فصل ۲۲ طراحی شئ گرا (OOD)

مفاهیم کلیدی (مرتب بر حروف الفبا)
اجزاء طراحی ، الگوهای طراحی ، برنامه سازی OO ، شیوه های OOD ، طراحی تفصیلی ، طراحی زیرسیستم ها ، طراحی سیستم ، طراحی شئ گرا ، عملیات ها ، لایه‌ها ، معیارهای طراحی ، هرم OOD ، UML

KEY CONCEPTS
component-level design , design components , design criteria , design patterns , layers , object design , OOD method , OOD pyramid , OO programming , operations , subsystem design , system design , UML

نگاه اجمالی

طراحی شئ گرا چیست؟ طراحی نرم‌افزار شیءگرا مستلزم تعریف یک معماری نرم‌افزاری چند لایه، تعیین مشخصات سیستم‌های فرعی که عملیات لازم را انجام داده و پشتیبانی زیربنایی را فراهم می‌آورند، توصیفی از اشیا (کلاس‌ها) که بلوک‌های سازنده سیستم هستند و توصیفی از مکانیزم‌های ارتباطی است که به اطلاعات امکان جریان یافتن بین لایه‌ها، سیستم‌های فرعی و اشیا را می‌دهند. طراحی شیءگرا به تمام این چیزها اطلاق می‌شود.

چه کسی عهده دار این امرمی‌باشد؟ OOD توسط مهندس نرم‌افزار صورت می‌گیرد.

چرا طراحی شئ گرا از اهمیت برخوردار است؟ سیستم شیءگرا بر پایه تعاریفی از کلاس ترسیم می‌شود که از مدل تحلیلی مشتق شده‌اند، بعضی از این تعاریف از نسخه اولیه تهیه شده اما بسیاری دیگر ممکن است در صورتی‌که الگوهای طراحی مناسبی وجود داشته باشند، مجدداً به‌کار گرفته شوند. OOD طرح اولیه کار طراحی را مهیا می‌سازد که مهندس نرم‌افزار را قادر می‌کند معماری OO یا شیءگرا را به‌صورتی تعریف کند که کاربرد مجدد را به حداکثر برساند بدین‌وسیله سرعت توسعه و کیفیت محصول‌نمایی را بهبود می‌بخشد.

مراحل کار کدام هاست؟ OOD به دو فعالیت عمده تقسیم می‌شود: طراحی سیستم و طراحی شیء. طراحی سیستم، معماری محصول را تعریف می‌کند یعنی یک سری لایه‌هایی را مشخص می‌سازد که کار رسیدن به کارکردهای سیستمی به‌خصوص را صورت داده و کلاس‌هایی را شناسایی می‌کند که درون

سیستم‌های فرعی قرار دارند و در هر لایه هستند. علاوه بر این طراح سیستم مشخصه سه جزء را مدنظر دارد یعنی: رابط کاربر، کارکردهای مدیریت داده‌ها و تسهیلات مدیریت وظیفه. طراحی شی‌ء روی جزئیات داخلی هر کلاس به طور مستقل تمرکز می‌کند، هم‌چنین بر روی تعریف صفات خاصه، عملیات و جزئیات پیام.

محصول کار چیست؟ مدل طراحی شی‌ءگرا در برگیرنده معماری نرم‌افزار، توصیف رابط کاربر، اجزای مدیریت داده، تسهیلات مدیریت وظیفه و توصیف دقیق هر کلاس‌ای که در سیستم استفاده می‌شود، می‌باشد.

چگونه از درست انجام گرفتن کار مطمئن شوم؟ در هر مرحله، عناصر مدل طراحی شی‌ءگرا از نظر وضوح، درستی، تکمیل بودن و سازگاری با نیازمندیهای مشتری مورد بازبینی قرار می‌گیرند.

طراحی شی‌ءگرا یا OOD مدل تحلیلی ایجاد شده را با استفاده از تحلیل شی‌ءگرا به یک مدل طراحی تبدیل می‌کند که به‌عنوان طرح اولیه ساخت نرم‌افزار استفاده می‌شود. با این حال، کار طراح نرم‌افزار می‌تواند دلهره‌آور باشد. گاما و همکارانش تصویر منطقی و دقیقی از طراحی شی‌ءگرا هنگامی ارائه می‌دهند که بیان می‌دارند: [GAM95] [1]

طراحی نرم‌افزار شی‌ءگرا سخت است و طراحی نرم‌افزار شی‌ءگرایی که قابل استفاده مجدد باشد از آن هم سخت‌تر است. باید اشیای مناسب را پیدا کنید، آنها را در کلاس‌هایی در گرانولیته درست قرار داده، رابط‌های کلاس را تعریف کرده و سلسله مراتب را انتقال دهید و ارتباطات اصلی میان آنها را به‌وجود آورید. طرح شما باید مختص مسئله مورد بررسی بوده اما آن‌قدر کلی نیز باشد که مشکلات و نیازمندیهای آتی را نیز مدنظر داشته باشد. شما می‌خواهید از طراحی مجدد خودداری کرده

یا حداقل آن را به کمترین میزان برسانید. طراحان مجرب شی‌ءگرا به شما می‌گویند که طراحی قابل استفاده مجدد و انعطاف‌پذیر اگر غیرممکن نباشد اما انجام آن در اولین بار بسیار سخت است. قبل از این‌که طراحی تمام شود، معمولاً آنها سعی می‌کنند آن را چند بار مجدداً استفاده کنند و هر بار آن را تغییر می‌دهند.

برخلاف روش‌های قراردادی طراحی نرم‌افزار، OOD منجر به طراحی‌ای می‌شود که به سطوح مختلفی در پیمانه سازی منجر می‌شود. اجزای اصلی سیستم در سیستم فرعی، پیمانه سطح سیستم، سازمان‌دهی می‌شوند. اطلاعات و عملیاتی که این اطلاعات را تغییر می‌دهند در اشیا قرار گرفته‌اند یعنی فرم پیمانه‌ای که بلوک سیستم شی‌ءگرا را می‌سازد. علاوه بر این OOD باید سازمان‌دهی اطلاعات مشخص روش‌ها و جزئیات روال هر یک از عملیات را توصیف کند. این موارد نمایانگر قطعات الگوریتمی و اطلاعاتی از سیستم شی‌ءگرا بوده و به پیمانه سازی کلی کمک می‌کنند. موجودیت منحصر به‌فرد طراحی شی‌ءگرا در توانایی آن برای ایجاد چهار مفهوم مهم طراحی نرم‌افزار نهفته است: تجرید، پنهان‌سازی

اطلاعات، استقلال عملکردی و پیمانه سازی (فصل ۱۳)، تمام روش‌های طراحی مورد استفاده برای نرم‌افزار سعی دارند این مشخصه‌های بنیادی را نشان دهند اما تنها طراحی شیءگرا مکانیزمی مهیا می‌کند که طراح را قادر می‌سازد بدون پیچیدگی یا تسلیم شدن به هر چهار مورد دست یابد.

طراحی شیءگرا، برنامه نویسی شیءگرا و آزمون شیءگرا فعالیت‌های سازنده‌ای برای سیستم شیءگرا می‌باشند. در این فصل اولین مرحله در انجام این کار را بررسی می‌کنیم.

۲۲-۱ طراحی سیستمهای شیء گرا

در فصل ۱۳ مفهوم هرم طراحی را برای نرم‌افزار قراردادی معرفی کردیم. چهار لایه طراحی یعنی داده، ساختار معماری، رابط و جزء/ سطح، هر کدام تعریف و مورد بحث قرار گرفتند. در مورد سیستم‌های شیءگرا نیز می‌توانیم هرم طراحی را تعریف کنیم اما لایه‌ها کمی متفاوتند. با توجه به شکل ۲۲-۱، چهار لایه هرم طراحی شیءگرا عبارتند از:

نقل قول

در طراحی ما به سیستم شکل می دهیم و فرم آن را پیدا می کنیم ...
ایوار ژاکوبسن، گریدی بوچ و جیمز رامباف

لایه سیستم فرعی حاوی نمایش هر سیستم فرعی است که نرم‌افزار را قادر می‌سازد به نیازمندیهای های معین شده از طرف مشتری دست یافته و زیربنای فنی را که از نیازمندی مشتری پشتیبانی می‌کند، اجرا سازد.

لایه شیء و کلاس حاوی سلسله مراتب کلاس است که سیستم را قادر می‌سازد با استفاده از تعمیم‌دهی‌ها و تخصیص‌های روز افزون با مشخصه‌های بیشتری، ایجاد شود. این لایه شامل نمایش هر شیء نیز هست.

لایه پیام حاوی جزئیات طراحی است که هر شیء را قادر به برقراری ارتباط با همکارانش می‌سازد. این لایه رابطهای داخلی و خارجی سیستم را نیز بوجود می‌آورد.

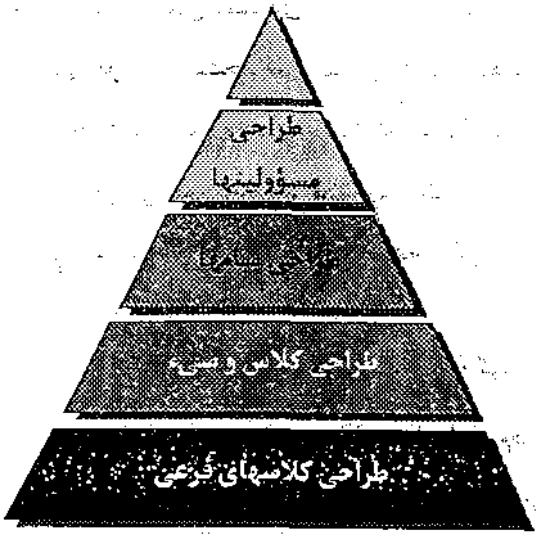
لایه مسئولیت‌ها و وظایف حاوی ساختار اطلاعاتی و طراحی الگوریتم برای همه روش‌ها و عملیات مربوط به هر شیء است.

هرم طراحی منحصراً روی طراحی محصول یا سیستم خاصی متمرکز است. باید توجه داشت که لایه دیگری نیز در طراحی وجود دارد و این لایه اساس و پایه هرم را تشکیل می‌دهد. لایه زیرین، روی طراحی اشیای میدان[2] (الگوهای طراحی[3]) متمرکز می‌شود. اشیای میدان نقش اصلی را در کارهای واسط میان انسان ـ کامپیوتر، مدیریت وظائف و داده‌ها ایفا می‌کنند. اشیای میدان را می‌توان برای Flesh out طراحی از خود برنامه کاربردی به‌کار گرفت.

1.Gamma,E.

2.domain objects

3.design patterns



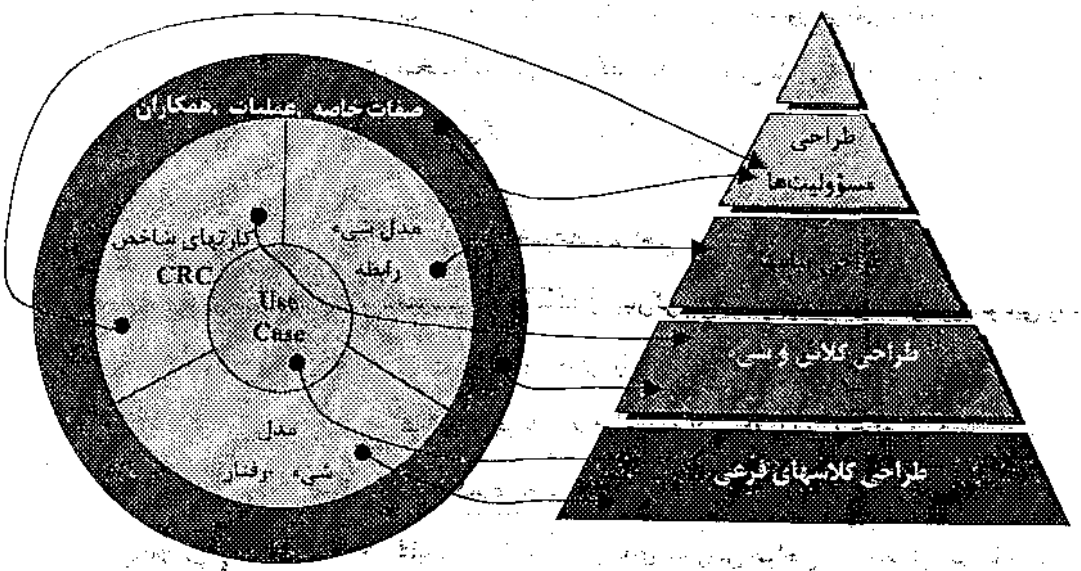
شکل ۲۲ـ۱ هرم طراحی شیء گرا

۲۲-۱-۱ رهیافت های متعارف در برابر رهیافت‌های شیٔ گرا

رهیافت‌های قراردادی یا متعارف در مورد طراحی نرم‌افزار علائمی خاص و مجموعه‌ای از روش‌های تجربی را برای نگاشت مدل تحلیلی به مدل طراحی به‌کار می‌گیرند. با یادآوری شکل ۱۳-۱ هر عنصر مدل تحلیل متعارف در یک یا چند لایه از مدل طراحی نگاشت شود. هم‌چون طراحی قراردادی نرم‌افزار، طراحی شی‌ءگرا، طراحی داده‌ها را وقتی به‌کار می‌گیرد که صفات خاصه ارائه شود، طراحی رابط را هنگامی انجام می‌دهد که وقتی‌که مدل پیام بر توسعه یابد و طراحی سطح جزء (رویه‌ای) وقتی صورت می‌گیرد که طراحی عملیات فرا می‌رسد. نکته مهم قابل توجه این است که معماری طراحی شی‌ءگرا بیشتر با همکاری میان اشیاء در ارتباط است تا با جریان کنترل بین اجزای سیستم.

با این‌که شباهت بین مدل‌های طراحی قراردادی و طراحی شی‌ءگرا وجود دارد، اما ما نام‌گذاری مجدد لایه‌هـــای هرم طراحی را انجام داده‌ایم تا به‌طور دقیق‌تر ماهیت طراحی شی‌ءگرا را منعکس سازیم. شکل ۲۲-۲ ارتباط بین مدل تحلیل شی‌ءگرا و مدل طراحی را که از آن مشتق شده، تشریح می‌کند.[۱]

۳. توجه به این نکته مهم است که اشتقاق همواره صحیح و آسان نخواهد بود. برای توضیحات بیشتر به [DAV95] مراجعه کنید.



شکل ۲۲ـ۲ ترجمان یک مدل تحلیل شیء گرا (OOA) به یک مدل طراحی شیءگرا (OOD)

طراحی سیستم فرعی با در نظر گرفتن نیازمندیهای کلی مشتری (که با موارد کاربرد ارائه می‌شوند) و رویدادها و حالاتی که به‌صورت بیرونی قابل رؤیتند (مدل رفتار شیء) ارائه می‌شود. طراحی کلاس و شیء از روی توصیف صفات خاصه، عملیات و همکاری‌های موجود در مدل CRC، کشیده می‌شود. طراحی پیام به‌وسیله مدل ارتباط شیء و طراحی مسئولیت‌ها با استفاده از صفات خاصه، عملیات و همکاری‌های توصیف شده در مدل CRC بدست می‌آیند.

فیچمن و کمرر، ده جزء مدل‌سازی طراحی را ارائه می‌دهند که ممکن است برای مقایسه روش‌های متعدد طراحی قراردادی و شیءگرا استفاده شوند:[FIC92] [1]

۱- بازنمایی سلسله مراتب پیمانه‌ها
۲- مشخصه‌های تعاریف داده‌ای
۳- مشخصه‌های منطق رویه
۴- نشانه توالی پردازش آخر به آخر
۵- بازنمایی حالات و انتقالات شیء
۶- تعریف کلاس‌ها و سلسله مراتب
۷- تخصیص عملیات به کلاس‌ها
۸- تعریف تفصیلی عملیات
۹- مشخصه‌های اتصالات پیامی
۱۰- شناسایی خدمات انحصاری

پرسش

کدام معیار ها می توانند برای مقایسه شیوه های طراحی متعارف و شیء گرا به کار روند؟

1.Fichman,R.and C.

از آن‌جا که روش‌های طراحی شی‌گرا و متداول فراوانی وجود دارند، انجام یک مقایسه عمومی بین دو روش سخت است. می‌توان گفت که ابعاد مدل‌سازی ۵ تا ۱۰ با استفاده از طراحی ساختیافته یا مشتقات آن، پشتیبانی نمی‌گردند. (فصل ۱۴)

۲۲-۱-۲ موضوعات طراحی

برتراند میر[MEY90][1] بیان‌گر پنج معیار برای قضاوت در مورد توانایی روش طراحی در دستیابی به پیمانه سازی بوده و این موارد را به طراحی شی‌گرا مرتبط می‌سازد:

- **قابلیت تجزیه‌پذیری**[2] – تسهیلاتی که با آن به طراح کمک می‌شود که یک مسئله بزرگ را به مسئله‌های کوچک‌تری تقسیم کند تا راحت‌تر بتوان آن را حل نمود.
- **قابلیت ترکیب**[3] – درجه‌ای که روش طراحی ما را مطمئن می‌سازد که اجزای مسئله (پیمانه‌ها) وقتی‌که طراحی و ساخته شدند، می‌توانند مجدداً مورد استفاده قرار گرفته تا سیستم‌های دیگری را ایجاد کنند.
- **قابلیت درک**[4] – سهولتی که با آن جزء و اجزای برنامه را می‌توان بدون رجوع به اطلاعات اضافی یا پیمانه دیگری شناخت.
- **توالی و استمرار**[5] – توانایی انجام تغییرات کوچکی در برنامه و اجازه بروز این تغییرات همراه با تغییراتی در یک یا چند پیمانه محدود.
- **حفاظت**[6] – یک خصوصیت معماری که اشاعه اثرات جانبی را در صورت وقوع خطایی در یک پیمانه فرضی، کاهش می‌دهد.

از روی این معیارها، میر [MEY90] بیان می‌دارد که پنج اصل اولیه طراحی را می‌توان در مورد معماری پیمانه ای بدست آورد: ۱) واحدهای پیمانه ای زبانی، ۲) چند رابط، ۳) رابطهای کوچک، ۴) رابطهای مشخص و صریح (پیوستگی ضعیف)، ۵) پنهان‌سازی اطلاعات. پیمانه‌ها، واحدهای پیمانه ای زبانی[7] هستند که با واحدهای نحوی در زبان مورد استفاده، مرتبطند. یعنی زبان برنامه‌نویسی مورد استفاده باید بتواند پیمانه سازی را که مستقیماً تعریف شده، پشتیبانی کند. مثلاً اگر طراح یک روال فرعی ایجاد کند هیچ‌کدام از زبان‌های برنامه‌نویسی قدیمی‌تر (مثل فورترن، C، پاسکال) نمی‌توانند آن را به‌عنوان یک

ارجاع به وب

برای پاسخ به این سوال که " چه چیز منجر به یک طراحی خوب شی‌گرا می شود؟ " توضیحاتی در آدرس زیر وجود دارد :
www.kinetica.com/ootips/ood_principles.html

پرسش

کدام اصول اولیه در طراحی معماری های پیمانه ای (ماجولار) راهنمای ما می باشند؟

1.Meyer,B.
2.Decomposability
3.composability
4.undrestandability
5.Continuity
6.protection
7.linguistic modular

واحد نحوی اجرا کنند. اما اگر بسته (package) ای که دارای ساختار و رویه‌های اطلاعاتی است و آنها را به‌عنوان یک واحد تعریف شده می‌شناسد، زبانی مثل Ada (یا زبان شی‌ءگرای دیگری) لازم است تا مستقیماً نمایان‌گر این نوع جزء در نحو زبانی باشد.

به‌منظور دستیابی به پیوستگی کم (یک مفهوم طراحی که در فصل ۱۳ معرفی شده)، تعداد رابط‌های بین پیمانه‌ها و میزان اطلاعاتی که در طول رابط حرکت می‌کند را باید به حداقل رساند. هرگاه اجزاء با هم ارتباط برقرار می‌کنند، باید آن را به شکلی مشخص و مستقیم انجام دهند. مثلاً، اگر جزء X و Y از طریق ناحیه کلی داده‌ها با هم ارتباط برقرار کنند (آنچه که ما در فصل ۱۳ پیوستگی مشترک[1] نامیدیم)، اصل رابط‌های صریح را نقض می‌کنند، زیرا ارتباط بین اجزاء برای ناظران خارجی قابل رؤیت نیست. وقتی به اصل پنهان‌سازی اطلاعات دست می‌یابیم که همه اطلاعات در مورد یک جزء از دسترسی خارجی مخفی مانده باشند، مگر این‌که اطلاعات به‌طور خاص به‌عنوان «اطلاعات عمومی»[2] تعریف شده باشند.

معیارها و اصول طراحی ارائه شده در این بخش را می‌توان در هر روش طراحی به‌کار گرفت (مثلاً می‌توانیم آن را در طراحی ساختیافته به‌کار گیریم). همان‌گونه که خواهیم دید، روش طراحی شی‌ءگرا کارآمدتر از دیگر روش‌ها به هر یک از معیارها دست یافته و منجر به معماری‌های پیمانه ای می‌شوند که به ما امکان می‌دهد به‌صورت مؤثرتر هر یک از معیارهای پیمانه سازی را بدست بیاوریم.

۲۲-۱-۳ چشم انداز طراحی شی‌ءگرا

همان‌گونه که در فصل ۲۱ توجه کردیم، طیف وسیعی از روش‌های طراحی و تحلیل شی‌ءگرا در طول دهه ۸۰ و اوایل دهه ۹۰ پیشنهاد و مورد استفاده قرار گرفتند. این روش‌ها اساس شیوه طراحی شی‌ءگرای مدرن، روش اکتشافی طراحی و مدل‌های مدرن را پایه‌گذاری کردند. جمع‌بندی خلاصه‌ای از مهم‌ترین روش‌های اولیه طراحی شی‌ءگرا در ادامه آمده است:

روش بوچ. همان‌گونه که در فصل ۲۱ اشاره کردیم، روش بوچ[BOO94][3] در برگیرنده فرآیند کوچک و بزرگ توسعه است (Macro & Micro). بافت طراحی، توسعه و رشد ماکرو (بزرگ) در برگیرنده فعالیت برنامه‌ریزی معماری است که اشیای مشابه را در قسمت‌های معماری جداگانه قرار می‌دهد، اشیای مختلف را به‌وسیله سطح انتزاع و تجرید در لایه‌ها قرار می‌دهد، سناریوهای مربوطه را شناسایی می‌کند، نمونه اولیه طراحی را ایجاد کرده و با به‌کارگیری سناریو اولیه ، طراحی نمنه اولیه را ارزیابی می‌کند. توسعه میکرو مجموعه قوانینی را تعریف می‌کند که بر استفاده عملیات و صفات خاصه و میدان سیاست‌های ویژه در مورد مدیریت حافظه، درست کردن خطاها و دیگر کارکردهای زیربنایی نظارت کرده،

نقل قول

دلیلی وجود ندارد که انتقال از نیازمندیها به طراحی در مهندسی نرم افزار ، ساده تر از دیگر دیسیپلین های مهندسی باشد. طراحی مشکل است.

آلن دیویس

1.common coupling
2.public information
3.Booch,G.

سناریوهایی ارائه می‌دهد که معنی قوانین و سیاست‌گذاری‌ها را توصیف نموده، نمونه اولیه کارکرد برای هر یک از این سیاستها ایجاد می‌کند، ابزار لازم را مهیا نموده و نمونه اولیه را پالایش می‌کند و نهایتاً هر سیاست وضع شده را بازبینی می‌نماید، به‌طوری‌که نگرش معماری آن را منتشر می‌سازد.

روش رامباف. تکنیک مدل‌سازی شیء[1] (OMT) شامل [RUM91][2] فعالیت طراحی است که اجرای طراحی را در دو سطح جداگانه تجزیه‌ی ترویج می‌دهد. طراحی سیستم[3] روی طرح‌بندی اجزایی متمرکز می‌شود که برای ساخت سیستم یا محصول کامل ضروری‌اند. مدل تحلیلی به‌صورت سیستم‌های فرعی تقسیم‌بندی می‌شود که سپس به پردازنده‌ها و وظائف تخصیص می‌یابد. یک راهبرد برای پیاده‌سازی مدیریت داده تعریف شده و منابع کلی و مکانیزم‌های کنترلی لازم برای دسترسی به آنها نیز تعریف شده‌اند.

طراحی شیء[4] بر طرح‌بندی دقیق هر شیء تأکید دارد. عملیات از روی مدل تحلیل انتخاب شده و الگوریتم‌ها برای هر عملیات تعریف می‌شوند. ساختارهای اطلاعاتی که برای روش‌ها و الگوریتم‌ها مناسبند، ارائه می‌گردند. کلاس‌ها و صفات خاصه آنها به شیوه‌ای طراحی می‌شوند که دسترسی به اطلاعات را به‌صورت بهینه درآورده و کارآیی محاسباتی را بهبود می‌بخشد. مدل پیغام‌بر برای اجرای روابط شیء ایجاد می‌شود.

روش یاکوبسن. فعالیت طراحی در مهندسی نرم‌افزار شیءگرا نسخه ساده شده‌ای از روش اختصاصی شیءگرا[5] است که آن هم توسط یاکوبسن [JAK92][6] ارائه شده است. مدل طراحی بر قابلیت پی‌گیری در مدل تحلیل مهندسی نرم‌افزاری شیءگرا تأکید دارد. ابتدا، مدل تحلیلی ایده‌آل گرفته می‌شود تا با محیط جهان واقعی سازگاری یابد. سپس اشیای طراحی اولیه به نام بلوک‌ها[7] ایجاد شده و بر اساس بلوک‌های رابط، بلوک موجودیت و بلوک‌های کنترلی طبقه‌بندی می‌شوند. ارتباط بین بلوک‌ها در طول اجرا تعریف شده و بلوک‌ها به‌صورت سیستم‌های فرعی سازمان‌دهی می‌شوند.

روش کُد و یوردون. روش کُد و یوردون برای طراحی شیءگرا [COA91] با بررسی این امر که چگونه "طراحان مؤثر شیءگرا" کار طراحی خود را انجام می‌دهند، توسعه یافت. رهیافت طراحی نه‌تنها برنامه کاربردی بلکه زیربنای برنامه را نیز مخاطب ساخته و روی بازنمایی چهار جزء اصلی سیستم متمرکز می‌شود: جزء میدان مسئله، جزء رابطه متقابل انسانی، جزء مدیریت وظیفه و جزء مدیریت اطلاعات.

1.Object Modeling Techniques

2.Rumbaugh,J.

3.system design

4.object design

5.objectory method

6.Jacobson,I.

7.Block

یک بلاک، تجرید طراحی است که اجازه می دهدکه یک شئ ترکیبی و متراکم بازنمایی شود.

روش ورفز – براک. ورفز - براک توالی وظائف فنی را تعریف می‌نماید که در آن تحلیل بدون وقفه به طراحی منجر می‌شود. پروتکل‌های[1] هر کلاس با پالایش قراردادهای میان اشیا منعقد می‌شوند. هر عملیات (مسئولیت)و پورتکل (طراحی رابط) در سطح دقیقی طراحی می‌شود که کار پیاده‌سازی را هدایت می‌کند. مشخصات هر کلاس (تعریف مسئولیتهای خصوصی و جزئیات عملیات) و هر سیستم فرعی (شناسایی همه کلاس‌های بسته‌بندی شده و ارتباط متقابل بین سیستم‌های فرعی) ارائه می‌شود.

هرچند شیوه ورفز بروک به قدرتمندی UML نباشد، ظرافتی دارد، که آنرا جایگزین جالبی برای رهیافت OOD می سازد.

گرچه مراحل فرآیند و واژه‌شناسی برای هر یک از روش‌های طراحی شیءگرا متفاوتند، اما فرآیندهای کلی طراحی شیءگرا به‌طور منطقی یکنواختند. برای انجام طراحی شیءگرا، مهندس نرم‌افزار باید مراحل کلی زیر را انجام دهد:

۱- هر سیستم فرعی را توصیف نموده و آن را به پردازنده‌ها و وظائف اختصاص دهید.

۲- برای پیاده‌سازی مدیریت داده‌ها، پشتیبانی رابط و مدیریت وظیفه، یک راهبرد طراحی انتخاب کنید.

۳- مکانیزم کنترلی مناسب را برای سیستم بیابید و طراحی کنید.

۴- با ایجاد یک بازنمایی رویه‌ای برای هر عملیات و ساختار داده‌ای در مورد صفات خاصه کلاسها، طراحی شیء را انجام دهید.

۵- با استفاده از همکاری بین اشیا و ارتباط اشیاء، طراحی پیام را انجام دهید.

۶- مدل پیامبر را ایجاد کنید.

۷- مدل طراحی را پالایش نموده و تا وقتی‌که لازم است تکرار کنید.

یک مجموعه از گامهای عمومی که طی فرآیند طراحی شیءگرا (OOD) بکار می روند، مستقل از شیوه طراحی انتخاب شده می باشند.

نکته مهم قابل توجه این است که مراحل طراحی مورد بحث در این بخش تکراری هستند. یعنی، ممکن است به‌صورت افزایش همراه با فعالیت‌های اضافی تحلیل شیءگرا انجام گیرند، تا وقتی‌که طراحی کامل ایجاد شود.

یک آموزش کامل به همراه لیستی از منابع UML شامل ابزارها، مقالات و مثالها در آدرس زیر وجود دارند: www.mininet/cetus/oo_uml.htm

۲۲-۱-۴ یک رهیافت یکنواخت برای طراحی شیءگرا

در فصل ۲۱ دیدیم که گرادی بوچ، جیمز رامباف و ایوار یاکوبسن بهترین مشخصه‌های روش‌های طراحی و تحلیل شیءگرای خود را به‌صورت یک روش متحدالشکل[2] درآورده‌اند. نتیجه آن که، زبان مدل‌سازی متحدالشکل یا UML است به‌طور گسترده‌ای در صنعت استفاده شده است.[3]

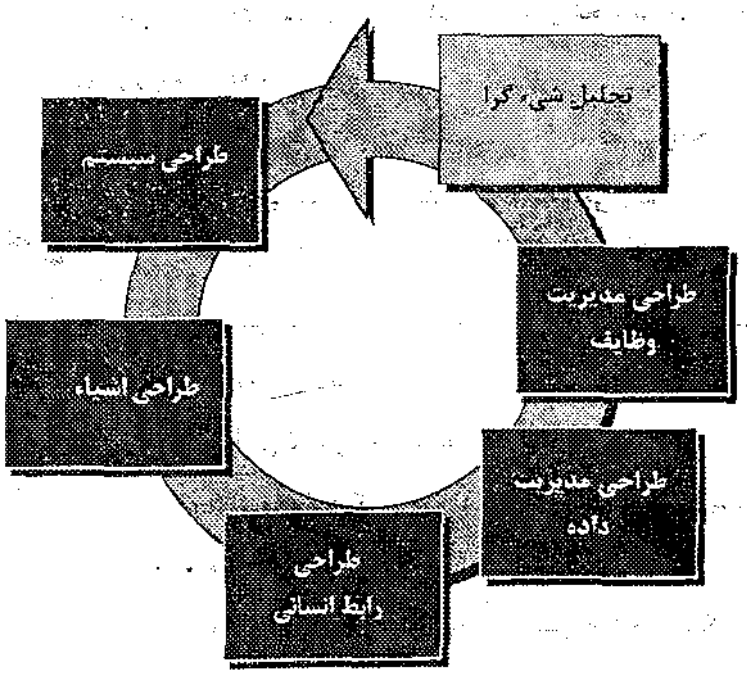
1.Protocols

یک پروتکل، توصیفی رسمی است از پیامهایی که یک کلاس در پاسخ ارسال می دارد.

2.Bennett,S.

۳. بوچ، رامباف، و یاکوبسن (یاکوبسن) مجموعه سه کتاب توصیفی درخصوص UML تالیف نموده اند. خوانندگان مشتاق می توانند به [BOO99] ، [RUM99] و [JAC99] مراجعه نمایند.

در طول مدل‌سازی تحلیل (فصل ۲۱) دیدگاه مدل کاربر و ساختاری ارائه می‌شوند. این دو دیدگاه، بینشی در مورد سناریوهای کاربرد برای سیستم ارائه نموده (راهنمایی در مورد مدل‌سازی رفتاری مهیا می‌نماید) و اساس مدل پیاده‌سازی و مدل محیطی را با شناسایی و توصیف عناصر استاتیک ساختاری سیستم، ایجاد می‌کند.



شکل ۲۲-۳ روند فرآیند برای طراحی شی ء گرا (OOD)

UML[1] به‌صورت دو فعالیت عمده طراحی سازمان می‌یابد: طراحی سیستم و طراحی شیء. هدف اولیه طراحی سیستم UML عبارتست از بازنمایی معماری نرم‌افزار. بنت و همکارانش این موضوع را به شکل زیر مورد بحث قرار داده‌اند: [BEN99][2]

از نظر توسعه و پیشرفت شی‌ءگرا، معماری مفهومی مربوط به ساختار مدل کلاس ایستا و ارتباطات بین اجزای مدل است. معماری پیمانه شیوه‌ای را توصیف می‌کند که سیستم را به دو زیر مجموعه یا چند پیمانه تقسیم نموده و چگونگی برقرار ارتباط آنها را به‌وسیله ورود و خروج اطلاعات تشریح می‌کند. معماری برنامه تعریف می‌کند که چگونه کد برنامه به‌صورت فایل‌ها و دایرکتوری‌ها سازمان‌دهی شده و به‌صورت کتابخانه کلاس‌بندی می‌شود. معماری اجرایی روی جنبه‌های دینامیک و پویای سیستم و ارتباط بین اجزا با ارجای فعالیت‌ها و عملیات متمرکز می‌شود.

1.UML system design

2.Bennett,S.

تعریف سیستم فرعی که توسط بنت ارائه شده، در طول طراحی سیستم به‌وسیله UML، مسئله اصلی خواهد بود. طراحی شیء به‌صورت UML[1] روی توصیف اشیا و تعامل متقابل آنها با یکدیگر متمرکز می‌شود. مشخصه‌های دقیق ساختارهای داده صفات خاصه و طراحی رویه‌ای همه عملیات در طول طراحی شیء، ایجاد می‌شوند. قابلیت رؤیت[2] روش همه کلاس‌ها تعریف شده و رابط بین اشیا بسط یافته‌اند تا جزئیات یک مدل پیام‌گیر کامل را تعریف کنند.

طراحی سیستم و شیء در UML توسعه یافته تا طراحی رابط کاربر، مدیریت داده‌ها یا سیستم ساخته شده و مدیریت کار را برای سیستم‌های فرعی که مشخص شده‌اند، بررسی کند. طراحی رابط کاربر در UML دارای همان اصول و مفاهیم مورد بحث در فصل ۱۵ است. دیدگاه مدل کاربر باعث جریان یافتن فرآیند طراحی رابط کاربر شده که سناریویی ارائه می‌نماید که به‌صورت مکرر بسط می‌یابد تا مجموعه‌ای از کلاس‌های رابط را تشکیل دهند.[3] طراحی مدیریت داده‌ها مجموعه‌ای از کلاس‌ها و همکاری‌ها ایجاد می‌کند که به سیستم امکان می‌دهد داده‌های متداول را (مثل فایل‌ها و بانک‌های اطلاعاتی) سازمان‌دهی کند. طراحی مدیریت وظیفه زیربنایی را به‌وجود می‌آورد که سیستم فرعی را در کارها سازمان‌دهی نموده و سپس وظائف را به‌طور همزمان مدیریت می‌کند. جریان فرآیند از تحلیل تا طراحی در شکل ۲۲-۳ آمده است.[4]

طراحی سیستم متمرکز بر معماری نرم افزار و تعریف زیرسیستم هاست. طراحی شیء اشیاء را در سطحی از جزئیات توصیف می کند که با یک زبان برنامه سازی قابل پیاده سازی باشند.

در سراسر فرآیند طراحی UML، دیدگاه‌های مدل کاربر و مدل ساختاری در بازنمایی طراحی ارائه شده فوق، بسط می‌یابند. این فعالیت توسعه در بخش‌های بعدی مورد بحث قرار گرفته است.

۲-۲۲ فرآیند طراحی سیستم

گامهای فرآیند طراحی سیستم کدام هاست؟

طراحی سیستم جزئیات معماری لازم برای ایجاد یک سیستم یا محصول را نشان می‌دهد. این فرآیند دربرگیرنده فعالیت‌های زیر است:

- مدل تحلیلی را به سیستم‌های فرعی تقسیم می‌کند.
- همروندی را که به‌وسیله مسئله به‌وجود آمده‌اند، مشخص سازد.
- سیستم‌های فرعی را به پردازنده‌ها و وظائف اختصاص دهد.
- یک طراحی برای رابط کاربر ایجاد کند.

---

1. Unified Modeling Language

۲. قابلیت رویت، معلوم می دارد که یک صفت خاصه عمومی است (دسترسی آزاد تمام موارد و مصادیق کلاس) یا خصوصی (قابل استفاده فقط برای یک کلاس مشخص) یا محافظت شده (صفت خاصه می تواند برای یک کلاس و زیرکلاسهای آن مورد استفاده واقع شود).

۳. به خاطر آورید که تحلیل ش، گرا یک فعالیت تکراریذیر است. محتمل است که مدل تحلیلی، حاصل بازبینی کار طراحی باشد.

۴. در یک معماری بسته، پیامها از یک لایه تنها به لایه مجاور پایینی ارسال خواهند شد. در یک معماری باز، پیامها به هر لایه پایین تر می توانند ارسال شوند.

- یک راهبرد مقدماتی را برای پیاده‌سازی مدیریت داده‌ها انتخاب کند.
- منابع جهانی و مکانیزم‌های کنترلی لازم برای دسترسی به آنها را مشخص سازد.
- مکانیزم کنترلی مناسب را برای سیستم طراحی نماید از جمله مدیریت وظیفه.
- در نظر بگیرد که چگونه شرایط مرزی باید اعمال شوند.
- توازن‌ها را مدنظر داشته و بازبینی کنید.

در بخش‌های بعدی، کارهای طراحی که مربوط به هر یک از این مراحل هستند به‌طور دقیق‌تر بررسی شده‌اند.

۱-۲-۲۲ تجزیه مدل تحلیل

یکی از اصول بنیادین تحلیل (فصل ۱۱) تقسیم‌بندی است. در طراحی سیستم شی‌گرا مدل تحلیلی را برای این تقسیم می‌کنیم تا مجموعه‌های منسجم کلاس‌ها، رابطه‌ها و رفتار تعریف شوند. این عناصر طراحی به‌عنوان سیستم فرعی کلاس‌بندی می‌شوند.

به‌طور کلی تمام عناصر سیستم فرعی خواصی به‌طور مشترک را دارا می‌باشند. آنها همگی در رسیدن به کارکردی یکسان مشارکت دارند. در سخت‌افزار یکسانی از محصول قرار دارند یا کلاس یکسانی از منابع را مدیریت می‌کنند. سیستم‌های فرعی به‌وسیله وظایفشان مشخص می‌شوند یعنی یک سیستم فرعی را می‌توان توسط خدماتی که ارائه می‌دهد، مشخص نمود. خدمات اگر در بستر طراحی سیستم شی‌گرا استفاده شوند مجموعه عملیاتی است که کارکرد ویژه‌ای را انجام می‌دهد. (مثل مدیریت فایل‌های واژه‌پرداز، تولید یک رندر سه بعدی[2]، تغییر سیگنال تصویری آنالوگ به یک تصویر دیجیتالی فشرده).

همان‌گونه که سیستم‌های فرعی تعریف می‌شوند (و طراحی می‌گردند)، باید با معیارهای طراحی زیر مطابقت داشته باشند:

- سیستم فرعی باید دارای یک رابط تعریف شده درست باشد که از طریق آن همه ارتباطات با بقیه قسمت‌های سیستم صورت گیرد.
- به غیراز کلاس‌های ارتباطی معدودی، بقیه کلاس‌های درون سیستم فرعی باید تنها با کلاس‌های درون سیستم فرعی همکاری کنند.
- تعداد سیستم‌های فرعی باید کم باشد.
- یک سیستم فرعی را باید به‌صورت داخلی تقسیم کرد تا به کاهش پیچیدگی کمک کند.

توصیه

مفهوم پیوستگی و چسبندگی می تواند در سطح زیر سیستم ها به کار رود. بکوشید تا استقلال عملیاتی خوبی را در سطح زیرسیستم ها بوجود آورید.

پرسش

کدام معیار ها ما را در طراحی زیر سیستم ها راهنمایی می کنند؟

1.Rumbaugh,J.

2.Three – dimention rendering

وقتی دو سیستم فرعی با هم ارتباط برقرار می‌کنند، می‌توانند یک ارتباط خادم/ مخدوم یا نظیر به نظیر[RUM91] برقرار کنند. در ارتباط خادم/ مخدوم، هر یک از سیستم‌های فرعی یکی از نقش‌های اجرا شده توسط مخدوم یا خادم را به‌عهده می‌گیرد. این سرویس از جانب خادم به مخدوم یا مشتری، تنها در یک جهت جریان می‌یابد. در ارتباط نظیر به نظیر ممکن است این سرویس در هر دو جهت باشد.

وقتی سیستمی به سیستم‌های کوچک‌تری تقسیم شد، کار طراحی دیگری به نام لایه‌گذاری رخ می‌دهد. هر لایه[BUS96][1] سیستم شیءگرا شامل یک یا چند سیستم فرعی بوده و نمایان‌گر سطح تجریدی متفاوتی نسبت به عملکرد و کارآیی لازم برای دست‌یابی به عملیات سیستم است. در اکثر موارد، سطوح تجریدی توسط درجه‌ای تعیین می‌شوند که تا آن مقدار پردازش مربوط به سیستم فرعی برای کاربر نهایی قابل رؤیت است.

مثلاً، یک معماری چهار لایه ممکن است شامل موارد زیر باشد: ۱) لایه بازنمایی (سیستم‌های فرعی مربوط به رابط کاربر) ۲) لایه برنامه کاربردی (سیستم‌های فرعی که پردازش مربوط به برنامه کاربردی را انجام می‌دهند). ۳) لایه قالب داده‌ها (سیستم‌های فرعی که داده‌ها را برای پردازش آماده می کنند) و ۴) لایه پایگاه داده‌ها (سیستم‌های فرعی مربوط به مدیریت داده‌ها). هر لایه به‌طور عمیق‌تر وارد سیستم می‌شود که نمایان‌گر افزایش پردازش خاص‌تری برای محیط است.

پرسش

چگونه می توانم یک پالایش لایه ای را ایجاد نمایم؟

بوشمن [BUS96] و همکارانش رهیافت طراحی زیر را برای لایه‌گذاری ارائه می‌دهند:

۱- معیار لایه‌بندی را برقرار سازید. یعنی تصمیم‌گیری در مورد این‌که چگونه سیستم‌های فرعی در معماری و ساختار لایه‌بندی شده، قرار گیرند.

۲- تعداد لایه‌ها را تعیین کنید. بیش از حد بودن آنها پیچیدگی دلیل ایجاد می‌کند و کم بودن بیش از حدشان نیز به استقلال کارکردی لطمه می‌زند.

۳- نام لایه‌ها را تعیین کنید و سیستم‌های فرعی را (با کلاس‌های موجود در آن) به یک لایه اختصاص دهید. مطمئن شوید که ارتباط بین سیستم‌های فرعی (کلاس‌ها) روی یک لایه و دیگر سیستم‌های فرعی روی لایه دیگر، از یک فلسفه طراحی برای معماری، برخوردارند.

۴- رابط‌هایی را برای هر لایه طراحی کنید.

۵- سیستم‌های فرعی را پالایش کنید تا ساختار کلاس را برای هر لایه به‌وجود آورید.

۶- مدل پیام‌بر را برای ارتباط بین لایه‌ها تعریف کنید.

۷- طراحی لایه را بازبینی کنید تا مطمئن شوید که ارتباط بین لایه‌ها به حداقل رسیده است. (یک پروتکل خادم / مخدوم می‌تواند در دستیابی به این امر یاریگر باشد)

۸- کار پالایش را برای طراحی لایه‌بندی شده تکرار کنید.

1.Buschmann,F.

۲۲-۲-۲ همروندی و تخصیص زیرسیستم

جنبه پویایی مدل رفتار شیء نشانه همروندی میان کلاس‌هاست (یا سیستم‌های فرعی). اگر کلاس‌ها (یا سیستم‌های فرعی) هم‌زمان فعال نباشند، لازم به پردازش هم‌زمان نیست. یعنی این‌که می‌توان کلاس‌ها را روی همان سخت‌افزار رایج پردازنده اجرا کرد. به‌عبارت دیگر اگر کلاس‌ها باید روی رویدادها به‌طور غیر هم‌زمان کار کنند، در همان حال به‌طور همروند رؤیت می‌شوند. وقتی سیستم‌های فرعی همروند می‌باشند، دو گزینه برای تخصیص وجود دارد: ۱) هر سیستم فرعی را به یک پردازنده مستقل تخصیص داده یا ۲) سیستم‌های فرعی را به همان پردازنده تخصیص داده و از طریق ویژگیهای سیستم عامل پشتیبانی هم‌زمانی مهیا کنیم.

توصیه

در بسیاری موارد یک پیاده سازی چند پردازنده ای، پیچیدگی و مخاطرات فنی را افزایش می دهد. هرکجا ممکن است، ساده ترین معماری پردازنده ای را در دستور کار خود قرار دهید.

وظائف همروند با بررسی نمودار وضعیت در مورد هر شیء تعریف شده‌اند[RUM91] اگر جریان رویدادها و انتقال‌ها نشان‌گر این باشد که تنها یک شیء در هر زمان فعال است، یک ریسمان[1] کنترلی ایجاد شده است. ریسمان کنترلی تا وقتی ادامه می‌یابد که شیء پیامی به شیء دیگر ارسال می‌کند که در عین‌حال شیء اول منتظر پاسخ است. اگر شیء اول بعد از ارسال پیام به پردازش ادامه دهد، ریسمان کنترلی قطع می‌شود.

وظائف در یک سیستم شی‌گرا با جداسازی ریسمان‌های کنترلی طراحی می‌شوند. مثلاً، وقتی سیستم ایمنی خانهٔ امن در حال کنترل سنسورهاست، می‌تواند با ایستگاه مرکزی برای شناسایی ارتباط، تماس بگیرد. از آن‌جا که اشیای درگیر در هر دو حالت در یک زمان فعالند هر یک نمایان‌گر ریسمان کنترلی جداگانه‌ای بوده و هر کدام به عنوان یک وظیفه جداگانه تعریف می‌شوند. اگرفعالیتهای مشاهده و برقراری تماس به‌طور پیاپی رخ دهند، می‌توان یک وظیفه منفرد را پیاده سازی نمود. به‌منظور تعیین این‌که کدام گزینه تخصیص پردازنده‌ای که در بالا به آن اشاره شده، مناسب است طراح باید نیازمندیهای عملکرد، هزینه‌ها و سربارهای[2] اعمال شده توسط ارتباطات درون پردازنده‌ای را مدنظر قرار دهد.

۲۲-۲-۳ جزء مدیریت وظیفه

کَد و یوردون[COA91][3] راهبرد زیر را برای طراحی اشیایی ارائه می‌دهند که وظائف همروند را مدیریت می‌کنند:

- مشخصه‌های وظائف تعیین می‌شوند.
- وظیفه هماهنگ‌کننده و اشیای مربوطه تعریف می‌شوند.
- وظیفه هماهنگ‌کننده و دیگران یکپارچه می‌شود.

نقل قول

انضباط و هشیاری تمرکز یافته ، هنرخلاقیت را یاری خواهند داد. جان پاپی

1 .Thread

2.Overhead

3.Coad,P.

مشخصه‌های یک وظیفه توسط شناخت چگونگی شروع وظیفه تعیین می‌شوند. وظایف مبتنی بر رویدادها و وظایف ناشی از زمان، بیشتر مورد توجه خواهد بود. هر دو توسط یک وقفه فعال می‌شوند، اما اولی وقفه را از منبع خارجی دریافت می‌کند (مثل یک پردازنده یا سنسور دیگر) در حالی‌که دومی تحت ساعت سیستم است.

علاوه بر شیوه‌ای که به واسطهٔ آن وظیفه آغاز می‌شود، اولویت و اهمیت وظیفه نیز باید تعیین شود. وظایف دارای اولویت زیاد، دسترسی فوری به منابع سیستم دارند. وظایف دارای اهمیت زیاد باید به عملیات ادامه دهند حتی اگر دسترسی به منبع کاهش یافته یا سیستم در حالتی تنزل کرده به کار ادامه می‌دهد.

وقتی مشخصه‌های وظیفه تعیین شد، صفات خاصه اشیا و عملیات لازم برای هماهنگی و برقراری ارتباط با دیگر وظایف تعریف می‌شوند. طرح اولیه وظیفه شکل زیر را به‌خود می‌گیرد:

نام وظیفه ـ نام شیء

توصیف ـ گزارش توصیفی در مورد منظور ایجاد شیء

اولویت ـ اولویت وظیفه (کم، متوسط، زیاد)

خدمات ـ فهرستی از عملیاتی که که در حوزهٔ مسئولیت شیء هستند.

هماهنگی از طریق رفتاری که توسط آن شیء برانگیخته می‌شود.

برقراری ارتباط از طریق ـ مقادیر داده‌های ورودی و خروجی مربوط به وظیفه

Task name--the name of the object
Description--a narrative describing the purpose of the object
Priority-task priority (e.g., low, medium, high)
Services--a list of operations that are responsibilities of the object
Coordinates by--the manner in which object behavior is invoked
Communicates via-input and output data values relevant to the task

سپس می‌توان این توصیف طرح را به مدل استاندارد طراحی برای اشیاء تغییر داد.

۲۲-۲-۴ جزء رابط کاربر

با این‌که جزء رابط کاربر در متن میدان مسئله پیاده می‌شود، خود رابط نمایان‌گر سیستم فرعی مهمی برای مدرن‌ترین برنامه‌های کاربردی است. مدل تحلیل شیءگرا (فصل ۲۱) دربرگیرنده طرح‌های کاربرد (به‌اصطلاح موارد استفاده)[1] و توصیفی از نقش‌هایی است که کاربران (به‌اصطلاح بازیگران) در هنگام برقراری ارتباط متقابل با سیستم ایفا می‌کنند. این موارد به‌عنوان اطلاعات ورودی در فرآیند طراحی رابط کاربر عمل می‌کنند.

ارجاع به فصلها

بسیاری از کلاسها نیازمند ساخت رابطی مدرن می باشند که موجود و در دسترس طراحان باشد. طراحی یک رابط مبتنی بر رهیافتی است که درفصل ۱۵ تعریف شده است.

1 use-cases

وقتی بازیگر و سناریوی مورد استفاده‌اش تعیین شدند، سلسله مراتب فرمان شناسایی می‌شوند. سلسله مراتب فرمان طبقات عمده منوی سیستم را (یعنی Menu. Bar یا Tool Palette) همراه با تمام عملیات جانبی که در متن این طبقه عمده منوی سیستم موجودند، تعریف می‌کند. سلسله مراتب فرمان به‌صورت مکرر پالایش می‌شوند تا وقتی‌که هر مورد کاربرد با جستجوی سلسله مراتب عملیاتی بتواند پیاده شود.

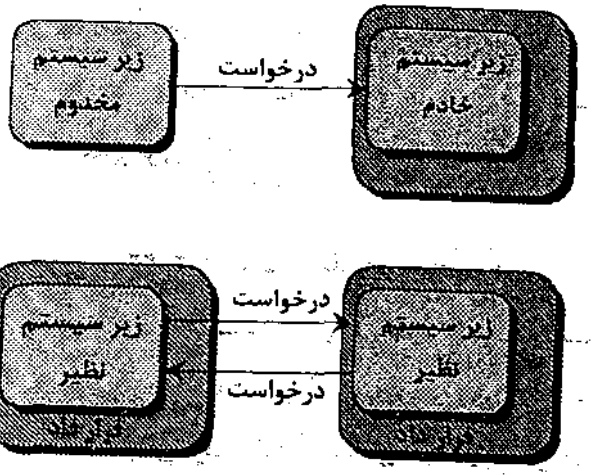
از آن‌جا که طیف وسیعی از محیط‌های توسعه رابط کاربر هم‌اکنون وجود دارند، طراحی عناصر GUI لازم نیست. کلاس‌های قابل استفاده مجدد (با صفات خاصه و عملیات مناسب) هم‌اکنون در مورد پنجره‌ها، ایکون‌ها، عملیات موس و طیف گسترده‌ای از کارکردهای محاوره‌ای دیگر، وجود دارد. پیاده‌سازی‌کننده تنها به راه‌اندازی اشیایی نیاز دارد که دارای مشخصه مناسب برای حوزه مسئله هستند.

۲۲-۲-۵ جزء مدیریت داده

مدیریت داده دربرگیرنده دو حوزه کاملاً متمایز است: ۱) مدیریت داده‌هایی که برای خود برنامه مهم هستند و ۲) ایجاد زیربنایی برای ذخیره و بازیابی اشیا. به‌طور کلی، مدیریت داده‌ها به‌صورت لایه‌بندی شده طراحی گردیده است. این ایده عبارتست از جداسازی نیازمندیهای سطح پایین برای تغییر ساختارهای داده‌ای از نیازمندیهای سطح بالا برای ارائه صفات خاصه سیستم.

در بافت سیستم، اغلب از یک سیستم مدیریت به‌عنوان یک مخزن مشترک داده‌ها برای همه سیستم‌های فرعی، استفاده می‌شود. اشیای لازم برای عمل با پایگاه داده‌ها اعضای کلاس‌های قابل استفاده مجددی هستند که با استفاده از تحلیل دامنه (فصل ۲۱) شناسایی شده یا مستقیماً توسط فروشنده بانک اطلاعاتی عرضه می‌شوند. بحث دقیق طراحی بانک اطلاعاتی برای سیستم‌های شیءگرا فراتر از دامنه این کتاب است.[۲]

۲. خوانندگان مشتاق می توانند به [BRO91] یا [TAY92] یا [RAO94] رجوع نمایند.



شکل ۲۲ـ۴ مدلی از همکاری (تشریک تساعی) بین زیر سیستم ها
[خادم /مخدوم و نظیر به نظیر ـ مترجم]

طراحی جزء مدیریت داده شامل طراحی صفات خاصه و عملیات لازم برای مدیریت اشیاء می‌باشد. نگرش‌های مربوطه در دامنه مسئله در مورد هر شیء ضمیمه شده‌اند و اطلاعاتی مهیا می‌کنند که سؤالاتی از قبیل موارد زیر را پاسخ می‌دهند: چگونه من خودم را ذخیره کنم؟ کُد و یوردون[COA91] ایجاد یک کلاس سرویس‌دهنده شیء را با خدمات برای الف) گفتن این مطلب به هر شیء که خود را ضبط کند و ب) بازیابی اشیای ذخیره شده برای استفاده توسط دیگر اجزای طراحی، پیشنها می‌کنند.

به‌عنوان مثال در مورد مدیریت داده برای شیء سنسور که به‌عنوان بخشی از سیستم ایمنی خانهٔ امن مورد بحث قرار گرفت، طراحی می‌توانست یک فایل سطحی به نام سنسور را مشخص کند. هر رکورد مربوط به یک لحظه نام‌گذاری شدهٔ سنسور بوده و شامل مقدار هر صفت خاصه سنسور برای آن لحظه نام‌گذاری شده است. عملیات درون کلاس سرویس‌دهنده ـ شیء یک شیء خاص را قادر به ذخیره سازی و بازیابی در زمانی می‌کند که سیستم به آن نیاز دارد. در مورد اشیای پیچیده‌تر، ممکن است لازم باشد که یک بانک اطلاعاتی رابطه‌ای یا بانک اطلاعاتی شیءگرا را برای رسیدن به کارکرد یکسان، مشخص می‌نمود.

۲۲-۲-۶ جزء مدیریت منبع

در مورد یک سیستم یا محصول شیءگرا یکسری منابع مختلف وجود دارد و در بسیاری از مواقع سیستم‌های فرعی در یک زمان به‌خاطر این منابع با یکدیگر رقابت می‌کنند. منابع عمومی سیستم می‌توانند موجودیت‌های برونی (مثل دیسک درایو، پردازنده یا خط ارتباطی) یا موجودیت‌های تجریدی (مثل بانک اطلاعاتی، یک شیء) باشند. بدون توجه به موجودیت منبع، مهندس نرم‌افزار باید یک مکانیزم کنترلی برای آن طراحی کند.

رامباف[RUM91][1] و همکارانش بیان داشتند که هر منبع، باید در اختیار یک شیء محافظ باشد. این شیء محافظ، نگهبان دروازه ورودی برای منبع، کنترل‌کننده دسترسی به آن و متعادل‌کننده تقاضاهای تنش‌زا برای آن است.

۲۲-۲-۷ ارتباط میان زیر سیستم ها

وقتی هر سیستم فرعی مشخص گردید، لازم است همکاری‌هایی را تعریف کرد که بین سیستم‌های فرعی وجود دارند. مدلی که ما برای همکاری شیء به شیء استفاده می‌کنیم می‌تواند به‌عنوان یک کل در سیستم‌های فرعی بسط یابد. شکل ۲۲-۴ یک مدل همکاری را تشریح می‌کند. همان‌گونه که پیش‌تر در این فصل اشاره کردیم، ارتباط می‌تواند بوسیله ایجاد یک رابط خادم / مخدوم یا رابط نظیر به نظیر رخ دهد. با توجه به شکل، ما باید قراردادی را مشخص سازیم که بین سیستم‌های فرعی وجود دارد. به یاد بیاورید که یک قرارداد نشانه شیوه‌هایی است که در آن، سیستم فرعی می‌تواند با دیگری ارتباط متقابل برقرار کند.

قالب پیام	عملیات ها	کلاسها	همکاران	نوع	قرار داد

شکل ۲۲ـ۵ جدول همکاری زیر سیستم

مراحل طراحی زیر را می‌توان برای مشخص کردن قراردادی در مورد یک سیستم فرعی به‌کار گرفت:

پرسش

کدام گامهای طراحی برای تعیین یک " قرارداد " جهت زیر سیستم، مورد نیاز است؟

۱- **هر تقاضایی را که توسط همکاران در سیستم فرعی ارائه می‌شود، یک فهرست کنید.** تقاضاها را به‌وسیله سیستم فرعی سازمان‌دهی نموده و آنها را در یک یا چند قرارداد مناسب تعریف کنید. مطمئن شوید که به قراردادهایی توجه دارید که از کلاس‌های مافوق به ارث رسیده‌اند.

۲- **برای هر قرارداد، به عملیاتی توجه کنید که برای اجرای وظایف اعمال شده از سوی قرارداد، لازمند.** اطمینان حاصل کنید که عملیات با کلاس‌های خاصی که در سیستم فرعی قرار دارند مرتبطند.

۳- **با در نظر گرفتن یک قرارداد در هر زمان، جدولی به شکل نشان داده شده در شکل ۲۲-۵ درست کنید.**

1.Rumbaugh,J.

برای هر قرارداد موارد زیر را در جدول ایجاد کنید:

نوع[1]. نوع قرارداد (یعنی خادم/ مخدوم یا نظیر به نظیر)

همکاران[2]. نام سیستم‌های فرعی که طرفین این قرارداد هستند.

کلاس[3]. نام کلاس‌هایی (در یک سیستم فرعی) که خدمات اعمالی توسط قرارداد را پشتیبانی می‌کنند.

عملیات[4]. نام عملیاتی (درون کلاس) که خدمات را پیاده‌سازی می‌کنند.

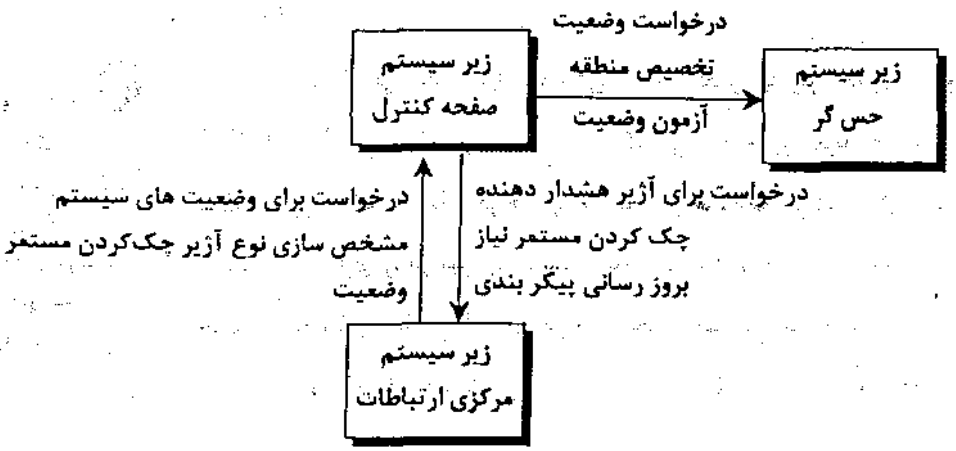
قالب پیام[5]. قالب پیام برای پیاده‌سازی ارتباط بین همکاران لازم است.

توصیف مناسب پیام را برای هر ارتباط متقابل بین سیستم‌های فرعی را مطرح کنید.

تعاریف پرسش

هر قراردادی بین زیرسیستم ها، با یک پیام یا بیشتر که در میان زیرسیستم ها و بین اشیاء حرکت می نماید، مشخص می گردد.

۴- **اگر حالات محاوره بین سیستم‌های فرعی پیچیده‌اند، نمودار سیستم فرعی ـ همکاری که در شکل ۲۲-۶ آمده ایجاد می‌شود.** نمودار همکاری از نظر فرم شبیه دیاگرام جریان وقایع است که در فصل ۲۱ مورد بحث قرار گرفت. هر سیستم فرعی همراه با تعاملش با سیستم‌های فرعی دیگر، بازنمایی می‌شود. قراردادهایی که در طول یک رابطه متقابل ایجاد می‌شوند همان‌گونه که نشان داده شده مورد نظرند. جزئیات رابطه متقابل با جستجو کردن قرارداد در جدول همکاری سیستم فرعی تعیین می‌شوند. (شکل ۲۲-۵).



شکل ۲۲ـ۶ گراف همکاری برای زیر سیستم خلاصه سازی خانه امن

1.type
2.collaboratores
3.class
4.operation
5.message format

۲۲-۳ فرآیند طراحی شیء

با وام‌گیری از استعاره‌ای که پیش‌تر در این کتاب معرفی شد، ممکن است طراحی سیستم شیءگرا را به‌عنوان طرح و نقشه طبقه اول و همکف یک ساختمان در نظر گرفت. طرح پایه یا همکف منظور از ساخت هر اتاق و مشخصه‌های معماری که اتاق‌ها را به یکدیگر و به محیط خارج مرتبط می‌سازد را بیان می‌دارد. حالا وقت آن فرا رسیده که جزئیاتی مهیا شوند که برای ساخت هر اتاق لازمند. در بافت و متن طراحی شیءگرا، طراحی شیء روی «اتاق‌ها» متمرکز می‌شود.

بنت و همکارانش طراحی شیء را به‌صورت زیر مورد بحث قرار می‌دهند:[BEN99] [1]

طراحی شیء مربوط به طراحی دقیق اشیا و روابط متقابل آنهاست. این طراحی در معماری کلی که در طول طراحی سیستم و مطابق رهنمودها و پروتکل‌های مورد توافق در طراحی، تعریف شده است، تکمیل می‌گردد. طراحی شیء به‌طور ویژه با مشخصه انواع صفات خاصه، چگونگی کارکرد عملیات‌ها و چگونگی ارتباط اشیاء با دیگر اشیاء مربوط می‌شود.

در این مرحله است که مفاهیم مقدماتی و اصول اولیه مربوط به طراحی در سطح جزء وارد کار می‌شوند. ساختارهای موضعی اطلاعاتی (برای صفات خاصه) تعریف شده و الگوریتم‌ها (برای عملیات) طراحی می‌شوند.

۲۲-۳-۱ توصیفات شیء

توصیه

پیش از آنکه کار بر روی اشیاء را آغاز کنید، اطمینان حاصل نمایید که معماری تعریف شده است. اجازه ندهید که معماری بعداً تعریف شود.

توصیف طراحی یک شیء (مصداقی از یک کلاس یا زیر کلاس) می‌تواند به دو شکل باشد: [GOL83] [2] ۱) توصیف پروتکل [3] که رابط یک شیء را از طریق تعریف هر پیامی که شیء دریافت کرده و عملیات مربوطه که به هنگام دریافت پیام انجام می‌دهد، ارائه می‌کند یا ۲) توصیف نحوه پیاده‌سازی [4]، که جزئیات پیاده‌سازی را در مورد هر عملیات اعمال شده از سوی پیامی که به شیء می‌رسد، نشان می‌دهد. جزئیات پیاده‌سازی شامل اطلاعاتی در مورد بخش خصوصی شیء است، یعنی جزئیات داخلی در مورد ساختارهای داده‌ای که جزئیات مربوط به روال کار و صفات خاصه شیء را که عملیات را تشریح می‌کنند، توصیف می‌نمایند.

توصیف پروتکل چیزی بیشتر از مجموعه‌ای از پیام‌ها و توضیح مربوطه در مورد هر پیام نیست. مثلاً، ممکن است قسمتی از توصیف پروتکل در مورد شیء Motion Sensor (حسگر حرکتی) این چنین باشد:

---

1 Bennett,S.

2 Goldberg,A.and D.

3 protocol description

4 implementation description

MESSAGE (Motion. Sensor) → read : RETURNS sensor. ID, sensor ;

که پیام لازم برای خواندن سنسور را توصیف می‌کند. بهمین شکل:

MESSAGE (motion. sensor) →set: SENDS sensor. ID, sensor. Status ;

که راه‌اندازی مجدد وضعیت سنسور را انجام می‌دهد.

برای سیستمی بزرگ با پیام‌های متعدد، اغلب ممکن است طبقات مختلفی برای پیام ایجاد نمود. مثلاً طبقه‌بندی پیام‌ها در مورد شیء سیستم در خانهٔ امن ممکن است شامل پیام‌های پیکربندی سیستم، پیام‌های کنترلی و نظارتی، پیام‌های وقایع و غیره باشد.

یک توصیف پیاده‌سازی شیء، جزئیات داخلی (پنهان شده) را مهیا می‌کند که برای پیاده‌سازی لازمند اما برای فراخوانی ضروری نیستند. یعنی طراحی شیء باید توصیف پیاده‌سازی را مهیا ساخته و بنابراین جزئیات داخلی شیء را خلق می‌کند. طراح یا پیاده‌سازی کننده دیگری که از شیء یا دیگر مصداق شیء استفاده می‌کند، تنها به توصیف پروتکل نیاز دارد اما توصیف پیاده‌سازی را نمی‌خواهد.

برای بدست آوردن مزایای پنهان سازی اطلاعات (فصل ۱۳) هرکس که شئ ای را مورد استفاده قرار می دهد، نیاز به تعریف پروتکل دارد. توصیفات پیاده سازی شامل جزئیاتی است که بایداز دید کسانی که به این جزئیات نیازی ندارند، مخفی بماند.

توصیف پیاده‌سازی متشکل از اطلاعات زیر است: ۱) مشخصه نام شیء و مرجع کلاس ۲) مشخصه ساختار داده خصوصی با اشاره به اقلام داده و انواعشان ۳) توصیف روال هر عملیات یا نشانگرهایی در مورد چنین توضیحاتی، توصیف پیاده‌سازی باید حاوی اطلاعات کافی بوده تا ارائه تمام پیام‌های توصیف شده در توضیح پروتکل را مهیا سازد.

کاکس[COX85][1] تفاوت بین اطلاعات موجود در توصیف پروتکل و اطلاعاتی که در توصیف پیاده‌سازی است را از نظر کاربران و تأمین‌کنندگان خدمات، تشریح می‌کند. کاربر خدمات که از شیء استفاده می‌کند باید با پروتکل فراخوانی خدمات آشنا باشد، یعنی با مشخص نمودن آنچه که مورد تمایلش است. تأمین‌کنندگان خدمات باید در مورد چگونگی ارائه خدمات به کاربر فکر کنند یعنی جزئیات پیاده‌سازی.

۲۲-۳-۲ طراحی الگوریتم ها و ساختارهای داده ای

ارجاع به فصلها

هر مفهومی که به طور مجازی در فصل ۱۳ ارائه گردیده است، در اینجا کاربردی خواهد شد. اطمینان حاصل کنیدکه با مباحث ارائه شده در اینجا، آشنا می باشید.

طیف گسترده‌ای از بازنمایی موجود در مدل تحلیل و طراحی سیستم، مشخصه‌ای برای تمام عملیات و صفات خاصه مهیا می‌سازند. الگوریتم‌ها و ساختارهای داده‌ای با استفاده از رهیافتی طراحی شده‌اند که تفاوت بسیار کمی با رهیافت‌های طراحی در سطح جزء و طراحی داده‌ها که در مهندسی نرم‌افزار متعارف مورد بحث قرار گرفت، دارند.

یک الگوریتم ایجاد می‌شود تا مشخصه‌ای را برای هر عملیات پیاده کند. در بسیاری از موارد، الگوریتم یک رشته محاسباتی یا رویه‌ای ساده است که می‌تواند بعنوان یک پیمانه نرم‌افزاری فراگیر پیاده شود. اگر

1.Cox,B.

تعیین عملیات سخت باشد، ممکن است نیاز به پیمانه سازی عملیات باشد. فنون متعارف طراحی در سطح جزء را می‌توان برای رسیدن به این منظور، استفاده نمود.

ساختارهای داده به‌طور هم‌زمان با الگوریتم‌ها طراحی می‌شوند. از آن‌جا که عملیات به‌طور ثابت صفات خاصه یک کلاس را تغییر می‌دهند، طراحی ساختارهای داده‌ای که به بهترین وجه صفات خاصه را منعکس می‌سازند، بار سنگینی روی طراحی الگوریتم‌های عملیات مربوطه دارد.

پرسش

آیا راهی برای طبقه بندی عملیات (شیوه ها) وجود دارد؟

با این‌که انواع عملیات مختلفی وجود دارند، آنها را معمولاً می‌توان به سه گروه عمده تقسیم کرد: ۱) عملیاتی که داده را به شکل تغییر می‌دهند (مثل افزودن، حذف، قالب بندی مجدد، انتخاب)، ۲) عملیاتی که محاسبه را انجام می‌دهند. ۳) عملیاتی که بر شیء را از نظر وقوع یک حادثه کنترلی، نظارت می‌نمایند.

مثلاً، روال پردازش خانهٔ امن حاوی قطعات جمله است: «سنسور دارای یک شماره و نوع است» و «کلمه عبور اصلی برای تجهیز کردن و غیرفعال نمودن سیستم به‌کار می‌رود» این دو عبارت نشان‌گر چند چیز هستند:

- یک عملیات انتساب[1] مربوط به شیء سنسور
- عملیات برنامه‌ای که در مورد شیء سیستم به‌کار می‌رود.
- مجهز نمودن[2] و خلع سلاح[3] نمودن که عملیاتی هستند که در مورد سیستم به‌کار می‌روند (هم‌چنین وضعیت آن سیستم ممکن است نهایتاً با استفاده از علائم فرهنگ داده‌ها تعریف شود) مانند:

system status=[armed / disarmed]

توصیه

پالایش عملیات مشابه راهی است که در پالایش یک تابع هنگام طراحی متعارف می پیمابیم. یک قصه پردازشی را بنویسید، یک تجزیه دستوری انجام دهید، و عملیات جدید را از سطح پایین تر تجریدی، جدا سازید.

برنامه عملیات در طول تحلیل شی‌گرا تخصیص می‌یابد، اما در طول طراحی شیء به‌صورت تعدادی عملیات خاص‌تر پالایش می‌شود که برای طراحی سیستم لازمند. مثلاً، بعد از بحثهایی با مهندسی تولید، تحلیلگر و احتمالاً بخش بازاریابی، طراح ممکن است روال پردازش اصلی را بسط داده و برنامه زیر را بنویسید (زیر عملیات بالقوه خط کشیده شده است):

برنامه، کاربر خانهٔ امن را قادر می‌سازد تا وقتی سیستم نصب شد پیکربندی را انجام دهد. کاربر می‌تواند ۱) شماره تلفن‌ها را در آن قرار دهد ۲) زمان‌های تأخیر را در مورد زنگ‌های خطر مشخص کند ۳) جدول سنسوری بسازد که حاوی ID هر سنسور، نوع آن و محلش باشد و ۴) یک کلمه رمز اصلی را وارد کند (بار می‌کند).

بنابراین طراح، برنامه تک عملیاتی را بازیابی نموده و آن را جایگزین این عملیات می‌کند: نصب[4]، تعریف[1]، ساخت[2] و بارگذاری[3]. هر یک از عملیات جدید بخشی از شیء سیستم هستند و شناختی از

1.assign
2.arm
3.disarm
4.install

ساختارهای اطلاعاتی داخلی دارند که صفات خاصه مربوط به شیء را پیاده می‌کنند و با ارسال پیام‌های به شیء بصورت:

MESSAGE (system) → install : SENDS telephone.number

تحریک می‌شود که نشان‌گر تهیه شماره تلفن اضطراری برای سیستم است و یک پیام نصب برای سیستم ارسال می‌شود.

افعال بر اعمال یا وقایع اشاره دارند. در متن فرموله کردن طراحی شیء، نه تنها افعال بلکه عبارت فعلی توصیفی و مسندها را (مثل عبارت «برابر است با» is equal to) به‌عنوان عملیات بالقوه در نظر می‌گیریم. تجزیه گرامری به‌صورت مکرر به‌کار گرفته می‌شود تا وقتی که هر عملیات به‌صورت تفصیلی، بازیابی شود.

۲۲-۳-۳ اجزاء برنامه و رابط

یک جنبه مهم از کیفیت نرم افزار پیمانه ای بودن است. که به معنای ترکیب اجزاء (پیمانه) های به شکل یک برنامه می باشد. رهیافت شئ گرا شئ را به صورت یک جزء برنامه تعریف می کند که به دیگر اجزاء متصل خواهد شد(مانند داده های خصوصی و عملیات). اما تعریف اشیاء و عملیات به تنهایی کافی نخواهند بود. در طی طراحی رابط میان اشیاء و ساختارهای آن اشیاء (از دید معماری) نیز باید تعریف شوند.

هرچند یک مولفه و جزء برنامه به صورت انتزاعی تعریف می شود باید توسط یک زبان برنامه سازی پیاده سازی شود. در طراحی شئ گرا، زبان برنامه سازی مورد استفاده باید قابلیت ایجاد اجزاء برنامه زیر را دارا باشد (به شکل زبان Ada) :

1.define

2.build

3.load

```
PACKAGE program-component-name IS
TYPE specification of data objects
.

   PROC specification of related operations. . .
PRIVATE
data structure details for objects
PACKAGE BODY program-component-name IS
PROC operation.1 (interface description) IS
.

END

PROC operation.n (Interface description) IS
.

END
END program-component-name
```

با نظر به زبان طراحی برنامه ای چون Ada که اکنون ملاحظه کردید، یک جزء برنامه به همراه داده ها و عملیاتش تعریف می شود. قسمت مشخصه سازی اجزاء، تمام اشیاء داده ای را تعریف می کند (این تعریف به کمک جمله **type** عملی گردیده است) و نیز عملیاتی که بر روی آنها کار می کند (در اینجا **proc** برای procedure آمده است). قسمت خصوصی – اختصاصی یا محلی **(private)** جزئیاتی از ساختار داده ای و پردازشها را در بر دارد که از دید دیگران پنهان گردیده است. با توجه به بحثی که پیشتر داشتیم، بسته **(package)** از نظر مفهومی مشابه اشیاء ای است که در این فصل توضیح داده شد.

جزء نخست برنامه ای که تعریف شد، پیمانه ای سطح بالا را نشان می دهد که کار با تمام ساختارهای داده ای را در بر گرفته است. یکبار دیگر به مثال خانه امن بازگردید، می توان جزء سطح بالایی به قرار زیر را برای آن تعریف نمود :

```
PROCEDURE SafeHome software
```

جزء یا مؤلفهٔ خانهٔ امن با بسته‌هایی (اشیائی) جفت خواهد شد که طراحی اولیه آنها در ذیل آمده است :

PACKAGE system IS
TYPE system data
PROC Install, define, build, load
PROC display, reset, query, modify, call
PRIVATE
PACKAGE BODY system IS
PRIVATE
system.Id IS STRING LENGTH (8);
verification phone.number, telephone. number, ...
IS STRING LENGTH (8);
sensor. table DEFINED
sensor. type IS STRING LENGTH (2),
sensor. number, alarm. threshold IS NUMERIC;
PROC Install RECEIVES (telephone.number)
{design detail *for* operation Install}

END system
PACKAGE sensor IS
TYPE sensor data
PROC read, set, test
PRIVATE
PACKAGE BODY sensor IS
PRIVATE
sensor.Id IS STRING LENGTH (8);
sensor. status IS STRING LENGTH (8);
alarm. characteristics DEFINED
threshold, signal type, signal level IS
NUMERIC,
hardware. interface DEFINED
type, a/d.characteristics, timing. data IS NUMERIC,
END sensor

END SafeHome software

اشیاء داده ای و عملیات متناظر آنها برای هر جزء از نرم افزار خانه امن تشریح گردیده اند. گام نهایی در فرآیند طراحی شئ تکمیل تمام اطلاعات مورد نیاز برای پیاده سازی کامل ساختار داده‌ها و نوع‌ها می باشد. ساختمان داده‌ها و نوع‌هایی که شامل بخش **private** بسته بوده و تمام جزئیات رویه ای بدنه بسته **(package body)** را در برداشته باشند.

برای توضیح طراحی تفصیلی جزء برنامه، دوباره بسته سنسور که پیشتر توضیح داده شد، را در نظر آورید. ساختارهای داده ای مربوط به صفات خاصه سنسور تعریف شده اند. بنابرین، گام نخست تعریف رابطهایی برای هر یک ازعملیات مربوط به سنسور می باشد:

PROC read (sensor.id, sensor.status: OUT);
PROC set (alarm.characteristics, hardware.thterface: IN)
PROC test (sensor.id, sensor. status , alarm. characteristics: OUT);

گام بعدی، پالایش گام به گام عملیاتی است که به بسته سنسور مربوط می شود. برای توضیح این پالایش یک حکایت پردازشی (یک راهبرد غیر رسمی) را برای *خواندن (read)* خواهیم آورد :

هنگامی که یک شئ سنسور یک پیام خواندن را دریافت می کند فرآیند خواندن صدا زده می شود. فرآیند نوع رابط و سیگنال را معین می کند، با رابط سنسور ارتباط برقرار می سازد، خصوصیات فیاسی / رقمی (A/D) را به سطح سیگنال داخلی تبدیل می کند، و سطح سیگنال

داخلی را با یک مقدار آستانه مقایسه می کند. اگر مقدار از آستانه عبور کرده بود، وضعیت سنسور به "رویداد" تنظیم می شود. در غیر این صورت وضعیت سنسور به "عدم رویداد" تنظیم می گردد. اگر هنگام برقراری ارتباط با سنسور خطایی تشخیص داده شود، وضعیت سنسور به "خطا" تنظیم می گردد.

با این حکایت پردازشی، یک PDL (زبان طراحی برنامه) که فرآیند *خواندن* را توصیف می کند به قرار ذیل ایجاد خواهد شد:

```
    PROC read (sensor.id, sensor. status: OUT);
    raw. signal IS BIT STRING
    IF (hardware. interface. type = "SII & alarm.characteristics.signal. type =
"BII
THEN
    GET (sensor, exception: sensor.status := error) raw.signal;
     CONVERT raw.signal TO internal.signal.level;
    IF internal.signal.level > threshold
    THEN sensor. status : = "event";
    ELSE sensor. status : = "no event";
    ENDIF
    ELSE {processing for other types of s interfaces would be specified}
ENDIF
    RETURN sensor.id, sensor. status;
    END read
```

بازنمایی PDL از عملیات خواندن *(read)* به یک زبان مناسب پیاده سازی ترجمه خواهد شد. فرض می شود که فانکشن **get** و **convert** در کتابخانه های زمان اجراء (از قبل طراحی شده) و در اختیار می باشند.

پالایش گام به گام و برنامه سازی ساختیافته (فصل ۱۶) در این مرحله جهت تکمیل طراحی هر عملیات به کار می رود.

۲۲-۴ الگوهای طراحی

بهترین طراحان در هر زمینه، دارای توانایی غیرمعمولی برای مشاهده الگوهایی هستند که مسئله و الگوهای مربوطه‌اش را توصیف می‌کند، الگویی که می‌توانند با هم ترکیب شوند تا راه‌حلی بیابند. گاما و همکارانش [GAM95][1] این مسئله را وقتی مورد بحث قرار می‌دهند که می‌گویند:

شما الگوهای برگشتی متعددی از کلاس‌ها و اشیای ارتباط برقرارکننده در بسیاری از سیستم‌های شیءگرا خواهید یافت. این الگوها مشکلات به‌خصوصی در طراحی را حل می‌کنند و طراحی شیءگرا را انعطاف‌پذیرتر، عالی‌تر و نهایتاً قابل استفاده مجدد خواهند ساخت. آنها به طراح کمک می‌کنند با مبنا قرار دادن طراحی‌های جدید بر اساس تجربه قبلی، طراحی‌ها را با موفقیت مورد استفاده مجدد قرار دهند. طراحی که با چنین الگوهایی آشناست می‌تواند فوراً آنها را در مسایل طراحی بدون اجبار به کشف مجدد آنها به‌کار گیرد.

ارجاع به فصلها

الگوها در معماری و سطح اجزاء وجود دارند. برای توضیح بیشتر به فصل ۱۴ مراجعه نمایید.

در طول فرآیند طراحی شیءگرا، مهندس نرم‌افزار باید در جستجوی فرصتی باشد تا از الگوهای موجود طراحی مجدداً استفاده کند نه این‌که الگوهای جدیدی خلق نماید.

۲۲-۴-۱ یک الگوی طراحی چیست ؟

قبل از این‌که نگاه دقیق‌تری به یک الگو بیاندازیم، بهتر است به مثال ساده‌ای از یک الگو توجه کنیم که بارها و بارها ظاهر شده‌اند. بسیاری از برنامه‌های کاربردی نیازمند آن هستند که تنها یک مورد از نیازهای هر شیء تحریک گردند. نمونه‌هایی از برنامه‌های کاربردی و اشیای تک موردی عبارتند از: [GAM95]

ارجاع به وب

یک مقاله ممتاز با عنوان مثالهای غیر نرم افزاری از الگوهای نرم افزار تهیه و در آدرس زیر ارائه گردیده است:
www.agcs.com/patterns/papers/patexamples.html

- در یک سیستم عامل تنها یک شیء مدیریت فایل وجود دارد که مسیر فایل‌های کاربر را نگهداری نموده و تسهیلاتی برای ایجاد، نام‌گذاری مجدد و حذف چنین فایل‌هایی مهیا می‌کند. تنها یک مورد از مدیر فایل این‌چنینی وجود دارد.
- در یک سیستم کنترل ترافیک هوایی تنها یک مورد کنترل‌کننده وجود دارد که مسیر هواپیماها و موقعیت آنها را پی‌گیری می‌کند.

نقل قول

(الگوها) به منزله ریشه گیاهان است، اجتماعی از تجربیات مهندسین نرم افزار و طراحان ماهر.
فریک بوچمن ات آل

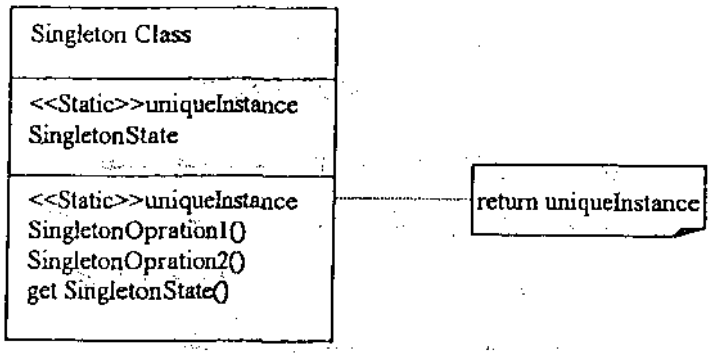
1.Gamma,E.

نقل قول

یک الگوی طراحی اگر منجر به مسائل حل نشده شود، تبدیل به ضد الگو خواهد شد.

ویلیام براون ات آل

• در یک برنامه بانکداری، تنها یک کنترل‌کننده وجود دارد که مسیر دستگاه‌های ATM مورد استفاده بانک را نگهداری می‌کند.



شکل ۲۲ـ۷ساختار کلی Singleton

شکل ۲۲-۷ ساختار کلی این الگو را نشان می‌دهد. متن <<Static>> [1] متغیر گسترده یک کلاس را توصیف می‌کند. در این شکل تنها دو عملیات روی یک سینگلتون (مورد منحصر بفرد) نشان داده می‌شوند، ممکن است بسته به بافت و ترکیب موارد متعدد دیگری،نیز وجود داشته باشد. ساختار یک جاوا برای الگو در زیر آمده است:

```
public class Singleton
{
private static SingleObject uniqueInstance = null;
    public static SingleObject uniqueInstance()
    {
    if (uniqueInstance == null)
       uniqueInstance = new SingleObject();
    return uniqueInstance;
    }
// Code for any constructor, these will be private
// Code for any methods which implement write
// operations on the Singleton, this might include
// operation1 and operation2
// Code for any methods which implement retrieval operations
// on the Singleton object, this might include operation1
// operation2
}
```

1.Static

این الگو منحصر به‌فرد از طریق متغیر موردی توصیفی به‌وسیله کلاس SingleObject اجرا شده است. در هنگام شروع این به منزله پوچ تلقی می‌شود.

دسترسی به شیء منحصر به‌فرد از طریق روش uniqueInstance می‌باشد. این روش ابتدا چک می‌کند که آیا این سینگلتون پوچ است یا خیر؟ اگر پوچ بود آن وقت یعنی هیچ شیء منحصر به‌فردی ایجاد نشده و سپس خود روش سازنده شخصی مناسب را فرا می‌خواند تا سینگلتون را تشکیل دهد. در کُد فوق این سازنده، آرگومان صفر است. سازنده مورد استفاده، خصوصی اعلام می‌شود زیرا ما نمی‌خواهیم هیچ کاربری اشیای منحصربه‌فرد یا سینگلتون را جز از طریق uniqueInstance ایجاد کند که تنها و یکبار برای همیشه ایجاد می‌شود. این کلاس به‌طور معمول حاوی روش‌هایی است که عملیاتی را روی شیء منحصر به‌فرد انجام می‌دهند.

بنابراین اگر الگوی سینگلتون در سیستم کنترل ترافیک هوایی استفاده می‌شد و تنها یک کنترل‌کننده هواپیما برای ارائه لازم بود آن‌وقت نام کلاس فوق ممکن بود Air Controller بوده و دارای روش‌هایی مثل getAircontroller باشد که با مورد یکتایی از یک کنترل‌کننده ترافیک هوایی بی‌نظیر بازمی‌گردد.

ارجاع به وب

انبار الگوی پوریلند (PPR) یک مجموعه از الگوهای طراحی را گرد آورده است: www.c2.com/ppr

۲۲-۴-۲ مثالی دیگر از یک الگو

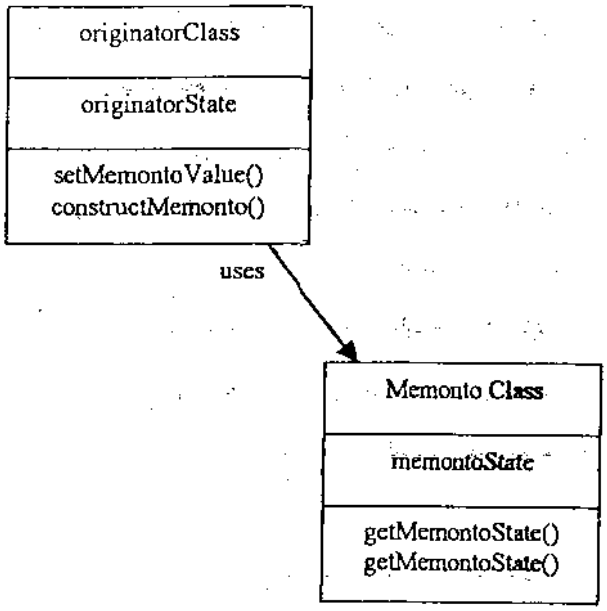
هم‌اکنون نمونه‌ای از یک الگو[1] به نام سینگلتون را دیده‌اید. هدف از این بخش توصیف الگوی پیچیده‌تری بنام میمنتو (Memento) است.

نقش الگوی میمنتو عبارتست نظارت بر ذخیره و بازیابی وضعیت سیستم وقتی انجام این کار لازم است. شکل ۲۲-۸ نمودار کلاس را برای این الگو نشان می‌دهد. سه عنصر در این الگو وجود دارد. اولی OriginatorClass است. این کلاس‌ای است که اشیایی را توصیف می‌کند که وضعیت آنها باید ضبط شود. دو روش به این کلاس مربوط می‌شود:

setMementoValue و constructMemento . اولی وضعیتش را مجدداً تنظیم کرده، آرگومانی می‌گیرد که شیءای است تعریف شده توسط کلاس میمنتو و با استفاده از آرگومانی مجدداً وضعیتش را تنظیم می‌کند. دومی شیء MementoClass را ایجاد می‌کند که حاوی وضعیت کنونی آن است. در عمل SetMemento مثل تسهیلات بازیابی عمل کرده و constructMemento نیز مثل محل ذخیره‌سازی عمل می‌کند.

---

۱. بوچمن [BUS96] و گاما [GAM95] در میان بسیاری دیگر کاتالوگهایی از الگوهای طراحی برای استفاده در سیستم های شئ گرا، نوشته اند.



شکل ۲۲ـ۸ الگوی Memonto

کلاس MementoClass اشیایی را تعریف می‌کند که دارای وضعیت شیء OriginatorClass هستند. این کلاس دارای دو روش getMementoState و setMementoState است. اولی به حالتی برمی‌گردد که در حال ذخیره‌سازی است، در حالی‌که دومی وضعیت را با مقداری تطبیق می‌کند که به‌عنوان یک آرگومان پذیرفته شده است. سومین عنصر الگوی میمنتو کلاس مشتری یا Caretaker است که در شکل ۲۲-۸ نشان داده شده است. این کلاس نمایان‌گر کلاسی است که اشیایی را پیاده‌سازی می‌کنند که حاوی شیء MementoClass است. این کلاس دارای مجموعه عملیات محدودی می‌باشد. به‌طوری‌که تمام کاری که انجام می‌دهد عبارتست از ذخیره کردن یک میمنتو، این عملیات محتویات میمنتو را تغییر نداده یا آزمون نمی‌کنند.

۲۲-۴-۳ مثال نهایی از یک الگو

اغلب نیاز به توسعه برنامه‌ای وجود دارد که پردازش اطلاعاتی را انجام می‌دهد که به‌صورت تسلسلی مورد دسترسی هستند. این دسترسی تسلسلی معمولاً دارای فرم یکسانی بوده و بنابراین برای یک الگو قابل استفاده است. هدف از این بخش عبارتست از توصیف چنین الگویی، این الگو منتسب به گرانت [GRA99] است. نمونه‌هایی از این نوع پردازش تسلسلی عبارتند از:

- برنامه گزارشی که ممکن است هر بار به هنگام خواندن یک رکورد، یک فایل اطلاعات مربوط به کارمندی را پردازش کرده و بقیه رکوردهای کارمندانی را که بالاتر از مقدار معینی حقوق دارند، رد کند.
- برنامه ویرایشگری که از طرف کاربر از آن خواسته می‌شود، خطوطی از متن را در یک فایل فهرست‌بندی کند که با الگوی خاصی مطابقت یابد.

- یک برنامه تحلیل وب ممکن است منبع سند HTML را بخواند تا معلوم کند چند ارجاع به سایت‌های دیگر در آن سند وجود دارند.

همه این‌ها اشکال مختلف پردازشند. در هر حال آنها به‌وسیله این حقیقت که اطلاعات تسلسلی پردازش می‌کنند با هم مرتبط شده‌اند. این کار ممکن است به‌صورت کلمه به کلمه یا رکورد به رکورد باشد. با این وجود مقداری پردازش تسلسلی به‌کار گرفته شده و فرصت خوبی است که در یک الگو به‌کار گرفته شود. این الگو با نام فیلتر، در شکل ۲۲-۹ آمده است. فیلتر شامل چند کلاس است:

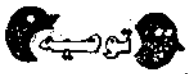
- Source Filter . این کلاس‌ای است که مانند یک کلاس مافوق عمل نموده تا کلاس‌هایی را که پردازش مورد نیاز را انجام می‌دهند، بپوشاند. این کلاس، مجموعه وظائف بازیابی اطلاعاتی را که قرار است پردازش شوند انجام نمی‌دهد اما آن را به شیء Source می‌فرستد که کلاس مربوطه‌اش در دیدگاه سوم که در زیر آمده توصیف می‌شود. شیء Source از طریق سازنده به کلاس انتقال می‌یابد. کلاس شامل روش گرفتن اطلاعات (getdata) است که اطلاعاتی را که قرار است پردازش شوند، بازیابی می‌کند.
- ConcreteSourceFilter . این مجموعه کلاس فرعی SourceFilter است. این کلاس از روش getdata صرفنظر می‌کند تا عملیات اضافی روی داده‌های خوانده شده انجام دهد مثلاً اگر الگو برای شمارش رشته‌هایی استفاده شود که با الگوی خاصی مطابقت داشته باشند سپس برنامه‌ای برای انجام این شمارش در این‌جا قرار داده می‌شود. معمولاً این روش از روش گرفتن اطلاعات مربوطه در کلاس مافوق استفاده می‌کند.
- Source. این کلاس با اشیایی ارتباط دارد که پردازش بازیابی داده‌ها را انجام می‌دهند که قرار است پردازش شوند. این کار با روش getdata انجام می‌شود.
- ConcreteSource . این کلاس زیر کلاس Source است و روش getdata را اجرا می‌کند. عملکرد آن عبارتست ازمهیا ساختن داده‌هایی در اشیای مربوط به کلاس‌های ConcreteSourceFilter که بعضی از پردازش‌های روی اطلاعات را انجام می‌دهد.

۲۲-۴-۴ توصیف یک الگوی طراحی

اصول مهندسی سطح عالی از الگوهای طراحی استفاده گسترده‌ای می‌نماید. مهندسی نرم‌افزار با این الگوها تنها در مراحل آغازین قرار دارد و به سرعت به‌سوی انجام طبقه‌بندی حرکت می‌کند. در توصیف کلی الگویی که تشکیل‌دهنده بخشی از طبقه‌بندی است باید داشته باشیم[GAM95] :

- نام الگو مثلاً Filter
- هدف از این الگو مثلاً ممکن است هدف یک الگو ساده‌سازی نگهداری و مرمت باشد. زیرا می‌تواند چند نوع شیء مختلف را در نظر بگیرد.

- «نیروهای طراحی» که الگو را سبب می‌شوند. مثلاً ممکن است طرحی با الگویی ارائه شود، به‌طوری‌که تعدادی از انتقالات و تغییرات مختلف داده‌ها را می‌توان در مورد یک شیء به‌کار گرفت که بسیاری از آنها وقتی اصولاً الگو ارائه می‌شود، ناشناخته هستند.
- راه‌حلی که این نیروهای طراحی را راحت‌تر می‌سازد.
- کلاس‌هایی که برای اجرای راه‌حل لازم است. مثلاً، چهار کلاس توصیف شده در شکل ۲۲-۹.
- مسئولیت‌ها و همکاری‌های میان کلاس‌ها. به‌طور مثال، ممکن است یک کلاس مسئول ساخت شیء باشد که رفتارش در زمان اجرا تفاوت می‌کند.
- راهنما از نظر برنامه‌سازی که منجر به اجرای کارآمدتری می‌شود. معمولاً چند شیوه برنامه‌نویسی مختلف برای یک الگو وجود دارد. به‌طور مثال، پردازش خطا ممکن است با چند شیوه سر و کار داشته باشد.



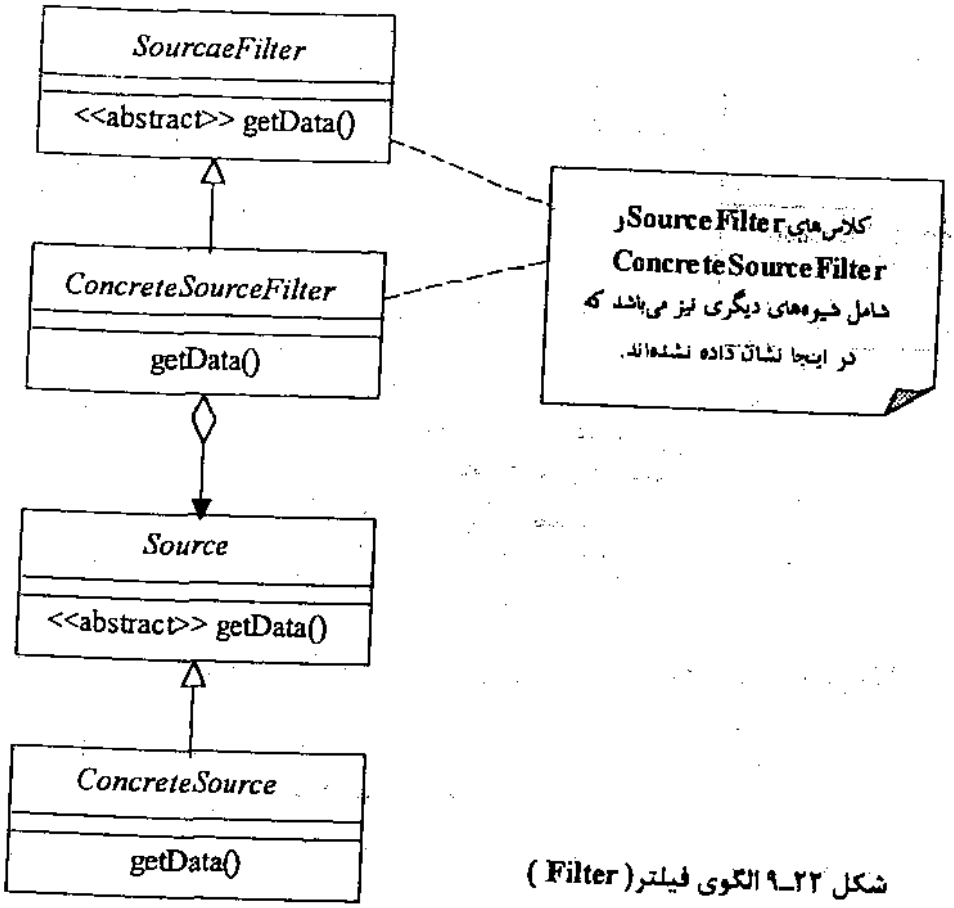
طراحی خوب همواره مبتنی بر ساده سازی است. بنابراین هنگامی که ساختارهای ارث بری ساده تر وجود دارد، از ترکیب بپرهیزید.

- مثال‌هایی از کُد منبع
- ارجاع منابع به الگوهای طراحی مربوطه یا الگوهایی که بتوانند در ارتباط با الگوی توصیفی استفاده شوند.

۲۲-۴-۵ آینده الگوها

در حال حاضر تعداد الگوهای نسبتاً کمی ارائه و کاتالوگ شده‌اند. پنج سال اخیر شاهد انفجار عظیمی از علایق صنعتی بوده‌ایم. الگوها همراه با اجزاء این امیدواری را می‌دهند که مهندسی نرم‌افزار نهایتاً هم‌چون دیگر اصول مهندسی درباره کلاس‌ها نرم‌افزار قابل قیاس با قطعات الکترونیکی شده و الگوها قابل قیاس با مدارهای کوچکی تبدیل شوند که از قطعات ساخته شده‌اند. قبل از این‌که این اتفاق رخ دهد، کار زیادی باید برای شناسایی الگوها و کلاس‌بندی آنها صورت گیرد. هم‌چنین نکته این است که وقتی تعدادی از الگوهای موجود بسیار بزرگ می‌شوند یک شکلی از شاخص‌گذاری خودکار یا نیمه خودکار لازم می‌شود.

هم اکنون الگوهای اندکی وجود دارند اگر چه چند سال بعد شاهد یک انفجار واقعی خواهیم بود

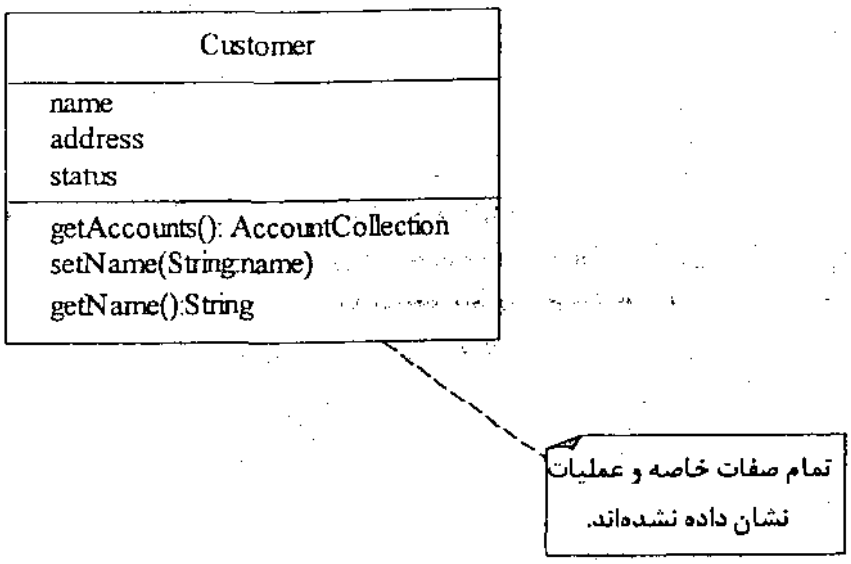


شکل ۲۲ـ۹ الگوی فیلتر (Filter)

۲۲-۵ برنامه سازی شئ گرا

UML بصورت یک استاندارد بالفعل برای تحلیل و طراحی شیءگرا در آمده است

هدف از این بخش توصیف کمی دقیق‌تری از مجموعه روش‌هایی است که زبان UML را می‌سازند. تاکنون در فصل ۲۱ بطور خلاصه تولد آن و اجزای اصلی‌اش را تشریح نمودیم. در واقع بسیاری از نمودارهای ارائه شده در فصل ۲۱ و این فصل به UML نوشته و بیان شده‌اند. تصمیم گرفته‌ایم تا این مجموعه روش‌ها را بسط دهیم، زیرا به سرعت در شرکت‌هایی گسترده شده‌اند که از ایده‌های مهندسی برای توسعه نرم‌افزار شیءگرا استفاده کرده‌اند. اولین جزیی که قرار است تشریح کنیم، مدل کلاس است.



شکل ۲۲ـ۱۰ یک مثال از یک کلاس توصیف شده در UML

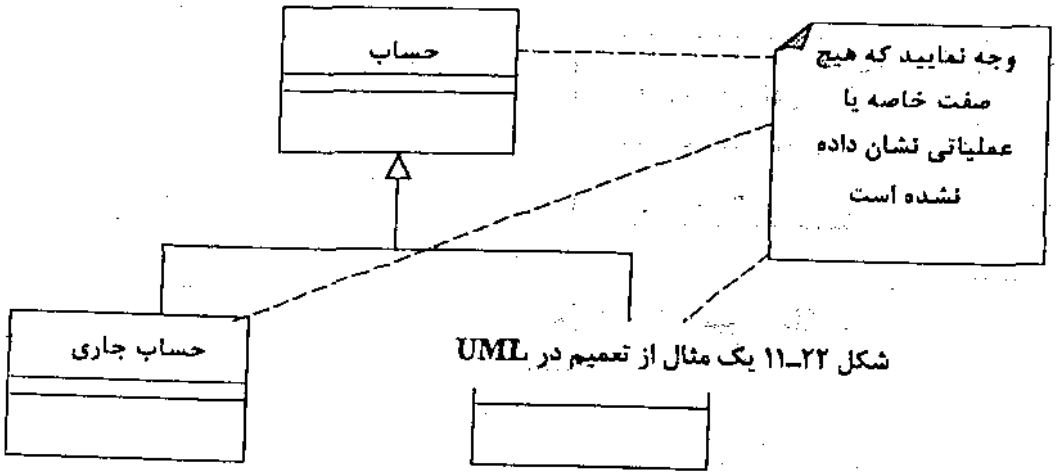
۲۲-۵-۱ مدل کلاس

یک مدل کلاس عبارتست از توصیف کلاس‌های یک سیستم و ارتباطات آنها. این مدل رفتار پویای سیستم مثلاً رفتار تک‌تک اشیا را توصیف نمی‌کند. اولین عنصر نمودار کلاس عبارتست از توصیف هر یک از کلاس‌های مستقل. شکل ۲۲-۱۰ نشان می‌دهد که چگونه یک کلاس توضیح داده می‌شود. این کلاس یک مشتری بانک را توصیف می‌کند.

این شکل بسیار ساده است زیرا تنها حاوی یک کلاس می‌باشد. این کلاس شامل نام کلاس (مشتری- Customer)، نام بعضی از صفات مشخصهٔ آن مثل صفت مشخصه آدرس که حاوی آدرس مشتری است و فهرستی از عملیات مثل عملیات getname می‌باشد که نام مشتری را بازیابی می‌کند. هر باکس یا جعبه نمایان‌گر یک کلاس است. بنابراین حاوی بخشی است که نام کلاس را می‌دهد، بخشی که مشخصه‌های اشیای تعریف شده توسط کلاس را فهرست می‌کند و بخشی که عملیات مربوط به چنین اشیایی را توصیف می‌کند.

هم‌چنین نسخه گرافیکی توصیه‌های به‌کار رفته در زبان برنامه‌نویسی در شکل ۲۲-۱۰ نشان داده شده‌اند. جعبه‌ای که گوشه سمت راست بالای آن تاخورده توجه خوانندگان را به جنبه‌ای از یک نمودار جلب می‌کند. در مورد شکل ۲۲-۱۰، توجه خواننده را به این حقیقت جلب می‌کند که بسیاری از مشخصه‌ها و عملیات مربوط به کلاس مشتریان نشان داده نشده‌اند، مثلاً مجموعه‌ای از حساب‌های مربوط به مشتری در صفات مشخصه این کلاس نیامده‌اند.

شکل ۲۲-۱۰ بسیار ساده است، در عمل نمودارهای کلاس به زبان UML ارتباط بین کلاس‌ها را نشان می‌دهند. چند نوع ارتباط مختلف وجود دارد که می‌توان آنها را بیان نمود. اولین مورد تعمیم[1] می‌باشد.



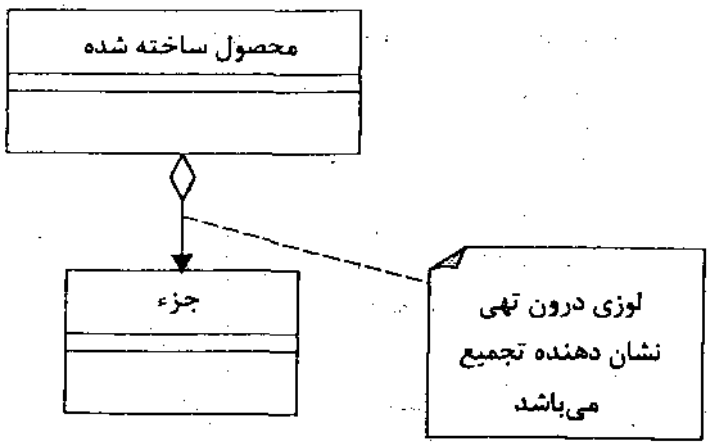
شکل ۲۲ـ۱۱ یک مثال از تعمیم در UML

شکل ۲۲ـ۱۱ یک مثال از تعمیم در UML

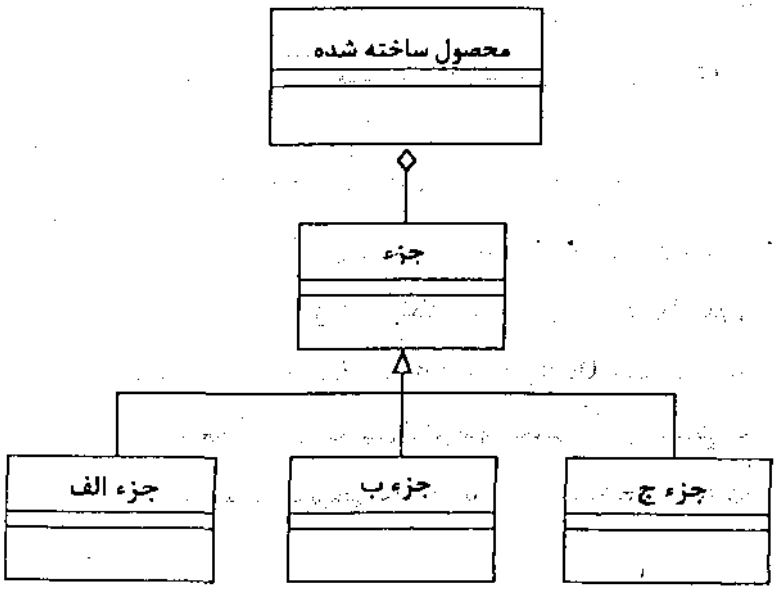
۲۲-۵-۲ مدل تعمیم (Generalization)

این ارتباط بین کلاس X و کلاس Y وقتی برقرار می‌شود که کلاس Y نمونه خاص‌تری از X باشد. مثلاً، یک نوع از این رابطه بین کلاس Account دارای ارتباط تعمیم با کلاس‌های خاص‌تر CurrentAccovnt و Deposit Accovnt است. این رابطه به‌وسیله یک خط فلش مانند از سوی کلاس‌های خاص‌تر به‌سوی کلاس‌های عمومی‌تر می‌رود. دوباره توجهتان را به این موضوع جلب می‌کنم که در مورد مقاصد تشریحی ما هیچ‌گونه عملیات یا مشخصه‌ای را نشان نداده‌ایم.(شکل ۲۲-۱۱)

1.Generalization



شکل ۲۲ـ۱۲ تجمیع در UML



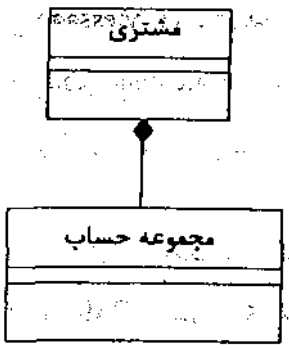
شکل ۲۲ـ۱۳ یک نمودار کلاس UML که تعمیم و تجمیع را با هم نشان می‌دهد

۲۲-۵-۳ تجمیع و ترکیب (Aggregation & Composition)

پویش
تجمیع چیست؟

بخش قبل رابطه‌ای را توصیف نمود که می‌توان از نمودار کلاس UML به نمایش درآورد یعنی تعمیم. دو رابطه مهم دیگر عبارتند از تجمیع و ترکیب. دو نوع رابطه وجود دارند که باعث می‌شوند یک کلاس اشیایی تولید کند که بخشی از شیء تعریف شده توسط کلاس دیگرند. مثلاً سیستمی برای تولید باید داده مربوط به اقلامی را که تولید شده‌اند و آن‌چه که این مواد از آنها ساخته شده‌اند را نگه دارد. مثلاً، یک کامپیوتر از یکسری قطعاتی تشکیل شده که شامل کابینت، دیسک سخت، مجموعه‌ای از کارت‌های حافظه و غیره می‌باشد. این کامپیوتر از تعدادی قطعه تشکیل شده و در سیستم شی‌گرایی قرار داده شده

که برای پشتیبانی تولیدی به‌کار گرفته می‌شود که در این‌جا رابطه تجمیع بین کلاس‌ای که برای تصویف محصول تولیدی استفاده شده و هر یک از اجزای آن، وجود دارد. در این‌جا می‌گوییم که رابطه تجمیع وجود دارد. شکل ۲۲-۱۲ نشان می‌دهد که چگونه این رابطه در نمودار کلاس UML نشان داده شده است.



شکل ۲۲ـ۱۴ یک نمودار کلاس UML که ترکیب را نشان می‌دهد.

در این‌جا خطی که در یک‌سر آن یک لوزی وجود دارد نشان‌دهنده این است که یک کلاس اشیایی را توصیف می‌کند که حاوی اشیای تعریف شده توسط کلاس دیگرند. در UML، روابط معمولاً ترکیب شده‌اند. مثلاً در شکل ۲۲-۱۲ چند قطعه وجود دارند که دارای یک رابطه تعمیمی با کلاس Component هستند. این مطلب در شکل ۲۲-۱۳ نشان داده شده که در آن Component با چند کلاس خاص‌تر مرتبط است که قطعاتی را توصیف می‌کنند که محصول کاری از آنها ساخته می‌شود.

پرسش
ترکیب چیست؟

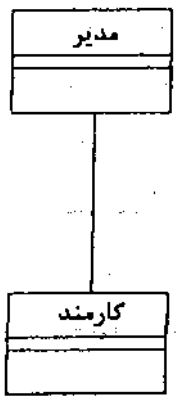
نوع خاصی از تجمیع وجود دارد که به نام ترکیب یا Composition شناخته می‌شود. این مورد وقتی استفاده می‌شود که شما موقعیتی دارید که در آن شیء دارای چند شیء دیگر است و وقتی شیء اصلی که آنها را در برمی‌گیرد حذف می‌شود همه موارد موجود در آن شیء نیز ناپدید می‌گردند. مثلاً، کلاس Customer که نمایان‌گر مشتریان یک بانک است دارای ارتباط ترکیبی است که با حساب‌هایی می‌باشد که مشتری در اختیار دارد زیرا اگر مشتری حذف گردد تمام حساب‌هایش نیز حذف می‌گردد. این فرم ارتباط از جهاتی شبیه تجمیع می‌باشد. در این‌جا لوزی به‌جای توخالی بودن، سیاه شده است که در شکل ۲۲-۱۴ نشان داده شده است.

۲۲-۵-۴ شرکت پذیری (Association)

تجمیع و ترکیب نمونه‌های خاصی از ارتباط بین دو کلاس‌اند. ارتباط وقتی بین دو کلاس رخ می‌دهد که ارتباطی بین کلاس‌ها وجود داشته باشد و در UML به آن شرکت‌پذیری می‌گویند. نمونه‌هایی از این مورد در زیر آمده‌اند:

- کلاس Manager یا مدیر به‌وسیله این حقیقت که مدیر بر تعدادی از کارمندان نظارت دارد، با کلاس Employee یا کارمند مرتبط است.
- کلاس Flight یا پرواز به‌خاطر این حقیقت که هواپیما یک پرواز خاص را انجام می‌دهد، با کلاس Plane یا هواپیما مرتبط است.
- کلاس Computer به‌خاطر این‌که مجموعه‌ای از پیام‌ها منتظر پردازش توسط کامپیوترند با کلاس Message یا پیام ارتباط پیدا می‌کند.
- کلاس BankStatement به‌خاطر این حقیقت که اعلامیه شامل جزئیات هر تراکنش است با کلاس تراکنش یا معامله مرتبط می‌شود.

از میان این ارتباطات تنها مورد آخری از نوع تجمیع است. بقیه ارتباطات ساده هستند. چنین ارتباطاتی در UML به‌عنوان یک خط مستقیم نوشته می‌شوند. مثلاً، شکل ۲۲-۱۵ اولین رابطه فوق را نشان می‌دهد.

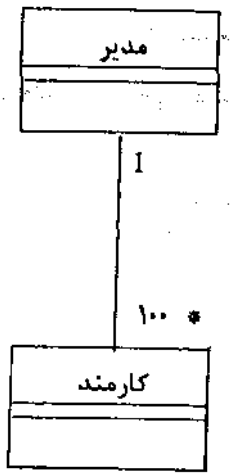


شکل ۲۲ـ۱۵ یک مثال از یک رابط ساده در UML

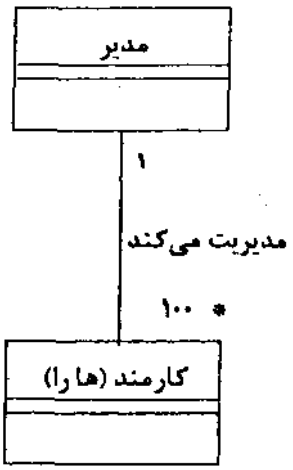
ارتباطات بین کلاس‌ها نیز از نظر تکثیر رابطه و نام ارتباط به ثبت رسیده‌اند. اجازه دهید ابتدا با بررسی مثالی در شکل ۲۲-۱۵، نگاهی به عمل تکثیر داشته باشیم. در این مثال یک مدیر به تنهایی یک یا چند کارمند را مدیریت نموده و هر یک کارمند تحت‌نظر یک مدیر می‌باشد. این رابطه در شکل ۲۲-۱۶ نشان داده شده است.

در این‌جا شماره ۱ که در انتهای بالای خط قرار دارد نشان‌دهنده این است که هر کارمندی تنها تحت‌نظر یک مدیر است. شماره ۱ ستاره‌دار در انتهای خط بیان می‌دارد که یک مدیر حداقل بر یک کارمند نظارت دارد.

ارتباطات بین کلاس‌ها می‌توانند دارای نام باشند تا مشخصاً این رابطه را به ثبت برسانند. مثلاً شکل ۲۲-۱۷ این حقیقت را می‌رساند که یک مدیر مجموعه‌ای از کارمندان را مدیریت می‌کند. جایگزین دیگری در مورد استناد این رابطه عبارتست از ثبت نقش‌هایی که هر یک از کلاس‌ها در یک رابطه ایفا می‌کنند. نمونه‌ای از این مثال در شکل ۲۲-۱۸ نشان داده شده است. در این‌جا کلاس University دارای نقش میزبانی یک‌سری دانشجویانی است که در عوض آنها نیز نقش اعضای دانشجو دانشگاه را ایفا می‌کنند. معمولاً وقتی ارتباطاتی را که انتخاب کرده‌اید ثبت می‌کنید که اسناد مورد استفاده را تشکیل می‌دهند: از ثبت ارتباطات یا ثبت نقش کلاس‌هایی که در این رابطه شرکت دارند. انجام هر دو کار، در عین حالی‌که کاملاً معتبر هم باشد، به‌عنوان Overkill در نظر گرفته می‌شود.



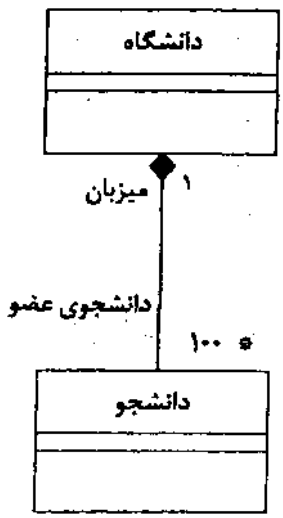
شکل ۲۲ـ۱۶ چند گانگی در یک نمودار کلاس UML



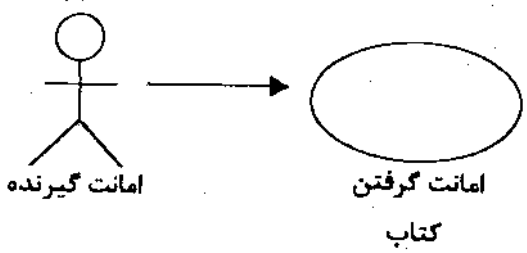
شکل ۲۲ـ۱۷مستند سازی یک همکاری

۵-۵-۲۲ Use-Case ها

تاکنون در فصل ۲۱ در مورد موارد کاربرد صحبت کرده‌ایم. در UML یک مورد کاربرد بصورت بسیار ساده از نظر بازیگران و یک مورد کاربرد ثبت می‌شود. بازیگر عاملی است که با سیستم ساخته شده در ارتباط متقابل است. مثلاً خلبان هواپیما، گیرنده کتاب از کتابخانه یا مدیر چند کارمند در شرکت. مورد کاربرد اعمالی را ثبت می‌کند که بازیگر انجام می‌دهد. مثلاً قرض کردن کتاب، تغییر جهت هواپیما یا افزودن عضوی به تیم برنامه‌نویسی. یک مورد استفاده ساده در شکل ۲۲-۱۹ نشان داده شده است. این شکل کاربر یک کتابخانه را نشان می‌دهد که کتاب قرض می‌کند.

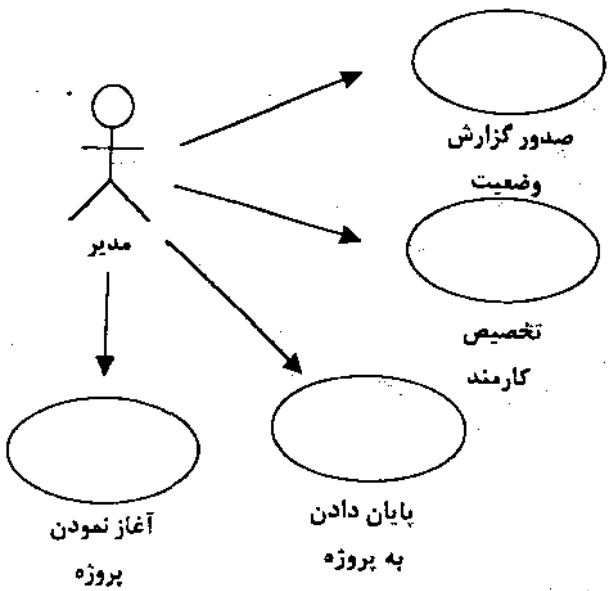


شکل ۲۲ـ۱۸ مستند سازی نقش‌ها



شکل ۲۲ـ۱۹یکUseCaseساده

بازیگر در این مورد قرض گیرنده‌ای است که از کتابخانه استفاده کرده و شکل بیضی مورد کاربرد با نام آن که در زیر آمده را نشان می‌دهد. موارد کاربرد نمایان‌گر دیدگاه سطح بالایی در عملیات یک سیستم به زبان UML است. به‌طور کلی، ممکن است یک سیستم بزرگ شامل صدها اگر نگوییم هزارها، مورد کاربرد باشد. بخشی از چندمورد کاربردی که همگی دارای یک بازیگر هستند در شکل ۲۲-۲۰ آمده است.

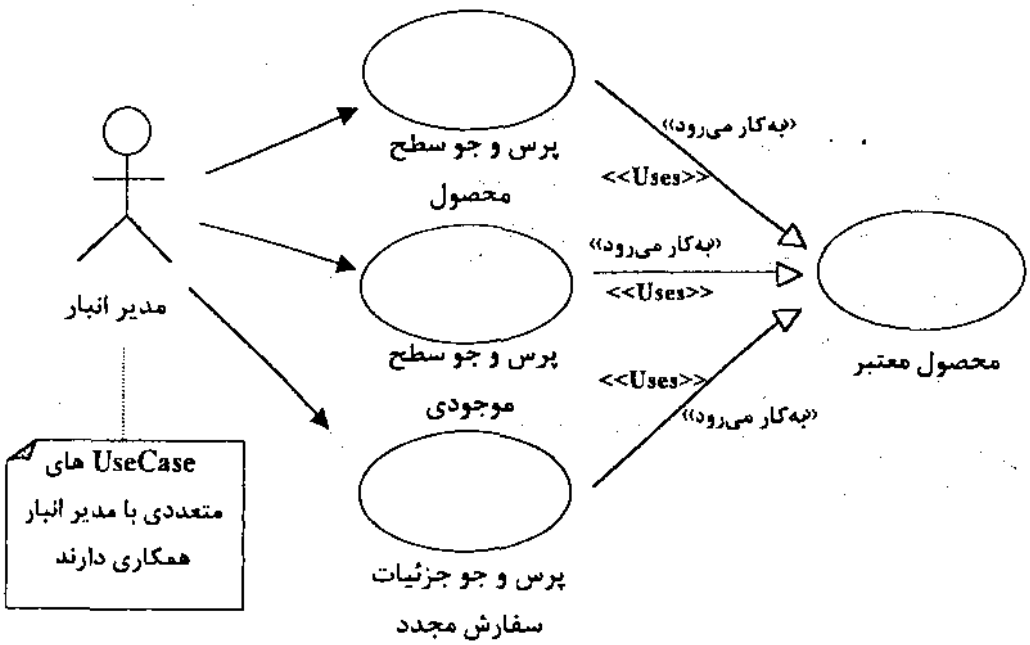


شکل ۲۲ـ۲۰ تعدادیUseCase

این شکل بعضی از اعمالی را که یک مدیر پروژه ممکن است در هنگام ارتباط متقابل با سیستم مدیریت پروژه انجام دهد، نشان می‌دهد.

وجود موارد کاربرد متعدد به این معنی است که موارد کاربردی وجود دارد که توسط موارد کاربرد دیگر مورد استفاده قرار می‌گیرد. وقتی این اتفاق رخ می‌دهد، نمودار مورد کاربرد UML شامل عنوان شناخته شده‌ای به نام <<Uses>> می‌باشد که روی پیکانی که منتهی به مورد کاربرد است نشان داده شده است. نمونه‌ای از آن در شکل ۲۲-۲۱ آمده است که بعضی از موارد کاربرد درگیر در سیستمی برای اجرای نظارت بر محصولاتی در انبار را نشان می‌دهد. مدیر انبار که به‌عنوان بازیگر در شکل ۲۲-۲۱ نشان داده شده به‌وسیله یک‌سری موارد کاربردی که به اطلاعات محصولات ذخیره شده در انبار دسترسی دارند، مرتبط است. مثلاً مدیر انبار می‌تواند میزان موجودی محصول به‌خصوصی را مورد تحقیق قرار دهد. در انجام این عملیات مدیر انبار چند مورد کاربرد ایجاد می‌کند که هر کدامشان از مورد کاربردی استفاده می‌کنند که نام محصولی را که به آن ارجاع می‌کند ارزیابی می‌نماید، مثلاً مدیر نام محصول معتبری را تایپ نموده است.

در این‌جا این حقیقت که مورد کاربردی توسط دیگر موارد کاربرد مورد استفاده قرار می‌گیرد توسط یک پیکان یا فلش با نوک توخالی نشان داده می‌شود.

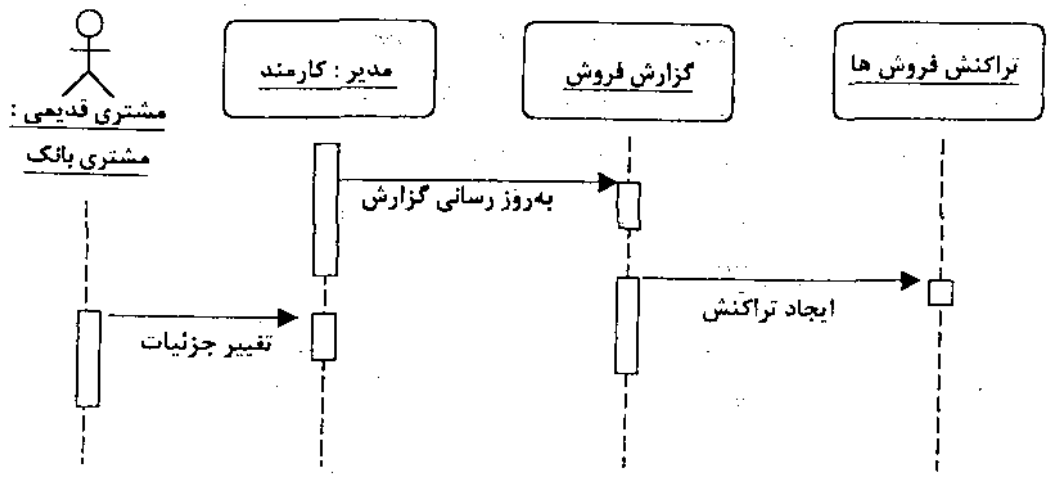


شکل ۲۲ـ۲۱ مثالی از یک UseCase که با دیگر UseCaseها کار می‌کند

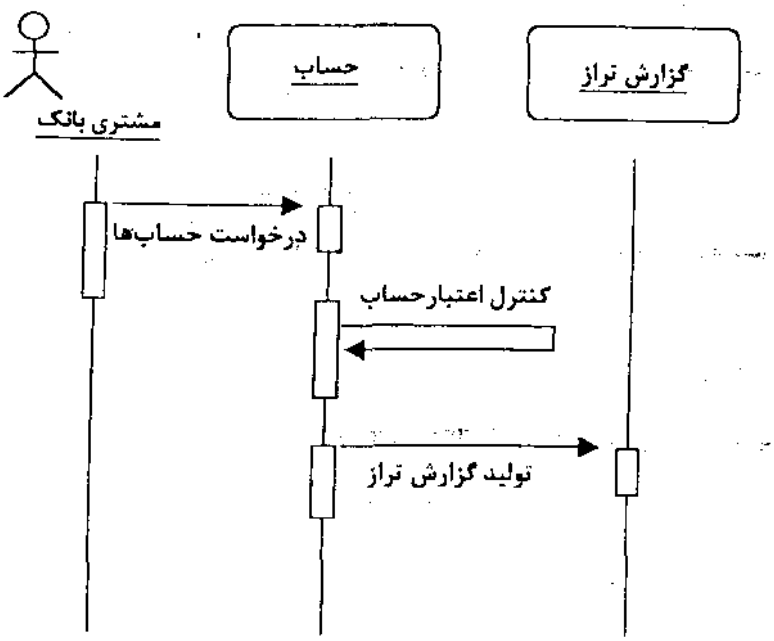
۲۲-۵-۶ مشارکت ها (Collaboration)

در طول اجرای سیستم شیءگرا، اشیا با یکدیگر رابطه متقابل دارند. مثلاً در سیستم بانکداری شیء Account ممکن است پیامی به شیء تراکنش بفرستد تا تراکنشی ایجاد نماید که روی آن حساب صورت گرفته مثلاً حساب دارای بدهی شده است. این نوع اطلاعات برای طراح سیستم شیءگرا در طول فرآیند

شناسایی و ارزیابی کلاس‌ها، مهم است. به‌خاطر این امر، UML دارای دو روش یکسان برای تعریف روابط متقابل است. در این کتاب تنها توجه خود را معطوف به یکی از آنها می‌کنیم: نمودار توالی. نمودار دیگر به‌عنوان نمودار همکاری شناخته شده و هم‌ارز نمودار توالی است. در واقع این دو آن‌قدر به هم شباهت دارند که ابزارهای CASE اغلب می‌توانند یک نمودار را از روی نمودار دیگری ایجاد کنند. شکل ۲۲-۲۲ مثال ساده‌ای از یک نمودار توالی را نشان می‌دهد.



شکل ۲۲ـ۲۲ یک نمودار ساده از ترتب (دنباله)



شکل ۲۲ـ۲۳ مثالی دیگر از یک نمودار دنباله (توالی)

در این نمودار، سه شیء وجود دارد که در یک رابطه متقابل و درونی درگیرند. اولی شیء manager (مدیر) است که توسط کلاس Employee (کارمند) توصیف شده است. این شیء یک پیام updateReport (به روزسازی گزارش) به شیء به نام salesReport می‌فرستد که سپس پیام‌های createTransaction (ایجاد تراکنش) را به شی‌های بیشتری از salesTransaction ارسال می‌کنند. در این نمودار توالی سه شیء وجود دارد یکی از آنها یعنی manager کلاس خود را یعنی Employee مشخص نموده که دیگران این کار را نکرده‌اند. محتویات یک جعبه در نمودار توالی می‌تواند شامل تنها نام شیء، نام شیء همراه با نام کلاس‌اش که توسط علامت (:) جدا شده یا تنها نام کلاس باشد که قبل از آن علامت (:) آمده است. در مورد آخری شیء مبهم است. شکل ۲۲-۲۲ نقش بازیگر را در همکاری نشان می‌دهد: در اینجا بازیگر BankCostomer (مشتری بانک) oldcostomer (مشتری قدیمی) توسط ارسال پیام *changeDetails* (تغییر جزئیات) با مدیر شیء *Employee* ارتباط برقرار می‌کند.

شکل ۲۲-۲۳ مثال بیشتری از نمودار توالی را نشان می‌دهد.

در اینجا یک بازیگر که توسط شیء مبهم و بی‌نامی نمایش داده شده و توسط کلاس BankCostomer تعریف شده پیامی به شیء account می‌فرستد که درخواست حساب را می‌کند. این شیء بررسی می‌کند که آیا این یک شماره حساب معتبر است یا خیر و سپس پیام generateBalanceReport (ارائه گزارش تراز) را به شیء balance Reports می‌فرستد که حاوی اطلاعاتی است که مشتری بانک تقاضا کرده بود.

۲۲-۵-۷ نمودارهای وضعیت

پرسش

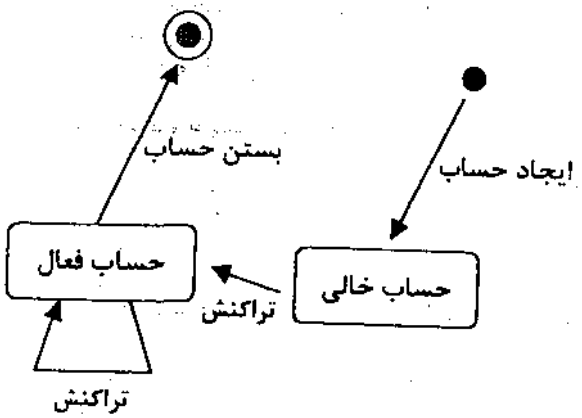
نمودار وضعیت چیست؟

جزء مهم دیگری از UML، نمودار وضعیت است. این نمودار حالات مختلفی را نشان می‌دهد که شیء خودش پیدا می‌کند و چگونگی انتقال هر حالت به حالات دیگر را نیز نشان می‌دهد. چنین نموداری شامل چند جزء است:

- حالات[1] که به‌عنوان جعبه با گوشه‌های گرد نشان داده می‌شوند.
- رویدادها[2] بین حالات که با خطوط فلش‌دار نشان داده می‌شوند.
- تراکنش[3] که باعث رویدادها بین حالات می‌شوند.
- یک startmarker یا علامت شروع که حالت یا وضعیت اولیه که شیء در هنگام ایجاد آن، پیدا می‌کند را نشان می‌دهد.

1.States
2.Transitions
3.Events

- علامت توقف یا stopmarker که نشان‌گر این است که شیء به انتهای عمر خودش رسیده است.



شکل ۲۲ـ۲۴ یک مثال از یک نمودار وضعیت

نموداری از وضعیت به‌عنوان نمونه در شکل ۲۲-۲۴ آمده است.

در این‌جا چرخه حیات حساب بانکی نشان داده شده است. وقتی حساب ایجاد شد به‌عنوان یک حساب خالی در نظر گرفته می‌شود. بلافاصله تراکنش روی حساب صورت می‌گیرد (وجوه وارد شده یا خارج شده از حساب باعث فعال شدن حساب می‌شود). نمودار وضعیت هم‌چنین نشان می‌دهد که چه وقت حساب بسته شده، از بین می‌رود.

۲۲-۶ بررسی موردی ـ کتاب‌های روی خط(شبکه) On-Line Books

هدف از این بخش تشریح استفاده از نمودارهای UML است که در بخش ۲۲-۵ توصیف شده و در سیستم واقعی کامپیوتری به‌کار گرفته شده‌اند. این سیستم یک سیستم Ecommerce یا تجارت الکترونیکی است.

۲۲-۶-۱ کتابهای روی خط

آن‌لاین بوکز شرکت جدیدالتأسیسی است که شعبه یک شرکت تجارت مولتی مدیای بزرگ به نام پول‌دی پابلیشینگ است. مدیران پول‌دی پابلیشینگ متوجه رشد عظیمی در اینترنت بر اساس خرده فروشی از میان کامپیوترهایشان شدند و تصمیم گرفتند برای پاسخ به این نیاز آن‌لاین بوکز را تأسیس کنند. مفهومی که پول دی پابلیشینگ داشت یک وب سایت تجارت الکترونیکی بود که حاوی جزئیات ارائه شده در هر کتاب همراه با تسهیلاتی بود که به‌وسیله آن کاربر یک وب سایت می‌توانست کتاب‌هایی را با

استفاده از فرم قرار داده درون صفحه وب، سفارش دهد. بعضی موارد از مجموعه اولیهٔ نیازمندیهای تعریف شده به‌صورت دقیق توسط کارکنان فنی که به آن‌لاین بوکز متصل شده در زیر نشان داده شده است:

۱- آن‌لاین بوکز مایل است یک مرکز فروش آن‌لاین از طریق وب ایجاد کند. آن سایت وب که این توانایی را ایجاد می‌کند باید مشتری را قادر سازد جزئیات مربوط به کتاب را جستجو کرده، کتابی را سفارش داده و نام خود را به‌عنوان گیرنده نامه الکترونیکی ثبت کند که جزئیات پیشنهادات ویژه، انتشارات جدید و بازبینی‌ها را با خود دارد.

۲- وقتی مشتری وارد سایت وب می‌شود، هر یک از تسهیلات ارائه شده در پاراگراف فوق نمایش داده می‌شوند.

۳- اگر مشتری به‌عنوان گیرنده نامه الکترونیکی ثبت شود از او خواسته می‌شود تا جزئیات آنها را به‌اطلاع برساند. این جزئیات شامل نام مشتری، آدرس پست الکترونیکی و آدرس پستی است. عضوی از کارکنان به نام مدیر ارتباطات مسئول ارسال این نامه‌های الکترونیکی با اطلاعات خواسته شده از طرف مشتری هستند.

۴- مشتری کتابها را از روی بروشور آن‌لاین با جستجو در صفحات وب همراه با جزئیات کتاب و انتخاب کتابی به‌وسیله استفاده از مکانیزمی مثل کیلیک کردن روی آن کتاب، می‌خرد. این سیستم یک مرکز خرید مجازی دارد که جزئیات هر کتاب در آن ذخیره شده‌اند. وقتی سبد خرید با کتابها پر شد، قیمت جمع زده شده همه خریدهای صورت گرفته به مشتری نشان داده می‌شود.

۵- وقتی مشتری خرید خود را روی شبکه وب تمام کرد، از او تقاضا اطلاعاتی در مورد نحوه پست مورد استفاده می‌شود. در حال حاضر، ما پست استاندارد را متصور می‌شویم و تعیین می‌کند که ظرف ۲۴ ساعت تحویل داده می‌شود.

۶- سایت وب باید با سیستم کنترل انبار که آن هم نیاز به توسعه دارد، در ارتباط متقابل باشد. این سیستم کنترل انبار باید از عهده دریافت کتابها از انبار ناشران، تحویل کتابها بعد از سفارش توسط مشتری و سفارش مجدد کتابهایی برآید که انبار آنها رو به کاهش گذارده، ما مشکل تهیه کتاب برای انبار را در یک دوره هفت روزه پیش‌بینی می‌کنیم.

۷- مدیر کنترل انبار طی یک دوره هفت هفته‌ای انتخاب می‌شود. او مسئولیت مراقبت از فروش و دسترسی به انبار را برعهده دارد و اینکه، وقتی موجودی یک کتاب رو به کاهش گذاشت آن را مجدداً سفارش دهند. برای انجام این کار، سیستم کنترل انبار باید گزارشات مرتبی را در مورد فروش و کاهش موجودی ارائه دهد.

۸- مدیر بازاریابی برای دوره‌ای هشت هفته‌ای انتخاب می‌شود. مدیر بازاریابی کار تعیین قیمت فروش کتابها را به‌عهده دارد. تصور می‌کنیم که یک کتاب در طول اولین دوره زندگی دارای

قیمت‌های مختلف فروش است. مثلاً ممکن است تصمیم بگیریم که در چند هفته اول آن را با تخفیف زیادی ارائه کنیم و سپس قیمت را به قیمت‌های پیشنهادی ناشر نزدیک‌تر کنیم.

شرکتی که نرم‌افزار برای شرکت آن‌لاین بوکز تولید کرده ابتدا چند کلاس‌بندی پیشنهادی را شناسایی کرد. جزئیات آن در زیر آمده است:

- پرونده مشتری[1] ـ جزئیات مربوط به کسی که کتاب می‌خرد یا برای دریافت نام الکترونیکی که حاوی اطلاعات بازاریابی است، ثبت‌نام کرده است.
- کتاب ـ قلم اصلی که آن‌لاین بوکز می‌فروشد.
- سفارش[2] ـ سفارشی که مشتری برای یک یا چند کتاب می‌دهد. ممکن است این سفارش برای یک نسخه از یک کتاب یا یک نسخه از هر کتاب یا چندین نسخه از کتابها باشد. سفارش شامل چند خط برای سفارش و مشخصات پستی است.
- محل نوشتن سفارش[3] ـ یک خط برای یک سفارش. مثلاً سفارش یک نسخه از یک کتاب. سفارش شامل یک یا چند خط مربوط به سفارش است. خط سفارش حاوی اطلاعاتی در مورد نوع کتاب مورد سفارش و تعداد سفارش داده شده است (که این معمولاً ۱ است).
- سبد خرید[4] ـ اصطلاحاً سبدی است برای خرید که در طول گردش در سایت وب همراه مشتری است تا سفارش دهد. محتویات سبد، خطوط سفارشند. وقتی مشتری فرم سفارش را درون سبد را تکمیل می‌کند که خطوط سفارش درون سبد خرید و اطلاعات پستی تهیه شده توسط مشتری به سفارش افزوده شده باشند.
- BackOrder ـ این بخشی از سفارش است که در حال حاضر توسط موجودی انبار شرکت ارائه نمی‌شود. اگر مشتری از منتظر ماندن برای یک کتاب راضی باشد، برای او یک Back Order صادر می‌شود. این Back Order وقتی برقرار می‌شود که موجودی کتابهای سفارشی توسط آن‌لاین بوکز دریافت می‌شود.
- انبار[5] ـ مجموعه‌ای از کتابهایی است که در حال حاضر ذخیره شده‌اند. سفارش یک کتاب یا مجموعه‌ای از آنها به انبار ارسال می‌شود و کارکنان آن‌جا کتابها را از قفسه‌های آن در انبار خارج کرده بسته‌بندی نموده و برای مشتری ارسال می‌کنند. جزئیات موجودی تنظیم می‌شوند.

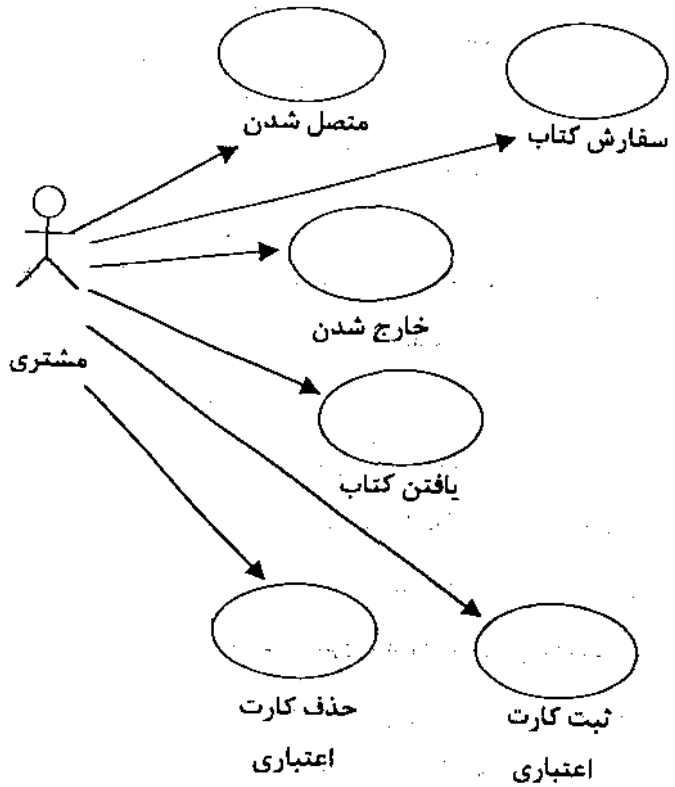
1.Customer Record
2.Order
3.Order Line
4.Shopping Cart
5.Warehouse

- پرونده موجودی[1] ـ اطلاعاتی است که جزئیات موجودی کتابی را تشریح می‌کند. مثلاً چند تا از این کتاب موجود است، سطح کنونی آن برای سفارش مجدد به ناشر چه میزان بوده و محل کتابها در داخل انبار کجاست.
- (اعلامیه بسته‌بندی)[2] ـ این یادداشتی است که با مجموعه‌ای از کتابها برای مشتری ارسال می‌شود. اعلامیه بسته‌بندی حاوی اطلاعاتی در مورد کتابهای ارسالی و نرخ پست به‌کار گرفته است. هم‌چنین شامل جزئیات هر کتابی است که به‌خاطر موجود نبودن در انبار، ارسال نشده است.
- کارت اعتباری[3] ـ مشتری با استفاده از کارت اعتباری پول کتابها را می‌دهد. این سیستم به مشتری اجازه می‌دهد کارت خود را از قبل ثبت کند. به‌طوری‌که آنها مجبور نیستند با هر سفارش جدید آنها را مجدداً تایپ کنند.

اینها کلاس‌های اصلی شناسایی شده‌اند. در این‌جا چند نفر به نام بازیگر نیز هستند که باید شناسایی شوند:

- مشتری[4] ـ این قرد بازیگر اصلی است: شخصی که کارهایی را انجام می‌دهد که منجر به تغییرات عمده‌ای در وضعیت سیستم می‌شود.
- مدیر بازاریابی[5] ـ بازیگر مهمی که بسیاری از پارامترهای سیستم مثل تنظیم قیمت کتاب را تنظیم می‌کند.
- مدیر کنترل انبار[6] ـ کسی که موجودی انبار را کنترل نموده و در مورد سفارش موجودی تصمیم‌گیری می‌کند.

1.Stock Record
2 . Packing Note
3.Credit Card
4.Customer
5.Marketing Manager
6.Stock Control Manager



شکل ۲۲ـ۲۵ برخی UseCaseها برای بازیگر مشتری

تعداد بسیار زیادی موارد کاربرد مربوط به این بازیگران وجود دارد که بسیاری از آنها که مربوط به مشتری هستند در شکل ۲۲-۲۵ نشان داده شده‌اند.

انتخاب موارد کاربرد مربوط به مشتری و آن‌چه که در شکل ۲۲-۲۵ آمده است شامل موارد کاربردی برای:

- اتصال به سیستم[1] ـ در این‌جا مشتری نام و کلمه رمز خود را ارائه می‌دهد. وقتی وارد سیستم شد می‌تواند کاتالوگ کتاب‌ها را مورد بازدید قرار دهد.
- سفارش[2] ـ در این‌جا مشتری یک یا چند نسخه از یک کتاب را سفارش می‌دهد.
- پایان کار[3] ـ مشتری سفارش خود را تکمیل کرده و به سیستم دستور می‌دهد از محل ارسال سفارش کار پردازش را آغاز کند.
- پیدا کردن یک کتاب[4] ـ مشتری کاتالوگ آن‌لاین را در مورد کتاب خاصی جستجو می‌کند.

1.Logging in
2.Ordering
3.Checking out
4.Finding a Book

- حذف یک کارت اعتباری[1] ـ در این‌جا مشتری یکی از کارت‌های اعتباری مربوط به خود را حذف می‌کند.
- ثبت کارت اعتباری[2] ـ در این‌جا مشتری یکی از کارت‌های اعتباری خود را در سیستم ثبت می‌کند.

بخشی از یک‌سری نمودارهای سیستم در شکل ۲۲-۲۶ نشان داده شده‌اند. یک‌سری نقش‌های مربوط به این نمودار حذف شده‌اند، معمولاً آنها نیز گنجانده می‌شوند. بعضی از ارتباط‌های بین کلاس‌ها حذف شده است.

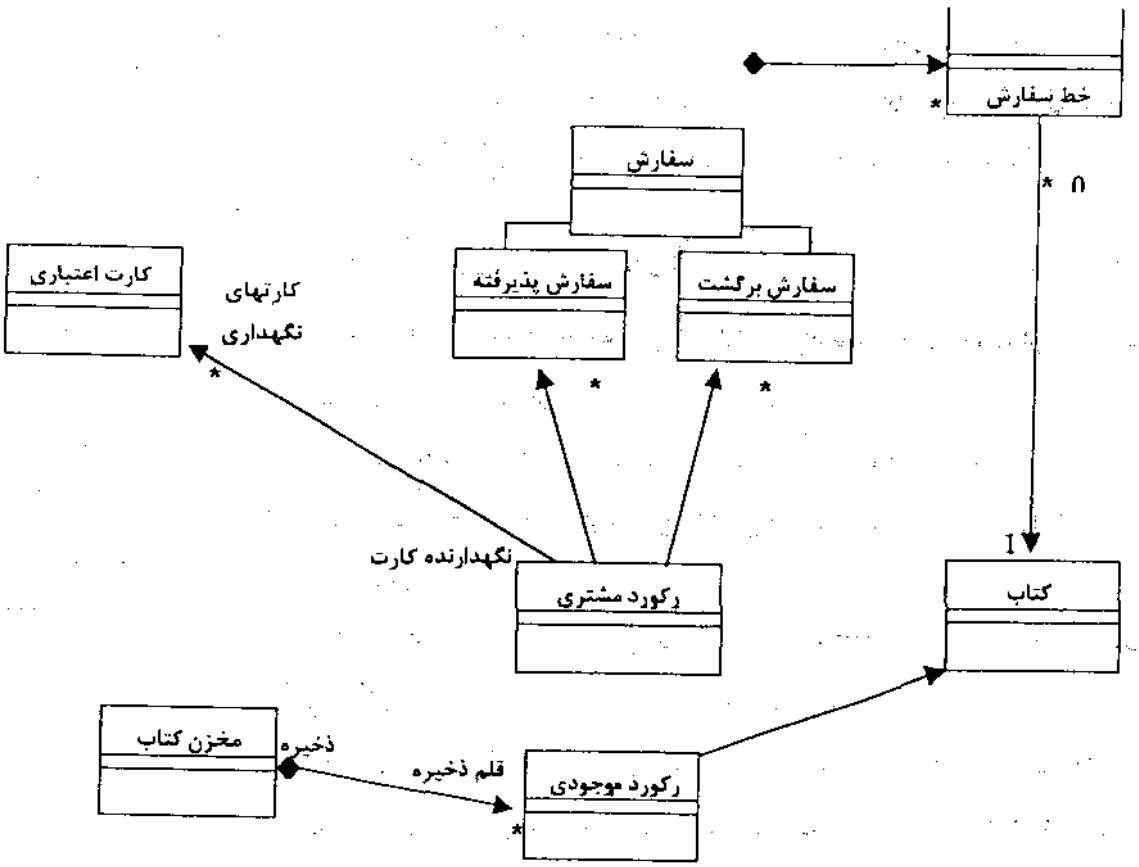
نمودار، بسیاری از کلاس‌هایی را نشان می‌دهد که قبلاً توصیف شده‌اند. تنها کلاس‌هایی که توصیف نشده‌اندSatisfied Order و BackOrder هستند. این دو کلاس نوع خاصی از کلاس Order هستند که نمایان‌گر سفارش کتاب‌ها از سوی مشتری است.

وقتی از سوی مشتری سفارشی داده شد، بعضی از اقلام سفارشی پیدا نمی‌شوند زیرا ممکن است این کتاب‌ها موجود نباشند. وقتی این امر رخ می‌دهد سفارش دو قسمت می‌شود: تمام کتاب‌هایی که موجودند در شیء SatisfieldOrder قرار می‌گیرند و کتاب‌هایی که موجود نمی‌باشند در شیء BackOrder قرار دارند. ارتباطات این نمودار کلاس در زیر به تفسیر آمده است:

- انبار با تعدادی پرونده موجودی مرتبط است که جزئیات کتاب‌های ذخیره شده در انبار را دارند. یک انبار با یک یا چند پرونده موجودی مرتبط است.
- یک پرونده موجودی با یک کتاب و یک کتاب با یک پرونده موجودی ارتباط دارد.
- یک کتاب را می‌توان در چند خط سفارش یافت، اما تنها یک کتاب در هر خط مورد رجوع قرار دارد.
- یک سفارش شامل چند خط سفارش و یک خط با یک سفارش مرتبط است.
- مشتری یک‌سری کارت‌های اعتباری را در سیستم ثبت می‌کند. یک کارت اعتباری با یک مشتری مرتبط است.
- هر مشتری با یک‌سری سفارش انجام گرفته در طول یک دوره زمانی، ارتباط دارد. هر سفارش مؤفق با یک مشتری رابطه دارد.
- هر مشتری با یک‌سری کتاب‌هایی که در حال حاضر موجود نیستند مرتبط است و هر کتاب عدم سفارش داده شده با یک مشتری مرتبط می‌باشد.

1.Deleting a Credit Card

2.Registering a Credit card



شکل ۲۲ـ۲۶ خلاصه ای از یک نمودار کلاس برای مطالعه موردی

شکل ۲۲-۲۶ تنها بعضی از ارتباطات مورد بحث را نشان می‌دهد، مثلاً ارتباطی بین کارت‌های اعتباری و سفارشات به‌خاطر این حقیقت وجود دارد که یک کارت اعتباری به‌خصوص برای پرداخت سفارش استفاده می‌شود. در هر حال، جزئیات کافی نشان داده شده تا اشاره‌ای باشد برای این‌که یک نمودار کلاس‌بندی پیچیده UML چه شکلی است.

مثالی از نمودار توالی مربوط به بررسی موردی در شکل ۲۲-۲۷ نشان داده شده است:

در این‌جا مشتری کتابی را سفارش می‌دهد. این کار باعث جستجو در موجودی انبار و هماهنگی لازم در صورت موجود بودن می‌شود. اگر کتاب وجود داشته باشد شیء خط سفارش ایجاد می‌گردد که سپس به سفارشی افزوده می‌شود که بر اثر تقاضای مشتری در سایت وب آن‌لاین بوکز، ایجاد شده است. (شکل ۲۲-۲۸) نمودار نهایی، نمودار وضعیت را در مورد شیء Order نشان می‌دهد.

مشتری سفارشی داده و وضعیت شیء Order به‌صورت یک سفارشی نسبی درمی‌آید سپس مشتری حق دارد کتاب‌های بیشتری را در صورت درخواست اضافه نموده یا کتابی را از سفارش حذف نماید. در هر مرحله‌ای در طول انجام سفارش مشتری می‌تواند سفارش خود را لغو کند که این کار منجر به خاتمه عملیات می‌شود. وقتی مشتری به پایان کار اشاره دارد سفارش تبدیل به یک سفارش تکمیل شده می‌شود. در این برهه مشتری دو راه پیش‌رو دارد: سفارش را کنسل نموده یا نوع ارسال را برای انجام سفارش،

معرفی کند. اگر نوع پست انتخاب شود کار انجام سفارش تکمیل می‌شود. در این مرحله مشتری دو راه دیگر دارد: سفارش را تأیید کند که در آن‌صورت سفارش تکمیلی برای پردازش ارسال می‌شود یا آن را لغو کند. هر دوی این گزینه‌ها منجر به نقطه خروجی در روی نمودار می‌شوند.

۷-۲۲ برنامه سازی شیٔ گرا

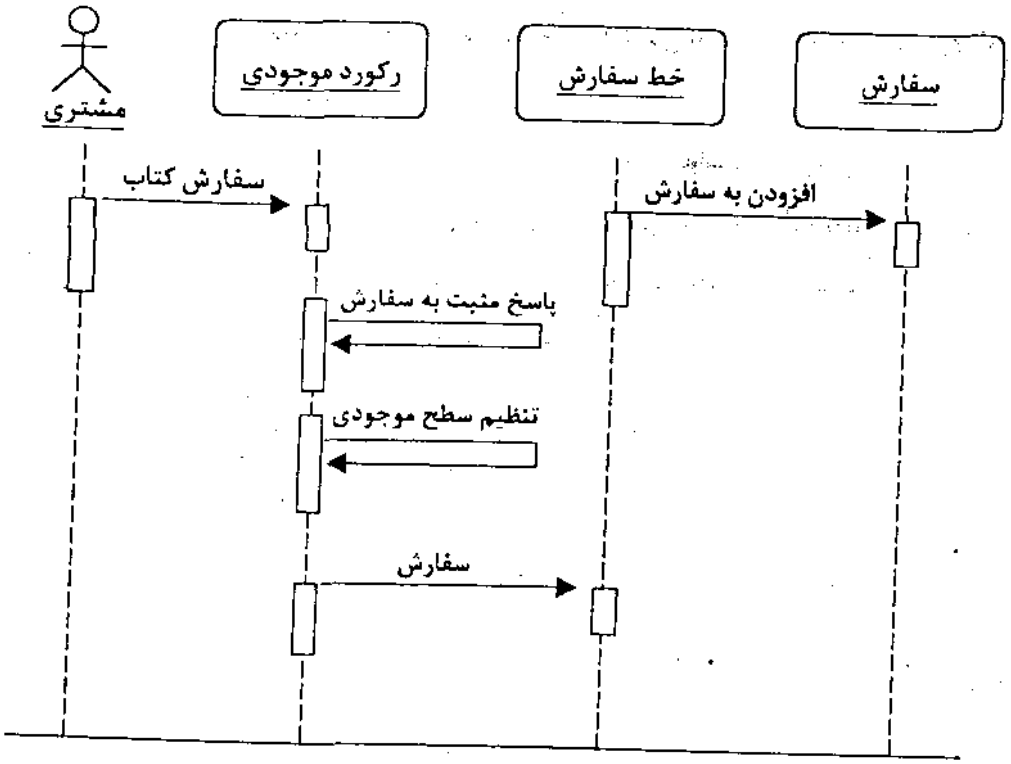
مرحله نهایی توسعه در چرخه عمر شیءگرایی، برنامه‌سازی است. هدف این کتاب وارد شدن به جزئیات مربوط به این کار نیست، زیرا با این‌که برنامه‌نویسی بسیار مهم است. اما به‌عنوان یک فعالیت جانبی در کنار تحلیل و طراحی در نظر گرفته می‌شود. اما یک معرفی‌نامه کوتاه در زبان برنامه‌نویسی جاوا آمده است. بخش «خواندنیهای دیگر و منابع اطلاعاتی» در پایان این فصل تعدادی از کتابهای خوب را در این زمینه معرفی می‌کند.

فرآیند برنامه‌نویسی مستلزم تبدیل طراحی شیءگرا به کُد برنامه است. این کار بدین‌معنی است که کلاس‌های تعریف شده در طراحی باید به کلاس‌هایی تعریف شوند که در یک زبان برنامه‌نویسی شیءگرا مثل Java, C++ یا Smalltalk بیان شده‌اند. در این بخش روی زبان جاوا متمرکز می‌شویم که به سرعت به زبان برنامه‌نویسی مؤثر موجود برای توسعه سیستم‌های توزیع شده توسط مودم تبدیل شده است.

هر کلاس در زبان جاوا توسط یک کلمه کلیدی معرفی می‌شود. داخل کُد کلاس، برنامه‌نویس، صفات خاصه و عملیات آن‌کلاس را مشخص می‌سازد. نمونه‌ای از کُد ساختاربندی کلاس در زیر نشان داده شده است:

```
class Customer
{
private String customerName;
private String customerAddress;
// More attributes defined here
    public String getCustomerName()
    {
    // Code for getCustomerName
    }
    public void modifyCustomerAddress(String address)
    {
    // Code for getCustomerName
    }
// Code for the reminder of the Operations
}
```

اولین خط برنامه نام کلاس‌ای به نام Customer را مشخص می‌سازد. بلافاصله به دنبال آن توصیف صفات خاصه کلاس شروع می‌شود. در برنامه فوق ما تنها دو صفت خاصه را نشان داده‌ایم: نام مشتری و آدرس آنها. هر دوی آنها به‌صورت رشته بیان شده‌اند. معمولاً در سیستم واقعی صفات خاصه بسیار بیشتری مربوط به چنین کلاس‌هایی وجود دارد. توصیف صفت خاصه شامل نوع (رشته) و رؤیت آنهاست.



شکل ۲۲ـ۲۷ یک نمودار ترتیبی (توالی - دنباله) برای مطالعه موردی

در مثال فوق دو صفت خاصه نشان داده شده دارای قابلیت رویت خصوصی هستند. یعنی از طریق هر برنامه‌ای در داخل کلاس قابل دسترسی هستند اما از برنامه‌ای خارج کلاس این امر میسر نیست، مثلاً برنامه‌ای که متعلق به کلاس دیگری می‌باشد. یعنی متغیرهای یک حالت در یک کلاس احاطه شده‌اند. تعیین‌کننده‌های قابلیت رؤیت دیگری نیز در جاوا وجود دارند. یکی از آنها را بعداً وقتی عملیات درون یک کلاس را شرح می‌دهیم خواهیم دید.

در کلاس Customer دو عملیات را نشان داده‌ایم. اولی عملیات getcustomername است که به نام مشتری توصیف شده در کلاس برمی‌گردد.

کلمه کلیدی String مشخص می‌کند که این عملیات یک رشته را برمی‌گرداند و کلمه رمز public این حقیقت را بازگو می‌کند که هر برنامه‌ای از کلاس دیگر می‌تواند از این عملیات استفاده کند. کلمه

public مخالف کلمه private است که به تفصیل در پاراگراف قبل گفته شد. برنامه این عملیات در براکت‌های شکسته عملیات قرار دارد.

عملیات modifyCustomeraddress از دو جهت با عملیات getcustomerName فرق دارد. اول این‌که، یک حرف زمز پوچ در مقدمه آن آورده شده که نشان‌گر این است که از عملیات نتیجه‌ای حاصل نمی‌شود: عملکرد آن تنها کارهایی را به انجام می‌رساند که صفات خاصه کلاس را تغییر می‌دهند. دوم این‌که، عملیات مربوط به یک آرگومان آدرس، رشته‌ای است نمایان‌گر آدرس جدید مشتری که باید جایگزین قبلی شود.

این فرم ابتدای یک کلاس در زبان جاواست. این فرم از بسیاری جهات مشابه ساختار کلاس‌هایی است که به سایر زبان‌های شی‌گرا بیان شده‌اند. به‌جز Smalltalk. برنامه کامل یک کلاس بسیار ساده در زیر آمده است. این کلاس نمایان‌گر یک شمارنده بازدید است یعنی دستگاهی که تعداد دفعات بازدید یک کاربر را از آن سایت وب نشان می‌دهد.

```
class Hitcounter
{
private int hits;
   public Hitcounter(int startValue)
    {
    hits = startValue;
    }
   public void setCounter(int value)
    {
    hits = value;
    }
   public int getCounter()
    {
    return hits;
    }
   public void incrementCounter()
    {
    hits++;
    }
}
```

این کلاس HitCounter نامیده می‌شود. یک مشخصه به نام hits دارد که تعداد دفعات بازدید کاربر را نشان می‌دهد. عملیات بعدی که دارای همان نام کلاس است به‌عنوان سازنده یا Constructor نامیده می‌شود. این کلاس‌ای از برنامه قابل اجراست که وقتی اجرا می‌شود که شیء ایجاد شده باشد. این

کلاس یک آرگومان تک رقمی دارد که مقدار اولیه این شماره است. وقتی کاربر این کلاس می‌خواهد شیء HitCounters را ایجاد کند، برنامه‌ای که این کار را می‌کند عبارتست از:

```
HitCounterhic = newHitCounter(0);
```

کلمه کلیدی new سازنده را فرا می‌خواند تا شیء HitCounter را ایجاد کند که دارای مقدار اولیه برابر با صفر است.

عملیات getCounter با مقدار کنونی صفت خاصه hits بازمی‌گردد. عملیات set Counter این مقادیر را با مقدار فرضی آرگومان آن مجدداً تنظیم می‌کند و عملیات IncrementCounter مقادیر hits را یکی افزایش می‌دهد (عملیات + + حاصل افزایش به‌وسیله یک عملیات است).

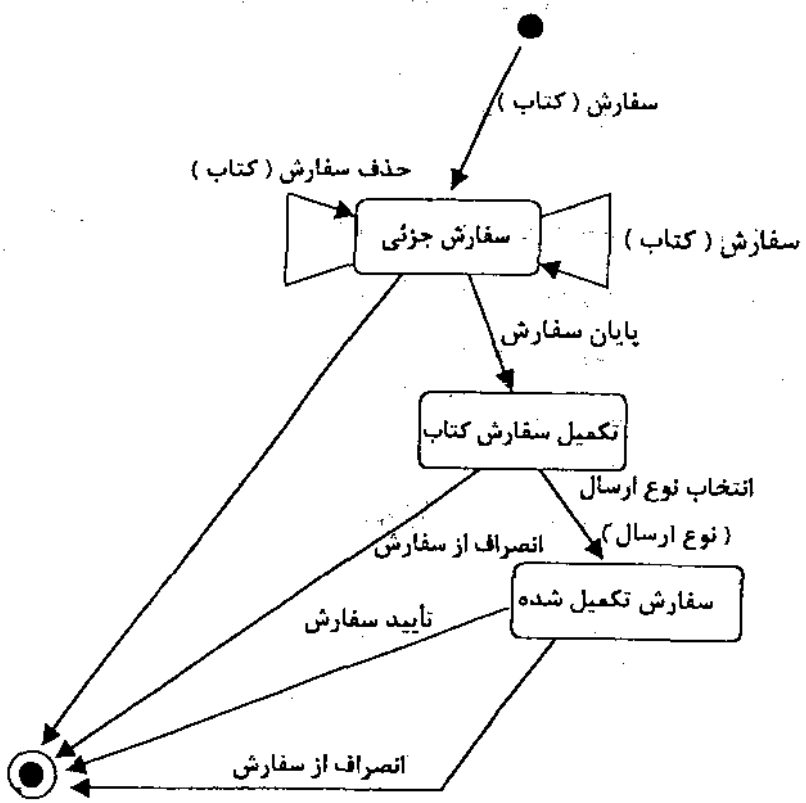
ترکیب زبان جاوا برای ارسال پیام‌ها به شیء بسیار ساده است و عبارتست از:

```
Object name. Operation (Arguments)
```

مثلاً برای افزایش شیء HitCounter برنامه hc خواهد بود:

```
hc. incrementCounter( )
```

کلاس فوق بسیار ساده است. این کلاس مشخصه‌های ساختار اصلی چگونگی تعریف یک کلاس را تشریح می‌کند. پیچیدگی‌های بیشتری وجود دارد مثل این حقیقت که چندین سطح رؤیت را می‌توان مشخص نمود. این مورد خارج از دامنه کتاب مهندسی نرم‌افزار است.



شکل ۲۲ـ۲۸ یک نمودار وضعیت

دو راه برای ترکیب کلاس‌ها وجود دارد: اولی عبارتست از وراثت و دومی تجمیع. زبان‌های برنامه‌نویسی شیءگرا حاوی تسهیلاتی هستند که هر دو این کارها را به‌راحتی ممکن می‌سازد.

در جاوا کلمه کلیدیextends برای اشتیاق کلاس‌ای از درون کلاس موجود به‌کار می‌رود یعنی همان وراثت. مثلاً فرض کنیم که لازم است کلاس جدیدی را که بسیار شبیه کلاسHitCounter است به‌دست آوردیم. اما زمانی را که در آن این بازدیدها رخ داده نیز ذخیره کنیم. ساختار چنین کلاس جدیدی که از HitCounter به‌صورت وراثت به‌وجود می‌آید در زیر آمده است:

```
class TimeHitCounter extends HitCounter
{
// any new attributes

// any new operations

}
```

کلمه کلیدیextends این حقیقت را بیان می‌دارد که کلاس TimedHitCounter وارث کلاس HitCounter است. برنامه این کلاس در زیرآمده است:

```
class TimeHitCounter extends HitCounter
{
Time timeAccesssed;
public TimeHitCounter(int startValue)
{
super(startValue);
timeAccesssed = new TimeAccesssed();
}
public void setCounter(int value)
{
super.setCounter(value);
timeAccesssed.setNow();
}
public Time getTime()
{
return timeAccesssed;
}
public void incrementCounter()
{
super incrementCounter();
timeAccesssed.setNow();
}
}
```

به‌خاطر دارید که در وراثت عملیات کلاس مافوق (در مورد ما HitCounter) به‌وسیله کلاس فرعی به بعدی می‌رسیدند (که در مورد ما TimedHitCounter است) مگر این‌که باطل شده باشند.

اولین عملیات یعنی setCounter روش مربوطه را در کلاس فرعی باطل می‌سازد. این عملیات ابتدا صفات مشخصه hits را درون کلاس فرعی با فراخوانی سازنده آن (کلمه رمز Super برای آن استفاده می‌شود) راه‌اندازی می‌کند و سپس شیء جدید Time را می‌سازد. این شیء در جای دیگری تعریف شده و ما برنامه آن را نشان نمی‌دهیم. همچنین عملیات set Counter عملیات مربوطه را درون HitCounter لغو می‌کند. کُد داخل این کلاس ابتدا صفت خاصه hits کلاس فرعی را مساوی ارزش آن (Value) تنظیم می‌کند. سپس پیام setNow به صفات خاصه timeAccessed ارسال می‌شود تا ارزش آن را

برابر با زمان کنونی حساب کند، برنامه عملیات setNow بخشی از کلاس Time را تشکیل داده و نشان داده نمی‌شود. روش

get Time ساده است. تمام چیزی که انجام می‌دهد عبارتست از بازگرداندن مقدار صفات خاصه timeAccessed. عملیات Increment Coonter عملیات مربوطه را درون Hit Coonter لغو می‌کند. ابتدا ارزش هیتها را با فراخوانی عملیاتIncrement Coonter درون کلاس فرعی افزایش می‌دهد. سپس با ارسال پیام Set Now به آن، مقدار Time Accessed را تنظیم می‌کند. روش Get Coonter که از Hit Coonter به ارث رسیده لازم نیست لغو شود زیرا همه کاری که می‌کند این است که با مقدار مشخصه Hits که درون کلاس فرعی است بازمی‌گردد.

این‌گونه یک زبان برنامه‌نویسی شی‌ءگرا نوعی، عمل وراثت را اجرا می‌کند. اجرای تجمیع ساده‌تر است: همه آن‌چه لازم است عبارتست از در برگرفتن قسمت‌های جمع‌آوری شده به‌عنوان صفات خاصه یک کلاس. مثلاً کلاس Computer که در زیر آمده نمایان‌گر یک کامپیوتر است. این کلاس بخشی از یک سیستم را برای زمان‌بندی ساخت کامپیوترها نشان می‌دهد. یک کامپیوتر تجمیعی از یک نمایشگر، صفحه‌کلید، پردازنده و غیره است. این موارد در کلاس به‌صورت یکسری صفات خاصه نشان داده شده است:

```
class computer
{
private Monitor mon;
private Keyboard kb;
private Processor proc;
// Remainder of attributes

//Definition of all the operations associated with computer
}
```

به‌عنوان مثال نهایی از کُدهای بیشتری از جاوا، از روی مطالعه موردی بررسی نشان داده شده در شکل ۲۲-۶ برنامه بیشتری برای مشتری تولید کرده‌ایم. مشتری کسی است که با استفاده از اینترنت کتاب می‌خرد. برنامه برای بسیاری از روش‌ها کاملاً ساده است:

```
class customer
{
String name, address, creditCardType, password, securityInformation;
BuyingHistory bHist;
CurrentOrders cOrders;
Preferences custPreferences;
```

```
Public getCustPreferences;
{
return custPreferences;
}
public String getName()
{
return name;
}
public String getAddress()
{
return Address;
}
public String getCreditCardType()
{
return creditCardType;
}
public String getPassword()
{
return password;
}
public String getSecurityInformation()
{
return SecurityInformation()
}
public void setName(String nm)
{
name = nm;
}
public void setAddress(String addr)
{
address = addr;
}
public void setCreditCardType(String ccType)
{
CreditCardType = ccType;
}
public void setPassword(String pWord)
{
password = pWord;
```

```
}
public void setSecurityInformation(String secInfo)
{
SecurityInformation = secInfo;
}
public void setCustomerPreference(Preferences pf)
{
custPreference = pf;
}
public void initialiseBuingHist()
{
// initialize the buing history using the method setClear
bHist.setClear();
}
public void addBuingTransaction(BuingTransaction bt)
{
// uses the metho add defined in BuingHist to add a book
// BuingTransaction to the customer object
bHist.add(bt);
}
public BuingHistory getHist()
{
return bHist;
}
public void initialiseCurrentOrders()
{
// uses the method setClear found in CurrentOrders to
// initialise the current orders
cOrders.setClear();
}
public void addOrder(Order ord);
{
// uses the method addCurrentOrder found in CurrentOrders
cOrders.addCurrentOrder(ord);
}
public void removeOrder(Order ord)
{
// uses the method removeCurrentOrder found in CurrentOrders
cOrders.removeCurrentOrder(ord);
```

```
}
public int noOfCurrentOrders()
{
// uses the method no found in CurrentOrders
return cOrders.no();
}
public CurrentOrders getCurrentOrders()
{
return cOrders;
}
...
}
```

بسیاری از روش‌ها ساده هستند: تمام آن‌چه که انجام می‌دهند عبارتست از تنظیم یا به‌دست آوردن مقادیر صفات خاصه‌ای که در شیء Customer یافت می‌شوند. این صفات خاصه عبارتند از:

- name ـ نام مشتری
- address ـ آدرس مشتری
- creditCardType ـ رشته‌ای که نوع کارت اعتباری مورد استفاده مشتری را نشان می‌دهد.
- passWord ـ رشته مورد استفاده مشتری برای دسترسی به سایت وب کتاب فروشی.
- secvrityInformation ـ این رشته‌ای است که توسط مشتری برای شناساندن خودشان به کارکنان سبرویس در مورد فراموش کردن کلمه رمزشان، استفاده می‌شود. مثلاً، ممکن است شامل مجل تولد مشتری باشد.
- BHist ـ این سابقه کتاب‌هایی است که مشتری خریده است.
- cOrders ـ شامل جزئیات هر سفارشی است که در حال حاضر مشهود است. مثلاً سفارشی که فعلاً قابل تأمین نیست.
- preferences ـ شامل فهرستی از اولویت‌های خرید برای مشتری. مثلاً این حقیقت که مشتری اغلب رمان‌های جنایی می‌خرد.

چندین شیوه وجود دارند که روش‌های تعریف شده در دیگر کلاس‌ها را فرا می‌خوانند. مثلاً روش removeOrder که یک سفارش مهم را از جزئیات مربوط به مشتری پاک می‌کند که از روش removeCorrentOrder از کلاس Current Orders استفاده می‌کند.

۲۲-۸ خلاصه

طراحی شی‌گرا مدل OOA را از جهان واقعی به مدل خاص پیاده‌سازی تبدیل می‌کند، که در نرم‌افزار شناسایی می‌شود. فرآیند طراحی شی‌گرا را می‌توان به‌صورت هرمی تشریح کرد که چهار لایه دارد. اولین لایه زیربن بر طراحی سیستم‌های فرعی متمرکز است که کارکردهای اصلی سیستم را پیادسازی می‌کند. لایه کلاس، معماری شی، و سلسله مراتب کلی کلاس را که برای پیاده‌سازی سیستم لازم است نشان می‌دهد. لایه پیام نشان‌گر چگونگی همکاری میان اشیا است که مشخص می‌شوند و لایه وظایف روش ها و عملیاتی را شناسایی می‌کند که هر کلاس را توصیف می‌نمایند.

مانند تحلیل شی‌گرا، روش‌های متفاوتی در طراحی شی‌گرا وجود دارد. UML تلاشی است برای به‌وجود آوردن رهیافتی در طراحی شی‌گرا که در تمام حوزه‌های کاربردی قابل کاربرد باشد. UML و دیگر روش‌ها از طریق دو سطح تجریدی به فرآیند طراحی می‌رسند ـ طراحی سیستم‌های فرعی و طراحی اشیای تکی.

در طول طراحی سیستم، معماری سیستم شی‌گرا گسترش می‌یابد. علاوه بر توسعه سیستم‌های فرعی، روابط متقابل آنها و جایگاه آنها در لایه‌های معماری، طراحی سیستم جزء تعامل کاربر، جزء مدیریت وظائف و مدیریت داده‌ها را نیز مدنظر دارد. این جزءهای سیستم فرعی زیربنای طراحی را مهیا می‌سازند که برنامه کاربردی را قادر می‌سازد به‌طور مؤثر عمل کند. فرآیند طراحی شی، روی توصیف ساختارهای داده‌ای متمرکز است که صفات خاصه کلاس را پیاده‌سازی کرده، الگوریتم‌هایی که عملیات را پیاده‌سازی می‌کنند و پیام‌هایی که همکاری و ارتباطات اشیاء را مقدور می‌سازند.

الگوهای طراحی به طراح امکان می‌دهند با یکپارچه‌سازی جزءهای قابل استفاده مجدد، معماری سیستم را ایجاد کند. برنامه‌نویسی شی‌گرا مدل طراحی را در یک حوزه قابل اجرا گسترش می‌دهد. یک زبان برنامه‌نویسی شی‌گرا برای تبدیل کلاس‌ها، صفات مشخصه، عملیات و پیام‌ها به شکلی مورد استفاده قرار می‌گیرد که توسط ماشین قابل اجرا باشد.

مسایل و نکاتی برای تفکر و تعمق بیشتر

۲۲-۱ هرم طراحی OOD تا اندازه ای با هرم طراحی نرم‌افزارهای سنتی (فصل ۱۳) متفاوت است. اختلافات و شباهت‌های این دو هرم را توضیح دهید.

۲۲-۲ تفاوت طراحی OOD و ساخت یافته چیست؟ کدام جنبه‌ها از این دو شیوه طراحی یکسان هستند؟

۲۲-۳ پنج معیار را که برای پیمانه‌ای بودن مؤثر OO در بخش ۲۲-۱-۲ عنوان شد، مرور کنید. با استفاده از رهیافت طراحی که در اواخر فصل شرح داده شد، چگونگی دستیابی به این پنج معیار را نشان دهید.

۲۲-۴ با استفاده از مراجع مربوط به UML، یک ارائه یک ساعته برای کلاس خود آماده کنید. اطمینان حاصل کنید که کلیه قراردادهای مدل‌سازی تصویری مهم به کار رفته در UML ، را نشان داده اید.

۲۲-۵ یک روش قدیمی‌تر OOD را که در بخش ۲۲-۱-۳ ارائه شد، انتخاب کرده یک ارائه یک ساعته برای کلاس خود آماده کنید. اطمینان حاصل کنید که کلیه قراردادهای مدل‌سازی تصویری مهم را که نویسندگان پیشنهاد می‌کنند، نشان داده اید.

۲۲-۶ بحث کنید که چگونه «مورد کاربرد یا Use-Case» می‌تواند به‌عنوان یک منبع اطلاعات مهم در طراحی منظور گردد؟

۲۲-۷ درباره یک محیط توسعه GUI تحقیق نموده، نشان دهید چگونه اجزاء محاوره با کاربر در جهان واقعی پیاده‌سازی می‌شود. چه الگوهای طراحی پیشنهاد شده و چگونه به‌کار می‌روند؟

۲۲-۸ مدیریت وظایف برای سیستم‌های OO می‌تواند بسیار پیچیده باشد. قدری تحقیقات را روی شیوه های OO برای سیستم‌های زمان حقیقی (مثلاً [BIH91] یا [DOU99]) به انجام رسانید و تعیین کنید که چگونه مدیریت وظایف در آن بافت صورت می‌پذیرد؟

۲۲-۹ توضیح دهید که چگونه جزء مدیریت داده‌ها در یک محیط توسعه OO پیاده‌سازی می‌شود؟

۲۲-۱۰ مقاله‌ای دو یا سه صفحه‌ای درباره بانک‌های اطلاعاتی شی‌گرا بنویسید و توضیح دهید که چگونه باید از آنها برای توسعه جزء مدیریت داده‌ها استفاده نمود؟

۲۲-۱۱ سه برنامه کاربردی را شرح دهید که ممکن است از الگوی Memento استفاده کنند؟

۲۲-۱۲ سه برنامه کاربردی را شرح دهید که ممکن است از الگوی Singleton استفاده کنند؟

۲۲-۱۳ رهیافت طراحی شئ گرا که در این فصل مورد بحث واقع شد را، در مورد سیستم PHTRS که در مسئله ۱۲-۱۳ شرح داده شد، به‌کار ببرید.

۲۲-۱۴ یک بازی ویدئویی را شرح دهید و رهیافت طراحی شئ گرا ارائه شده در این فصل را در مورد طراحی آن به‌کار برید.

۲۲-۱۵ شما مسئول توسعه یک سیستم پست الکترونیک(E-mail) هستید که بناست بر یک شبکه PC پیاده‌سازی شود. این سیستم، کاربران را قادر می‌سازد تا نامه‌هایی را برای دیگر کاربران ایجاد و به آدرسهای مشخص ارسال نمایند. نامه‌ها می توانند خوانده شوند یا کپی و نسخه برداری شوند و یا ذخیره و نگهداری گردند و اعمالی نظیر آن. این سیستم پست الکترونیکی از واژه‌پرداز که هم اکنون موجود است جهت ایجاد نامه‌ها استفاده می‌کند. این تعریف را نقطه آغازین بدانید و مجموعه نیازمندیها را با استفاده از آن به دست آورید و تکنیکها و فنون طراحی شئ گرا را برای ایجاد یک طراحی سطح بالا جهت سیستم پست الکترونیک به‌کار برید.

۲۲-۱۶ یک جزیره‌ای کوچک تصمیم دارد که یک سیستم کنترل ترافیک هوایی (ATC) برای تنها فرودگاه خود بسازد. این سیستم به‌صورت زیر مشخص شده است:

تمام هواپیماهایی که قصد فرود در فرودگاه دارند، دارای فرستنده‌ای هستند که داده‌های مربوط به نوع هواپیما و پرواز را در قالب فشرده به ایستگاه زمینی ATC ارسال می‌کنند. ایستگاه زمینی ATC می‌تواند از هواپیما اطلاعات خاصی را درخواست نماید. هنگامی که ایستگاه زمینی ATC داده‌ها را دریافت می‌کند، از حالت فشرده خارج کرده در بانک اطلاعاتی هواپیما نگهداری می‌کند. داده‌های ذخیره شده به صورت گرافیکی در آمده و برای کنترل‌کننده ترافیک هوایی به نمایش درمی‌آید. صفحه نمایش هر ۱۰ ثانیه یک بار به‌هنگام می‌شود. تحلیلی بر تمام اطلاعات صورت می پذیرد تا تعیین شود که آیا «شرایط خطرناک» وجود دارد یا خیر. کنترل‌کننده ترافیک هوایی می‌تواند برای هر هواپیمایی بر صفحه نمایش دیده می شود درخواست اطلاعات خاص (از بانک اطلاعاتی) داشته باشد.

با استفاده از طراحی شئ گرا یک طراحی برای سیستم ATC ایجاد کنید. برای پیاده سازی آن تلاش نکنید!

فهرست منابع و مراجع

[BEN99] Bennett,S., S. McRobb, and R. Farmer, *Object Oriented System Analysis and Design Using UML*, McGraw-Hill, 1999.

[BIH92] Bihari, T. and P. Gopinath, "Object-Oriented Real-Time Systems: Concepts and Examples," *Computer*, vol. 25, no. 12, December 1992, pp. 25-32.

[BOO94] Booch, G., *Object-Oriented Analysis and Design*, 2nd ed., Benjamin Cummings, 1 994.

[BOO99] Booch, G., I. Jacobson, J. Rumbaugh, *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.

[BRO91] Brown, A.W., *Object-Oriented Databases*, McGraw-Hill, 1991.

[BUS96] Buschmann, F., et al., *A System of Patterns: Pattern Oriented System Architecture*, Wiley, 1996.

[CHA93] De Champeaux, D., D. Lea, and P. Faure, *Object-Oriented System Development*, Addison-Wesley, 1993.

[COA91] Coad, P. and E. Yourdon, *Object-Oriented Design*, Prentice-Hall, 1991.

[COX85] Cox, B., "Software ICs and Objective-C," *UnixWorld*, Spring 1985.

[DAV95] Davis, A., "Object-Oriented Requirements to Object-Oriented Design: An Easy Transition?" *journal of Systems Software*, vol. 30, 1995, pp. 151-159.

[DOU99] Douglass, B., *Real-Time UML: Developing Efficient Objects for Embedded Systems*, Addison-Wesley, 1999.

[FIC92] Fichman, R. and C. Kemerer, "Object-Oriented and Conceptual Design Methodologies," *Computer*, vol. 25, no. 10, October 1992, pp. 22-39.

[GAM95] Gamma, E., et al., *Design Patterns*, Addison-Wesley, 1995.

[GOL83] Goldberg, A. and D. Robson, *Smalltalk -80: The Language and Its Implementation*, Addison-Wesley, 1983.

[JAC92] Jacobson, I., *Object-Oriented Software Engineering*, Addison-Wesley, 1992.

[JAC99] Jacobson, I., G. Booch, J. Rumbaugh, *Unified Software Development Process*, Addison-Wesley, 1999.

[MEY90] Meyer, B., *Object-Oriented Software Construction*, 2nd ed., Prentice-Hall, 1 988.

[PRE95] Pree, W., *Design Patterns for Object-Oriented Software Development*, Addison-Wesley, 1995.

[RUM91] Rumbaugh, J., et al., *Object-Oriented Modeling and Design*, Prentice-Hall, 1 991.

[RUM99] Rumbaugh, J., I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1999.

[RAO94] Rao, B.A., *Object-Oriented Databases: Technology, Applications and Products*, McGraw-Hill, 1994.

[TAY92] Taylor, D.A., *Object-Oriented Information Systems*, Wiley, 1992.

[WIR90] Wirfs-Brock, R., B. Wilkerson, and L. Weiner, *Designing Object-Oriented Software*, Prentice-Hall" 1990.

خواندنیهای دیگر و منابع اطلاعاتی

In addition to the many references in this chapter, books by Gossain and Graham. *(Object Modeling and Design Strategies*, SIGS Books, 1998); Meyer *(Object-Oriented*

Software Construction, 2nd ed., Prentice-Hall, 1997); Reil *(Object-Oriented Design Through Heuristics*, Addison-Wesley, 1996); and Walden and Nerson *(Seamless Object-Oriented Software Architecture: Analysis and Design of Reliable Systems*, Prentice-Hall, 1995) cover 000 in considerable detail. Fowler *(Refactoring: Improving the Design of Existing Code*, Addison-Wesley, 1999) addresses the use of object -oriented techniques to redesign and rebuild old programs to improve their design quality.

Many recent books published on object -oriented design emphasize UML. Those serious about applying UML in their work should acquire [BO099], [RUM99], and [JAC99]. In addition, many of the books referenced in the Further Reading and Information Sources section of Chapter 21 also address design in considerable detail.

The use of design patterns for the development of object -oriented software has important implications for component -based software engineering, reusability in general, and the overall quality of resultant systems. In addition to [BUS96] and [GAM95], many recent books are dedicated to the subject:

Ambler, S.W., *Process Patterns: Building Large-Scale Systems Using Object Technology*, Cambridge University Press, 1999.

Coplien, J.O. and D.C. Schmidt, *Pattern Languages of Program Design*, Addison-Wesley, 1995.

Fowler, M., *Analysis Patterns: Reusable Object Models*, Addison-Wesley, 1996.

Larman, C., *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*, Prentice-Hall, 1997.

Martin, R.C., et al., *Pattern Languages of Program Design* 3, Addison-Wesley, 1997.

Rising, L. and J. Coplien (eds.), *The Patterns Handbook: Techniques, Strategies, and Applications*, SIGS Books, 1998.

Pree, W., *Design Patterns for Object-Oriented Software Development* Addison-Wesley, 1995.

Vlissides, J., Pattern Hatching: *Design Patterns Applied*, Addison-Wesley, 1998.

Vlissides, J.M., J.O. Coplien, and N. Kerth, *Pattern Languages of Program Design* 2, Addison-Wesley, 1996.

Hundreds of books have been published on object-oriented programming. A sampling of OOP language-specific books follows:C++: Cohoon, J.P., c++ *Program Design: An Introduction to Programming and Object-Oriented Design*, McGraw Hill, 1998.

Barclay, K. and J. Savage, *Object-Oriented Design with* C++, Prentice-Hall, 1997.

Eiffel: Thomas, P. and R. Weedon, *Object-Oriented Programming in Eiffel*, Addison-Wesley, 1997.

Jezequel, J.M., *Object-Oriented Software Engineering with Eiffel*, Addison-Wesley, 1996.

Java: Coad, P., M. Mayfield, ,and J. Kern, *Java Design: Building Better Apps and Applets*, 2nd ed., Prentice-Hall, 1998.

Lewis, J. and. W. Loftus, *Java Software Solutions: Foundations of Program*, Addison Wesley, 1997.

Smalltalk: Sharp, A., *Smalltalk by Example: The Developer's Guide*, McGraw-Hill, 1997.

lalonde, W.R. and J.R. Pugh, *Programming in Smalltalk*, Prentice-Hall, 1995.

Books that cover 000 topics using two or more 00 programming languages provide insight and comparison of language features. Titles include:

Drake, C., *Object-Oriented Programming With c++ and Smalltalk*, Prentice-Hall, 1998.

Joyner, I., *Objects Unencapsulated: Java, Eiffel and* C++, Prentice-Hall, 1999.

Zeigler, B.P., *Objects and Systems: Principled Design with Implementations in* C++ *and Java*, Springer-Verlag, 1997.

A wide variety of information sources on object-oriented design and related subjects is available on the Internet. An up-to-date list of World Wide Web references that are relevant to 000 can be found at the SEPA Web site:

http://www.mhhe.com/ engcs/ compsci/pressman/resources/OOD.mhtml