزی کم است. هر کدام از اعضای کلاس اثاثیه را می‌توان به چندین شیوه دستکاری کرد. می‌توان آن را خرید، فروخت، تغییر فیزیکی در آن ایجاد کرد (مثلاً پایه‌ها را اره کرد یا آن را ارغوانی رنگ کرد) یا از مکانی به مکان دیگر جابجا کرد. هر یک از **عملیات** (یا **متدها** یا **سرویس ها**) یک یا چند صفت شیء را تغییر می‌دهند. برای مثال، اگر صفت مکان یک عنصر داده‌ای مرکب به صورت زیر باشد:

اتاق + طبقه + ساختمان= مکان



شکل ۱ : وراثت بین کلاس ها

در این صورت، عملی که جابجایی نام دارد، یک یا چند مورد از این عناصر داده‌ای (ساختمان، طبقه، اتاق) را که مکان را تشکیل می‌دهند، تغییر می‌دهد. برای انجام این کار، عمل جابجایی باید از این عناصر داده‌ای داده‌ای **آگاه** باشد. مادامی که صندلی و میز هر دو نمونه‌های کلاس اثاثیه باشند، از عمل جابجایی می‌توان برای آنها استفاده کرد. همهٔ عملیات معتبر (مثل خریدن، فروختن، توزیع کردن) برای کلاس اثاثیه به تعریف شیء **متصل** هستند و برای کلیه نمونه‌های این کلاس به ارث گذاشته می‌شوند.

شیء صندلی (و کلاً همهٔ اشیاء) داده‌ها (مقادیر صفاتی که صندلی را تعریف می‌کنند)، عملیات (عملیاتی که برای تغییر دادن صفات صندلی به کار می‌روند)، اشیای دیگر (اشیای مرکبی که قابل تعریف هستند)، ثابت ها(مقادیر ثابت) و اطلاعات مربوطه دیگر را **بسته‌بندی** می‌کنند. بسته‌بندی بدان معنا است که کلیهٔ این اطلاعات تحت یک نام بسته‌بندی شوند و به عنوان یک مشخصه یا قطعه برنامه به کار برده شوند.

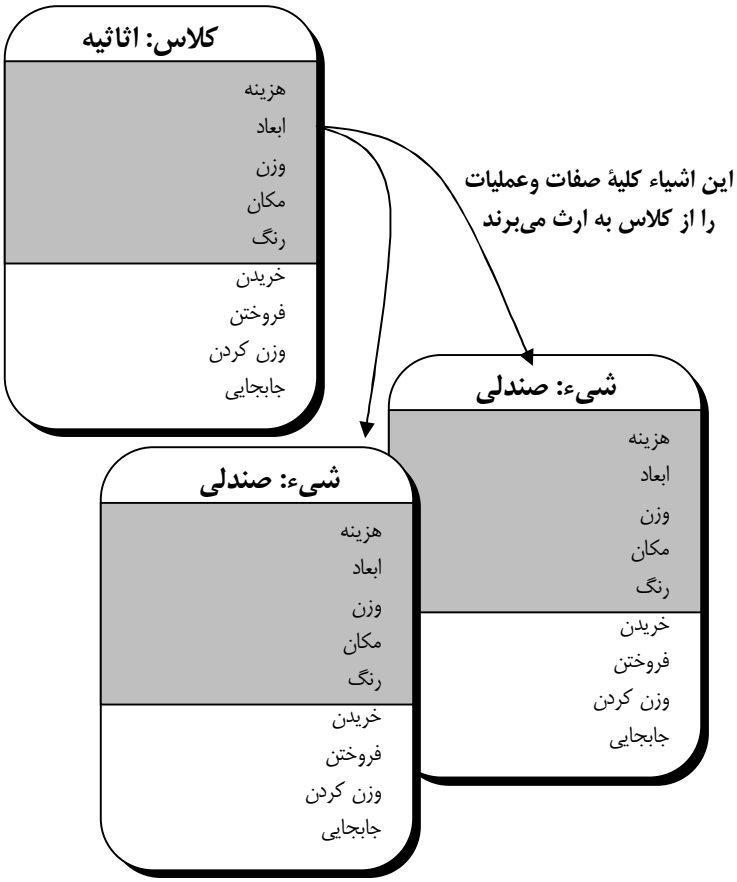
اکنون که با چند مفهوم اساسی آشنا شدیم، تعریفی رسمی‌تر از شیءگرایی، بی‌مناسبت نخواهد بود. Coad و Yourdon [COA91] این اصطلاح را چنین تعریف می‌کنند:

ارتباطات + وراثت + طبقه‌بندی+ اشیاء= شیءگرایی

بنابراین هر شیء از کلاس **اثاث خانه** به روش های مختلف قابل تغییر است : خریده شود ، فروخته شود یا از یک مکان به مکان دیگر منتقل شود.

- هر کدام از این عملیات(operations) یا خدمات(services) یا متدها(methods) یک یا چند صفت از صفات شیء را تغییر می دهند.

- **اشیا** در درون خود داده ها (مقادیر صفات)، عملیات (اعمالی که بر شیء وارد می شوند تا صفات آن را تغییر دهد)، اشیای دیگر(اشیای ترکیبی)، ثابت ها و سایر اطلاعات مربوط را بسته بندی می کنند.

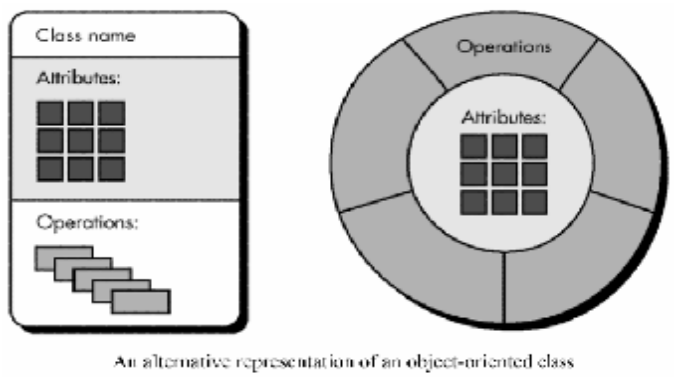


شکل ۲ : ارث بری عملیات از کلاس به عملیات

کلاس ها و اشیاء

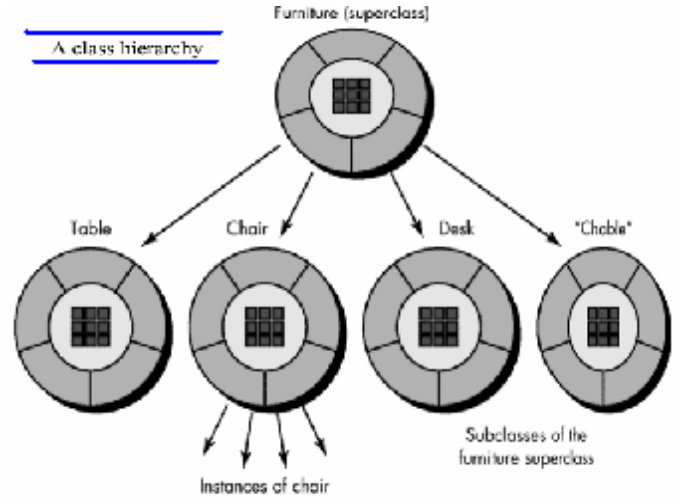
کلاس یک مفهوم شی گرایی است که داده ها و رویه هایی را که برای توصیف محتوا(content) و رفتار(behaviour) یک موجودیت دنیای واقعی لازم است ، بسته بندی می کند.

انتزاع داده ای(صفات) که کلاس را توصیف می کند در میان یک "دیوار" به نام **انتزاع رویه ای**(که **عملیات**، **خدمات** یا **متدها** نامیده می شود) قرار گرفته اند که قادر به تغییر داده ها در بعضی موارد است.



شکل ۳ : نمایشی از کلاس ها

تنها راه برای دسترسی به صفات از طریق یکی از متدهاست(کلاس داده ها و متدها را **بسته بندی می کند**).
این کار باعث **پنهان سازی اطلاعات(Information Hiding)** می شود و باعث کمترشدن تاثیر اثرات جانبی مرتبط با تغییرات می شود.
از آنجایی که متدها تعداد محدودی از صفات را تغییر می دهند، آنها **منسجم(Cohesive)** هستند؛ و بخاطر اینکه ارتباطات فقط از طریق متدها رخ می دهد ، کلاس از سایر اجزای سیستم **جدا(Decoupled)** می شود. این خصوصیات منجر به شکل گیری یک **نرم افزار با کیفیت بالا(High-Quality Software)**می شود.
زبر کلاس (Superclass) مجموعه ای از کلاس هاست، و **زیر کلاس(Subclass)** یک نمونه خاص از یک کلاس است.
این تعاریف دلالت بر وجود **سلسله مراتب کلاس(** Class Hierarchy) می کند که در آن صفات و عملیات زبر کلاس بوسیله زیر کلاس ها به ارث برده می شوند. البته هر کدام از این زیر کلاس ها ممکن است داده ها و متدهای "خصوصی" داشته باشند.



شکل ۴ : نمایشی از سلسه مراتب کلاس ها

صفات

پیش از این دیدیم که صفات به کلاس ها واشیاء متصل هستند و شیء یا کلاس را به نحوی توصیف می‌کنند. بحثی دربارهٔ صفات توسط de Champeaux و همکاران وی [CHA93] ارایه شده است:

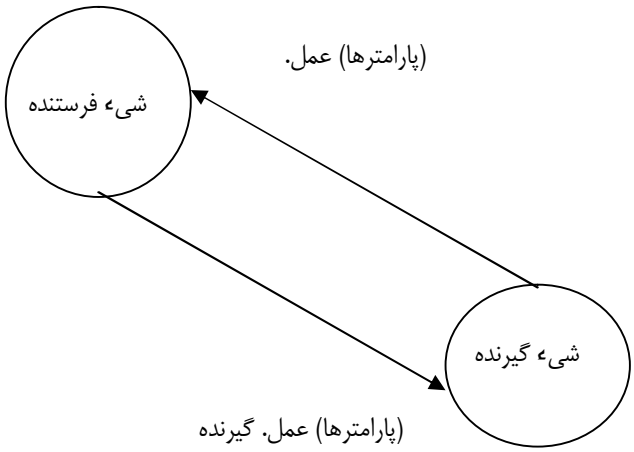
نهادهای موجود در زندگی واقعی، غالباً با واژه‌هایی توصیف می‌شوند که نشانگر ویژگی های پایدارند. اکثر اشیای فیزیکی دارای ویژگی هایی از قبیل رنگ، شکل، وزن و جنس مواد هستند. افراد دارای ویژگی هایی مثل تاریخ تولد، والدین، نام و رنگ چشم هستند. هر ویژگی را می‌توان به عنوان رابطه‌ای دودویی میان یک کلاس و یک دامنه معین در نظر گرفت.

- صفات به کلاس ها و اشیا متصل هستند، و آنها کلاس یا شی را به نوعی تعریف می کنند.
- یک صفت مقدار خود را از یک میدان مقادیر می گیرد. در اکثر حالات، دامنه فقط مجموعه ای از مقادیر است. مثال: دامنه مقادیر برای رنگ عبارتند از { سفید ، سیاه ، نقره ، خاکستری ، آبی ، ... } .
- در حالات پیچیده تر، دامنه می تواند مجموعه ای از کلاس ها باشد. مثال: دامنه مقادیر برای کلاس خودرو عبارت است از { ۴ سیلندر، ۶ سیلندر، ۸ سیلندر، ۱۰ سیلندر، ۲۴ سیلندر، و ... } .

عملیات، متدها و خدمات

یک شیء، داده‌ها را (که به صورت مجموعه‌ای از صفات نمایش داده می‌شود) و الگوریتم هایی که این داده‌ها را پردازش می‌کنند، بسته‌بندی می‌کند. این الگوریتم ها را که عملیات، متدها یا سرویس ها می‌نامند، می‌توان از دیدگاه سنتی به عنوان پیمانه در نظر گرفت.

هر کدام از این عملیات که در شیء بسته بندی شده است، در واقع یک نوع نمایش برای یکی از رفتارهای شیء را تامین می کند.



شکل ۵ : مبادله پیام ها بین اشیاء

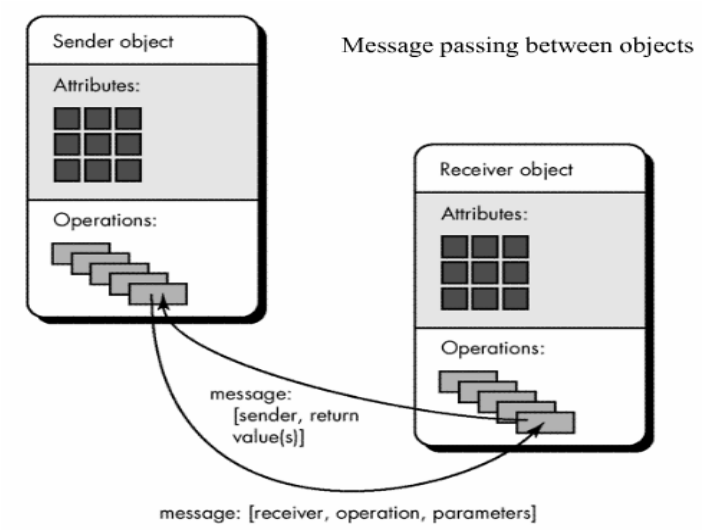
پیغام ها (Messages)

پیغام ها ابزارهای تعامل اشیاء هستند. با استفاده از فناوری معرفی شده در بخش قبل، پیغام باعث برانگیختن رفتاری خاص در شیء گیرنده می‌شود. این رفتار زمانی مشاهده می‌شود که عملی اجرا شود. بنابراین:

- پیغام ها ابزاری هستند که اشیا از طریق آنها با یکدیگر ارتباط برقرار می کنند.
- عملیات موجود در شی فرستنده پیغامی به فرم زیر تولید می کند:
- پیغام : [مقصد ، عملیات ، پارامترها]

که **مقصد** شی گیرنده را – که با دریافت پیغام فعال می شود – تعریف می کند ، **عملیات** به عملیاتی اشاره می کند که پیغام را قرار است دریافت کند، و **پارامترها** اطلاعاتی را تامین می کنند که برای درست انجام شدن عملیات لازمند.

شی گیرنده با انتخاب عملیاتی که نام پیغام را پیاده سازی می کند، اجرای آن، و بازگرداندن کنترل به فراخواننده؛ به پیغام پاسخ می دهد.



شکل٦ : مبادله پیام ها بین دو شیء

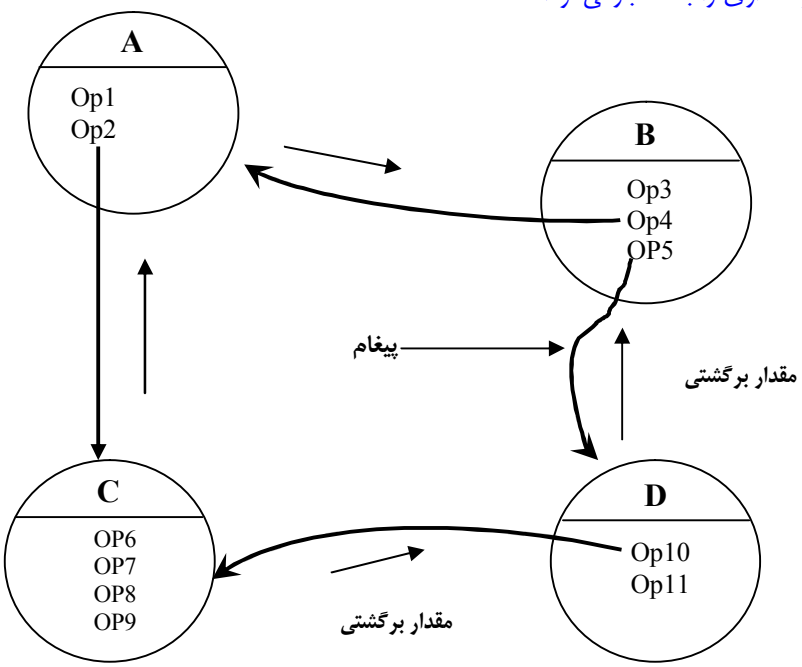
به عنوان مثالی از مبادله پیغامها در داخل یک سیستم OO، اشیای شکل ۷ را در نظر بگیرید. چهار شیء A، B، C و D با مبادله پیغام با یکدیگر ارتباط برقرار می‌کنند. برای مثال، اگر شیء B بخواهد عملیات op10 از شیء D را اجرا کند، پیغامی به شکل زیر به D ارسال می‌کند:

D . op10 (data)

شیء D نیز به عنوان بخشی از اجرای op10 ممکن است پیغامی به شکل زیر به C بفرستد:

C . op08 (data)

C عمل op08 را می‌یابد، آن را اجرا می‌کند، و سپس یک مقدار بازگشتی مناسب به D ارسال می‌کند. عمل op10 کامل می‌شود و مقداری را به B بازمی‌گرداند.



شکل۷ : مبادله پیام ها بین اشیاء

بسته بندی (Encapsulation)

گرچه ساختار و اصطلاحات معرفی شده در بخش های قبل سیستم های OO را از همتاهای سنتی آنها متفاوت می‌سازد، سه ویژگی سیستم های شیءگرا آنها را منحصر به فرد می‌سازد. چنان که پیش از این نیزگفتیم، کلاس و اشیایی که از آن ایجاد می‌شوند، داده‌ها و عملیاتی را که بر روی آنها عمل می‌کنند، یک جا **بسته‌بندی** می‌کنند.

مزایای بسته بندی

۱. جزییات داخلی پیاده سازی داده ها و رویه ها از دنیای خارج پنهان هستند (پنهان سازی اطلاعات). این باعث می شود که گسترش اثرات جانبی در هنگام تغییرات کم شود.
۲. ساختارهای داده و عملیاتی که آنها را تغییر می دهند در یک موجودیت تکی به نام کلاس با یکدیگر ادغام می شوند. این باعث می شود که استفاده مجدد از مولفه (Component) راحت تر شود.
۳. واسط های بین اشیای بسته بندی شده ساذه تر می شوند. شیء فرستنده یک پیغام لازم نیست که به جزییات داخلی ساختارهای داده توجه کند. این باعث می شود که اتصال(Coupling) سیستم کاهش یابد.

وراثت (Inheritance)

وراثت یکی از مهمترین تفاوت ها بین سیستم های سنتی و سیستم های شی گراست. یک زیر کلاس Y تمام صفات و عملیات را از زبر کلاس خود X به ارث می برد. این بدین معنی است که تمام ساختارهای داده و الگوریتم هایی که در ابتدا برای X طراحی و پیاده سازی شده اند ، اکنون برای Y نیز موجود می باشند.
هر تغییر در داده ها یا عملیات موجود در زبرکلاس سریعا بوسیله تمام زیرکلاسهایی که از آن زبرکلاس ارث می برند، به ارث برده می شود. بنابراین سلسله مراتب کلاس تبدیل به مکانیزمی شده است که بوسیله آن تغییرات (در سطوح بالا) سریعا می توانند در سیستم پخش شوند.

- **گزینه های لازم برای تولید یک کلاس جدید**

۱. می توان یک کلاس را از ابتدا طراحی و ساخت در این حالت از وراثت استفاده نمی شود.
۲. سلسله مراتب کلاس را جستجو کنیم تا کلاسی بالاتر را در سلسله مراتب پیدا کنیم که اکثر صفات و عملیات مورد نیاز را داشته باشد. کلاس جدید از کلاس بالاتر ارث می برد و در صورت نیاز صفات و عملیات جدید را اضافه می کنیم.
۳. سلسله مراتب کلاس را دوباره سازماندهی کنیم تا صفات و عملیات مورد نیاز قابل ارث بری شوند.
۴. خصوصیات یک کلاس موجود را می توان دوباره نویسی (Override) کرد و نسخه های خصوصی صفات و عملیات را برای این کلاس جدید پیاده سازی کرد.

چند ریختی (Polymorphism)

چند ریختی باعث می شود که چندین عملیات بتوانند از یک نام استفاده کنند. این کار باعث می شود که تعداد خطوط برنامه برای پیاده سازی کاهش یابد و اعمال تغییرات را تسهیل کنند.

- برای درک چند ریختی ، یک برنامه سنتی(رویه ای) را در نظر بگیرید که چهار نوع گراف را باید بکشد : گراف‌های خطی، pie charts ، histograms و Kiviat diagrams . ایده آل این است که هنگامی که داده ها برای یک نوع خاص گراف فراهم شد، گراف باید خودش را بکشد. برای انجام این کار در یک برنامه سنتی(رویه ای) باید برای هر نوع، یک پیمانه را در نظر بگیریم و توسعه دهیم.

- سپس، در داخل طراحی هر نوع گراف، منطق کنترلی مشابه زیر باید تعبیه شود :

```
case of graphtype:
    if graphtype = linegraph then DrawLineGraph (data);
    if graphtype = piechart then DrawPieChart (data);
    if graphtype = histogram then DrawHisto (data);
    if graphtype = kiviat then DrawKiviat (data);
end case;
```

برای حل این مشکل، تمام گراف ها را زیر کلاس هایی از یک کلاس کلی به نام **graph** قرار می دهیم. با استفاده از مفهوم overloading هر زیر کلاس یک عملیات به نام draw تعریف می کند. یک شی می تواند پیغام draw را به هریک از اشیایی که از هر یک از زیر کلاس ها نمونه سازی (instantiate) شده است بفرستد. شیئی که پیغام را دریافت می کند، عملیات draw مربوط به خود را فراخوانی می کند تا گراف مناسب را تولید کند.

شناسایی عناصر یک مدل شیء گرا

عناصر مدل تحلیل شیء گرا شامل موارد زیر است:

- شناسایی کلاس ها و اشیاء
- مشخص کردن صفات
- تعریف عملیات
- به پایان رساندن تعریف اشیاء

در ادامه به بحث در مورد هری‌ک از این عناصر خواهیم پرداخت.

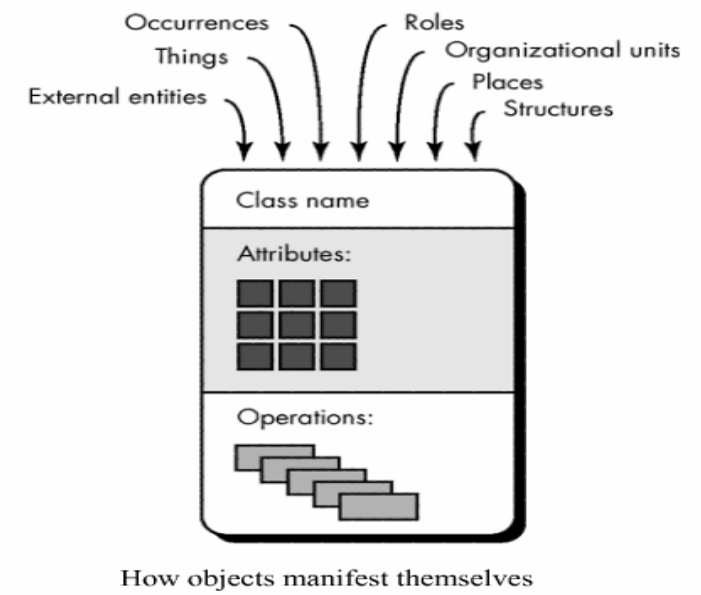
شناسایی کلاس ها و اشیاء

شناسایی اشیاء را با بررسی بیان مساله یا (با استفاده از اصطلاحی که در فصل ۱۲ معرفی شد) با اجرای **تجزیه گرامری** روی شرح پردازشی سیستمی که قرار است ساخته شود، آغاز می‌کنیم. برای تعیین اشیاء، زیر اسم ها یا عبارات اسمی خط کشیده، آنها را در جدولی قرار می‌دهیم. به اسم های مترادف نیز باید توجه داشت. اگر یک شیء برای پیاده‌سازی یک راهکار، مورد نیاز باشد، در آن صورت بخشی از فضای حل را تشکیل می‌دهد؛ در غیر این صورت، اگر شیء فقط برای توصیف راهکار لازم باشد، بخشی از فضای مساله به شمار می‌رود. ولی هنگامی که کلیه اسم ها را مشخص کردیم، چه باید بکنیم؟

اشیاء می توانند یکی از طرق زیر خود را نشان دهند:

۱. **موجودیت های خارجی**(مثلا سیستم های دیگر، دستگاه ها ،افراد) که اطلاعات مورد استفاده سیستم کامپیوتری را تولید یا مصرف می کنند.
۲. **چیزهایی**(مثلاً گزارش ها، صفحات نمایش، نامه ها، علایم الکترونیکی) که بخشی از دامنه اطلاعاتی مسأله به شمار می روند.
۳. **رخدادها** یا **وقایعی**(مثلاً کامل شدن حرکات روبات) که در حیطه عملکرد سیستم رخ می دهند.
۴. **نقش هایی**(مثل مدیر، مهندس، فروشنده) که افراد در حال تعامل با سیستم، بر عهده دارند.
۵. **واحدهای سازمانی**(مثل بخش، گروه ، تیم) که با یک کاربرد خاص سروکار دارند.
۶. **مکان هایی**(مثل سالن تولید یا بارانداز) که حیطه مسأله و وظیفه کلی سیستم را مشخص می کند.

۷. **ساختارهایی**(مثل حسگرها، وسایل نقلیه چهار چرخ یا کامپیوترها) که کلاسی از اشیاء یا کلاس های مرتبطی از اشیاء را تعریف می کنند.



شکل۸ : چگونگی نشان دادن اشیاء

توجه به این نکته نیز مهم است که اشیاء چه چیزهایی نیستند. به طور کلی یک شیء هیچگاه نباید یک "نام رویه ای دستوری" داشته باشد. یک تجزیه گرامری را می توان برای تفکیک اشیاء (اسامی) و عملیات به کار برد.

- شش خصوصیت گزینشی که تحلیلگر باید در مورد هر شیء بالقوه ای که در مدل تحلیل بکار می برد در نظر بگیرد :

۱. **اطلاعات ذخیره شده**– شیء بالقوه فقط در صورتی هنگام تحلیل مفید خواهد بود که برای عملکرد سیستم به اطلاعات آن نیاز باشد.

۲. **خدمات مورد نیاز**– شیء بالقوه باید مجموعه ای از عملیات قابل شناسایی داشته باشد که می توانند صفات آنرا به طریقی تغییر دهند.

۳. **صفات چندگانه**– یک شیء با یک صفت ممکن است در طول طراحی مفید باشد ولی ممکن است بهتر باشد در هنگام تحلیل به عنوان صفتی از شیء دیگر منظور شود.

۴. **صفات مشترک**– مجموعه ای از صفات را می توان برای یک شیء بالقوه تعریف کرد و این صفت ها در تمام نمونه های شیء بکار می روند.

۵. **عملیات مشترک**– مجموعه ای از عملیات را می توان برای یک شیء بالقوه تعریف کرد و این عملیات در تمام نمونه های شیء بکار می روند.

۶. **نیازهای ضروری**– موجودیت های خارجی که در فضای مسأله ظاهر می شوند و اطلاعات ضروری برای کارکرد هر راهکار را تولید یا مصرف می کنند، باید به عنوان شیء در مدل نیازها تعریف شوند.

مشخص کردن صفات

- صفات شیء آن را طوری تعریف و توصیف می کنند که برای ورود به مدل تحلیل انتخاب شده است.

- برای توسعه یک مجموعه بامعنی از صفات، تحلیلگر می تواند شرح پردازش مسأله را مطالعه کند و چیزهایی را انتخاب کند که به طور منطقی به شیء تعلق دارند.
- به علاوه باید به این سؤال پاسخ داد: «چه عناصر داده ای (مرکب ویا ساده) این شیء را در حیطه مسأله به طور کامل تعریف می کنند؟»

تعریف عملیات

عملیات، رفتار یک شیء را تعریف می کنند و صفات آنرا به طریقی تغییر می دهند. به طور مشخص، یک عمل مقدار یک یا چند صفت موجود در شیء را تغییر می دهد. یک عمل باید از ماهیت صفات شیء آگاه باشد و باید به شیوه ای پیاده سازی شود که دستکاری داده های به دست آمده از صفات را میسر سازد.

- **رده بندی عملیات**

۱. عملیاتی که با داده ها را به طریقی کار می کنند(مثلاً اضافه کردن، حذف کردن، قالب بندی دوباره و گزینش)،
۲. عملیاتی که محاسبه ای را انجام می دهند، و
۳. عملیاتی برای رخ دادن یک رویداد کنترلی بر شیء نظارت می کنند.

نهایی سازی تعریف اشیاء

تعریف عملیات، آخرین مرحله در کامل کردن تشخیص اشیاء است. عملیات اضافه ای را می توان با در نظر گرفتن **تاریخچه حیات** یک شیء و پیغام هایی که در میان اشیاء تعریف شده در سیستم مبادله می شوند، تعیین کرد. تاریخچه حیات کلی یک شیء را می توان با در نظر گرفتن این نکته تعیین کرد که شیء باید به شیوه های دیگری ایجاد، اصلاح، دستکاری یا خوانده شود و احتمالاً حذف شود.

مدیریت پروژه های نرم افزاری شیءگرا

- **<u>فعالیت های مدیریت مدرن پروژه نرم افزاری</u>**

۱. ساختن یک چهارچوب فرایند مشترک برای پروژه.
۲. استفاده از چهارچوب و معیارهای تاریخی برای براورد کار و زمان لازم.
۳. ایجاد نقاط عطف که اندازه گیری پیشرفت کار را میسر می کنند.
۴. تعریف یکسری نقاط کنترلی برای مدیریت ریسک، تضمین کیفیت و کنترل.
۵. مدیریت تغییراتی که به طور ذاتی در پیشرفت پروژه رخ می دهند.
۶. پیگیری، نظارت و کنترل پیشرفت.

در ادامه عناوین زیر را بررسی می کنیم:

- **چهارچوب فرایند مشترک برای OO**
- **معیارها و برآورد پروژه شیءگرا**
- **یک رهیافت براورد و زمانبندی OO**
- **پیگیری پیشرفت برای یک پروژه شیءگرا**

چهارچوب فرایند مشترک(Common Process Framework-CPF)

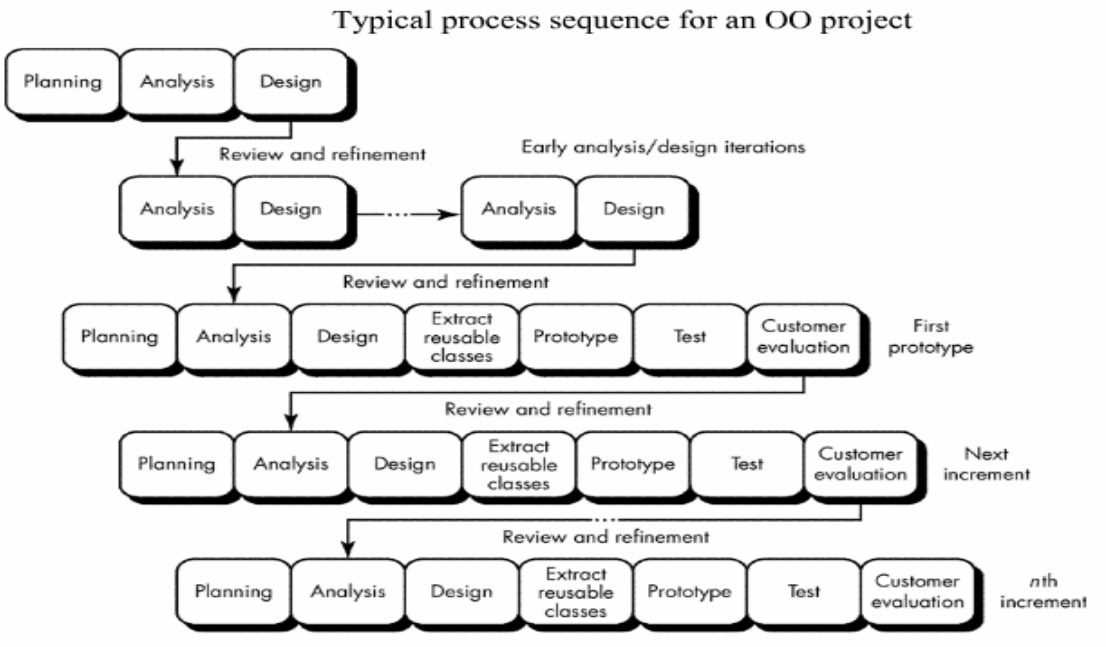
- یک چهارچوب فرایند مشترک (CPF) روشی را تعیین می کند که یک سازمان برای مهندسی نرم افزار تعیین می کند.
- (CPF) تعیین می کند که چه الگویی برای ساختن و نگهداری نرم افزار باید اعمال شود و چه وظایف و نقاط عطفی مورد نیاز خواهد بود.
- CPF درجه سختی که پروژه های مختلف دارند را تعیین می کند.
- CPF همواره قابل انطباق است و می تواند نیازهای فردی تیم پروژه را برآورده سازد. این مهمترین خصوصیت آن است.
- مهندسی نرم افزار شیءگرا مدل بازگشتی/ موازی را برای توسعه نرم افزار تقویت می کند.
- حال چگونه ما مدل بازگشتی/موازی را برای مهندسی نرم افزار شیءگرا به کار بریم؟

<u>پاسخ</u>

۱. انجام تحلیل کافی برای جداکردن کلاس ها و ارتباطات اصلی مسأله
۲. انجام اندکی طراحی برای تعیین اینکه آیا کلاس ها و ارتباطات را می توان عملاً پیاده سازی کرد یا خیر.
۳. استخراج اشیای قابل استفاده مجدد از یک کتابخانه، برای ساختن یک نمونه اولیه تقریبی
۴. اجرای چند آزمون جهت کشف خطاها در نمونه اولیه.
۵. دریافت نظرات مشتری در خصوص نمونه اولیه.
۶. اصلاح مدل تحلیل براساس آنچه که از نمونه اولیه، از انجام طراحی و از نظرات مشتری فرا گرفته اید.
۷. پالایش طراحی به منظور انجام دادن تغییرات.
۸. برنامه نویسی اشیای خاص (که در کتابخانه موجود نیست.)
۹. مونتاژ یک نمونه اولیه جدید با استفاده از اشیای کتابخانه و اشیای جدیدی که ایجاد کرده اید.
۱۰. اجرای چند آزمون برای کشف خطاها در نمونه اولیه.
۱۱. دریافت نظریات مشتری در خصوص نمونه اولیه.

- هردور تکرار نیاز به برنامه ریزی، مهندسی(تحلیل، طراحی، استخراج کلاس ها، ایجاد نمونه اولیه و آزمون) و فعالیتهای ارزیابی دارد.

- در اثنای طرح ریزی، فعالیت های مرتبط با هر یک از مؤلفه های مستقل، برنامه ریزی و زمانبندی می شود.
- طی مراحل اولیه مهندسی، تحلیل و طراحی به طور تکراری انجام می شود. هدف آن است که کلیه عناصر مهم مدل های تحلیل و طراحی OO مشخص شوند.
- به موازات پیشرفت کارهای مهندسی، نسخه های تدریجی از نرم افزار ساخته می شود.
- در اثنای این تکامل، برای هر گام یک سری بازبینی، ارزیابی مشتری و آزمون اجرا می شود و نتیجه آن بر فعالیت برنامه ریزی بعدی تأثیر خواهد گذاشت.



شکل۹ : فرایندهای متداول برای یک پروژه OO

معیارها و برآورد پروژه

- تکنیک های سنتی برآورد پروژه نرم افزاری، نیازمند برآورد کردن تعداد خطوط کد(LOC) یا ارزش تابعی(عملکرد) یعنی FP به عنوان مبنای برآورد است.
- از آنجا که یکی از اهداف پروژه های OO استفاده مجدد است ، برآوردهای مبتنی بر LOC چندان معنی پیدا نمی کند.
- برآوردهای FP را می توان به طور کارامد استفاده نمود، زیرا شمارش های دامنه اطلاعاتی مورد نیاز را به راحتی از روی بیان مسأله می توان به دست آورد.

یک مجموعه پیشنهادی از معیارهای پروژه

تعداد متون سناریو– یک متن سناریو، مراحلی است که تعامل میان کاربر و برنامه کاربردی را توصیف می کند. هر متن به شکل زیر سازماندهی می شود.

{**آغازگر، عمل، شرکت کننده**}

که در آن **آغازگر** یک شیء است که سرویسی را درخواست می کند، **عمل** نتیجه درخواست و **شرکت کننده** شیء سرویس دهنده ای است که درخواست را برآورده می کند.

- تعداد متون سناریو مستقیماً با اندازه برنامه کاربردی و با تعداد نمونه های آزمونی که باید پس از ساخته شدن سیستم روی آن امتحان شوند، ارتباط دارد.

معیارها و برآورد پروژه

۱. **تعداد کلاس های کلیدی.** کلاس های کلیدی «**مؤلفه های بسیار مستقل**» هستند که در اوایل OOA تعریف می شوند.

از آنجا که کلاس های کلیدی، در دامنه مسأله اهمیت اساسی دارند، تعداد این گونه کلاس ها شاخصی از مقدار کار لازم برای توسعه نرم افزار و نیز شاخصی از مقدار بالقوه استفاده مجدد در اثنای توسعه سیستم است.

۲. **تعدادکلاس های پشتیبان.** کلاس های پشتیبان برای پیاده سازی سیستم لازمند ولی ارتباط مستقیم با دامنه مسأله ندارند.

مثال ها عبارتند از کلاس های GUI، کلاس های دستیابی به بانک های اطلاعاتی و کلاس های محاسباتی.

۳. **تعداد میانگین کلاس های پشتیبان به ازای هر کلاس کلیدی**– اگر تعداد میانگین کلاس های پشتیبان به ازای هر کلاس برای یک مسأله مشخص باشد، برآورد بسیار ساده می شود. پیشنهاد می شود که در برنامه کاربردی با GUI تعداد کلاس های پشتیبان دو تاسه برابر کلاس های کلیدی و در برنامه های فاقد GUI یک تا دو برابر کلاس های کلیدی باشد.

۴. **تعداد زیرسیستم ها**– یک زیرسیستم، مجموعه ای از کلاس هاست که پشتیبان عملکردی است که برای کاربر نهایی سیستم قابل مشاهده است. هنگامی که زیرسیستم ها شناسایی شدند، تنظیم یک زمانبندی منطقی، آسانتر می شود.

رهیافت براورد و زمانبندی

رهیافت پیشنهادی

۱. ایجاد برآوردهایی با استفاده از تجزیه کار، تحلیلFP و هر روش دیگری که برای کاربردهای سنتی قابل اجرا باشد.
۲. با استفاده از OOA، متون سناریو را تهیه کرده یک شمارش تعیین کنید. تعداد متون سناریو ممکن است با پیشرفت پروژه تغییر کند.
۳. با استفاده از OOA تعداد کلاسهای کلیدی را تعین کنید.
۴. نوع واسط را برای برنامه کاربردی دسته بندی کرده برای کلاسهای پشتیبان ضریبی ایجاد کنید.

جزئیات مرحله

نوع واسط	ضریب
بدون GUI	2.0
واسط کاربر متنی	2.25
GUI	2.5
GUI پیچیده	3.0

۵. تعداد کل کلاس ها (کلیدی+ پشتیبان) را در تعداد میانگین واحدهای کاری ضرب کنید.Lorenz و Kidd به ازای هر کلاس ۱۵ تا ۲۰ نفر–روز کار پیشنهاد می کنند.
۶. برآورد مبتنی بر کلاسها را با ضرب تعداد میانگین واحدهای کار به ازای هر متن سناریو، چک کنید.

زمانبندی پروژه های شیءگرا با توجه به ماهیت تکراری چهارچوب فرآیند، دشوار است. Lorenz و Kidd مجموعه ای از معیارها را پیشنهاد می کنند که به زمانبندی پروژه کمک می کند:

۱. **تعدادتکرارهای اصلی**- با یادآوری مدل حلزونی یک دور تکرار اصلی متناظر با طی کردن ۳۶۰ درجه از حلزون است. Lorenz و Kidd پیشنهاد می کنند تکرارهایی با طول ۲,۵ تا ۴ ماه را به بهترین وجه می توان پیگیری و مدیریت کرد.
۲. **تعداد قراردادهای کامل شده**- قرارداد به گروهی از مسؤولیت های عمومی مرتبط اطلاق می شود که توسط زیرسیستم ها و کلاس ها فراهم می شوند. قرارداد یک نقطه عطف عالی است و حداقل یک قرارداد باید با هربار تکرار مرتبط شود.

پیگیری پیشرفت برای یک پروژه شیءگرا

با اجرای همزمان فعالیت های یک پروژه OO نقاط عطفی وجود دارد که باید توسط مدیر پروژه کنترل گردد. این نتقاط عطف و موارد مربوطه در ادامه ارایه شذه است.

۱- نقاط عطف تکنیکی: تکمیل شدن تحلیل شیءگرا

- همه کلاس ها و سلسله مراتب کلاسها تعریف و بازبینی شده اند.
- صفات و عملیات کلاس های مرتبط با یک کلاس تعریف و بازبینی شده اند.
- روابط کلاس ها تعیین و بازبینی شده اند.
- یک مدل رفتاری ایجاد و بازبینی شده است.
- کلاس های قابل استفاده مجدد مشخص شده اند.

۲- نقاط عطف تکنیکی: تکمیل شدن طراحی شیءگرا

- مجموعه ای از زیرسیستم ها تعریف و بازبینی شده است.
- کلاس ها به زیرسیستم ها تخصیص داده و بازبینی شده است.
- تخصیص وظایف انجام و بازبینی شده است.
- مسؤولیت ها و مشارکت ها مشخص شده اند.
- صفات و عملیات طراحی و بازبینی شده اند.
- مدل رد و بدل کردن پیغام ایجاد و بازبینی شده است.

۳- نقاط عطف تکنیکی: تکمیل شدن برنامه نویسی شیءگرا

- هر یک از کلاس های جدید از مدل طراحی به کد تبدیل شده است.
- کلاسهای استخراج شده (از کتابخانه) پیاده سازی شده اند.
- نمونه ساخته شده است.

۴- نقاط عطف تکنیکی: انجام آزمون شیءگرا

- درستی و کامل بودن مدل تحلیل و طراحی OO مورد بازبینی قرار گرفته است.
- یک شبکه کلاس- مسؤولیت- مشارکت توسعه یافته و مورد بازبینی قرار گرفته است.
- موارد آزمون طراحی شده اند و آزمون هایی در سطح کلاس برای هر کلاس اجرا شده اند.
- موارد آزمون طراحی شده و آزمون های خوشه ای کامل شده و کلاس ها یکپارچه شده اند.
- آزمون های در سطح سیستم کامل شده اند.

تست های فصل ۱۷: اصول و مفاهیم تحلیل شیء گرا

۱- توصیف کلی مجموعه‌ای از اشیای مشابه ... نام دارد.

الف) کلاس ب) نمونه ج) زیر کلاس د) زیر کلاس

۲- مقادیری که به صفات اشیا نسبت داده می‌شود آن شی را یکتا می‌کند:

الف- درست ب- نادرست

۳- عملیات، رویه‌های اشیا هستند که وقتی شی پیغام دریافت می‌کند فعال می‌شوند:

الف- درست ب- نادرست

۴- کدام یک از موارد زیر جزء مزیت‌های عمده معماری شی‌گرا نیست:

الف) آسانی استفاده مجدد از مولفه‌ها ب) پیشرفت کارایی و اجرایی

ج) پنهان‌سازی اطلاعات د) واسطهای ساده شده

۵- وراثت مکانیزمی است که بوسیله آن تغییرات در سطوح پایین سریعاً می‌توانند در سیستم پخش شوند:

الف) درست ب) نادرست

۶- کدام یک از گزینه‌های زیر باید بعنوان اشیای کاندید در فضای مساله دیده شوند:

الف) رویدادها ب) افراد ج) ساختارها د) هرسه

۷- صفات بوسیله آزمایش دیکشنری داده‌ها و مشخص کردن موجودیت‌های مربوط، انتخاب می‌شوند:

الف) درست ب) نادرست

۸- کدام یک از گزینه‌های زیر جزء دسته‌هایی که برای طبقه‌بندی عملیات بکار می‌روند نیست:

الف) محاسبه ب) عملیات در داده‌ها ج) event monitors د) transformer

۹- پروژه‌های شی‌گرا احتیاج به مدیریت و برنامه‌ریزی کمتری نسبت به پروژه‌های سنتی دارند.

الف) درست ب) نادرست

۱۰- کدام یک از گزینه‌های زیر از خصوصیات اصلی شی‌گرایی نیست:

الف) بسته‌بندی ب) وراثت ج) چسبندگی د) چند ریختی

۱۱- کدام گزینه از مراحل زیر مرحله مهمی از توسعه یک سیستم شی‌گراست:

الف) پایان تحلیل شی‌گرا ب) پایان طراحی شی‌گرا ج) پایان برنامه‌نویسی شی‌گرا د) هرسه

۱۲- چون هدف عمده سیستم‌های شی‌گرا، قابلیت استفاده مجدد است LOC معیار خوبی برای این پروژه‌هاست:

الف) درست ب) نادرست