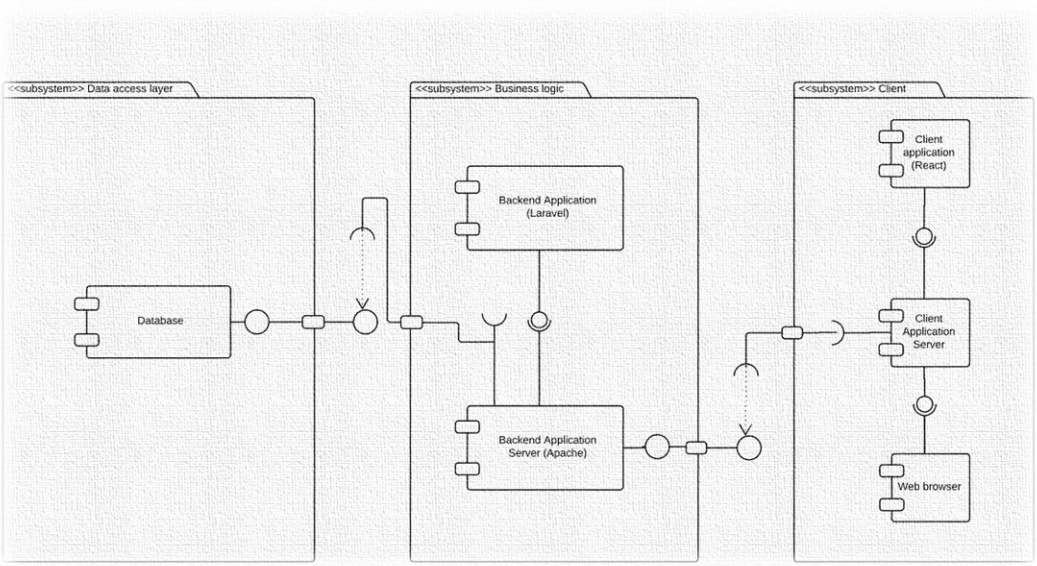
نمودار مؤلفه یا Component Diagram در UML



تهیه و تنظیم: پیمان مالکی

فهرست مطالب

نمودارمؤلفه

نمودارمؤلفه‌ها[1]، از **مؤلفه‌ها**[2]، **واسط‌های‌ارائه‌شده**[3]، **واسط‌های‌موردنیاز**[4]، **درگاه‌ها**[5] و روابط بین آنها تشکیل شده است. این نوع نمودارها در توسعه‌مبتنی‌برمؤلفه[6] (CBD) برای توصیف سیستم‌های دارای معماری‌سرویس‌گرا[7] (SOA) استفاده می‌شود. توسعه‌مبتنی‌برمؤلفه بر این فرض استوار است که **مؤلفه‌های** ساخته‌شده قبلی می‌توانند مجدداً مورد استفاده قرار گیرند و در صورت نیاز می‌توان **مؤلفه‌ها** را با برخی **مؤلفه‌های** "معادل" یا "سازگار" جایگزین کرد.

مصنوعاتی که **مؤلفه** را پیاده‌سازی می‌کنند به گونه‌ای در نظر گرفته شده‌اند که بتوانند به طور مستقل، مستقر شوند و مجدداً مستقر شوند، به عنوان مثال برای به‌روزرسانی یک سیستم موجود. در UML **مؤلفه‌ها** می‌توانند معرف موارد زیر باشند یا مشتمل بر این موارد باشند:

- **مؤلفه‌های** منطقی (به عنوان مثال، **مؤلفه** کسب‌وکاری، **مؤلفه** فرآیندی)
- **مؤلفه‌های** فیزیکی (به عنوان مثال، **مؤلفه** CORBA، **مؤلفه** EJB، **مؤلفه** COM+ و **مؤلفه** .NET.، **مؤلفه** WSDL و غیره)

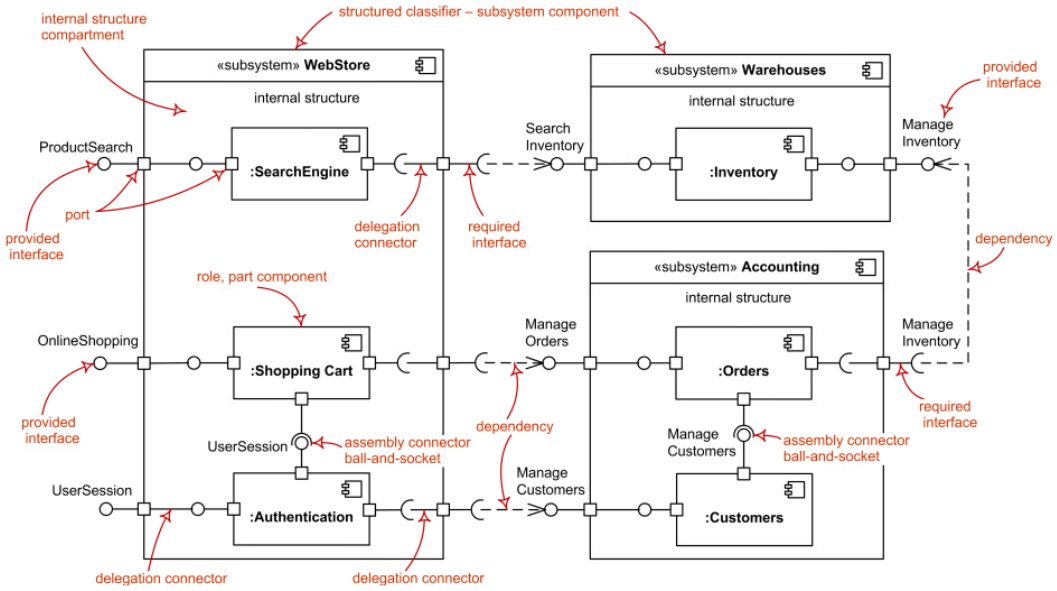
به همراه مصنوعاتی که آنها را پیاده‌سازی می‌کنند و **نودهایی** که روی آنها مستقر و اجرا می‌شوند. پیش‌بینی می‌شود که پروفایل‌های مبتنی بر **مؤلفه‌ها** برای فناوری‌های **مؤلفه** خاص و محیط‌های سخت‌افزاری و نرم‌افزاری مرتبط ایجاد شود. **نودها** و **انتقال‌دهنده‌های** زیر معمولاً در **نمودارمؤلفه** ترسیم می‌شوند:

- **مؤلفه**
- **واسط**
- **واسط‌ارائه‌شده**
- **واسط‌موردنیاز**
- **کلاس**
- **درگاه**
- **اتصال‌دهنده**
- **مصنوع**
- **ارتباط از نوع تحقق مؤلفه**
- **ارتباط از نوع وابستگی نقطه‌به‌نقطه یک‌طرفه**
- **کاربرد**

این عناصر اصلی در تصویر زیر نشان داده شده است.

---

[1] Component diagram
[2] Component
[3] Provided interface
[4] Required interface
[5] Port
[6] Component Base Development
[7] Service-Oriented Architecture



عناصر اصلی نمودار مؤلفه UML شامل: مؤلفه، واسط‌ارائه‌شده، واسط‌موردنیاز، درگاه، اتصال‌دهنده

چه زمانی و به چه دلایلی تهیه این نمودار اهمیت خواهد داشت؟

هدف از **نمودارمولفه** نشان دادن رابطه بین **مؤلفه‌های** مختلف در یک سیستم است. در UML 2.0، اصطلاح "component" یا **مؤلفه** به ماژولی از **کلاس‌ها** اشاره دارد که سیستم‌ها یا زیرسیستم‌های مستقل را با قابلیت ارتباط با بقیه سیستم نشان می‌دهند.

نمودارمؤلفه، عملکرد سیستم را توصیف نمی‌کند، اما **مؤلفه‌های** مورد استفاده برای ایجاد قابلیت‌های مورد نظر سیستم را توصیف می‌کند. بنابراین از این دیدگاه، **نمودارهای‌مؤلفه** برای تجسم **مؤلفه‌های** فیزیکی در یک سیستم استفاده می‌شوند. این **مؤلفه‌ها**، **کتابخانه‌ها**، **پکیج‌ها**، **فایل‌ها** و غیره را تشکیل می‌دهند.

زمانی که می‌خواهید سیستم خود را به شکل مجموعه‌ای از **مؤلفه** و روابط متقابل آنها را از طریق **واسط‌ها** نمایش دهید، می‌توانید از **نمودارمؤلفه** استفاده کنید. این به شما کمک می‌کند تا ایده‌ای از پیاده‌سازی سیستم داشته باشید.

خلاصه و چکیده – Component Diagram

نمودارمؤلفه، نموداری است که در توسعه‌مبتنی‌برمؤلفه (CBD) اهمیت دارد و این نوع توسعه در معماری‌سرویس‌گرا (SOA) مورد کاربرد دارد. این نمودار روابط بین **مؤلفه‌ها**، **درگاه‌ها**، **واسط‌های‌ارائه‌شده**، **واسط‌های‌موردنیاز** و **اتصال‌دهنده** آنها را نمایش می‌دهد. CBD بر این قاعده کلی استوار است که، به گونه‌ای طراحی کنیم که بتوانیم عناصر یک سیستم را به صورت **مؤلفه‌های** قابل استفاده مجدد، طراحی کنیم که باعث شود توسعه صورت گرفته بهینه شده و از کد توسعه یافته در موارد مختلف به صورت تکرار شونده استفاده کنیم. هدف این توسعه، کاهش میزان کد و جلوگیری از تکرار کد است که در نتیجه باعث کاهش هزینه توسعه و نگهداری می‌شود.

یک **مؤلفه** می‌تواند از مصنوعات زیادی تشکیل شود، این مصنوعات می‌توانند **مؤلفه** باشند و در نتیجه به صورت تودرتو این **مؤلفه‌ها** در قالب مصنوعات مورد استفاده قرار گیرند. این مصنوعات یا به عبارت دیگر این **مؤلفه‌ها** دو دسته کلی دارند:

- **مؤلفه‌های** منطقی (مانند: **مؤلفه** کسب‌وکاری، **مؤلفه** فرآیندی)
- **مؤلفه‌های** فیزیکی (مانند: **مؤلفه** CORBA، **مؤلفه** EJB، **مؤلفه** COM+ و **مؤلفه** .NET، **مؤلفه** WSDL و غیره)

به طور کلی موارد زیر در یک **نمودارمؤلفه** قابل مشاهده خواهند بود:

- **مصنوع**

- **مؤلفه** (**مؤلفه‌های** منطقی یا **مؤلفه‌های** فیزیکی)
- **واسط**
- **واسط‌ارائه‌شده**
- **واسط‌موردنیاز**
- **کلاس**
- **درگاه**
- **اتصال‌دهنده**
- ارتباط از نوع **تحقق مؤلفه**
- ارتباط از نوع **وابستگی نقطه‌به‌نقطه یک‌طرفه**
- **کاربرد**

مؤلفه

مؤلفه، کلاسی است که نمایانگر بخش ماژولار یک سیستم با محتوای محصور شده است و جلوه آن در محیط‌های مختلف، قابل تعویض است. رفتار یک **مؤلفه** بر حسب **واسط‌های‌ارائه‌شده** و **واسط‌های‌موردنیاز** (که به طور بالقوه از طریق **درگاه‌ها** قابل دسترس خواهند بود) تعریف شده است.

مؤلفه به عنوان اصطلاحی عمل می‌کند که انطباق آن توسط این **واسط‌های‌ارائه‌شده** و **واسط‌های‌موردنیاز** (شامل معنایی استاتیک و پویا آنها) تعریف می‌شود. بنابراین تنها در صورتی می‌توان یک **مؤلفه** را با دیگری جایگزین کرد که این دو با اصطلاح مورد نظر، سازگار باشند. قطعات بزرگتر از عملکرد یک سیستم ممکن است با استفاده مجدد از **مؤلفه‌ها** به عنوان قطعات یک **مؤلفه** دربرگیرنده یا سرهم شدن مجموعه‌ای از **مؤلفه‌ها** و **واسط‌های‌موردنیاز** و **واسط‌های‌ارائه‌شده** آنها محقق شود.

یک **مؤلفه** در طول چرخه عمر توسعه، مدل شده و به طور متوالی در زمان استقرار و زمان اجرا، پالایش می‌شود. یک **مؤلفه** ممکن است با یک یا چند مصنوع، محقق شود.

مؤلفه، یک عنصر نمونه‌سازی‌شده غیرمستقیم در زمان طراحی، تعریف می‌شود اما در زمان اجرا به عنوان یک **شیء** آدرس پذیر، وجود نخواهد داشت. رفتار زمان اجرا **مؤلفه** و **درگاه‌های** آن با رفتار زمان اجرا **طبقه‌بندی‌کننده‌ها** یا **قطعاتی** که آن را تشخیص می‌دهند، تعریف می‌شود. چندین **کلیشه** استاندارد این ویژگی را بیشتر توصیف می‌کنند، به عنوان مثال، «Specification»، «Focus»، «Subsystem».

مؤلفه داخلی، اگر توسط **واسطی** ارائه نشود، پنهان و غیرقابل دسترسی خواهد بود. حتی اگر توسط **واسط‌های‌موردنیاز** به عناصر دیگر وابسته باشد، یک **مؤلفه** محصور شده محسوب‌می‌شود و وابستگی‌های آن به گونه‌ای طراحی شده است که بتوان آن را تا حد امکان به طور مستقل مورد بررسی قرار داد.

نماد مؤلفه

یک **مؤلفه** به شکل یک مستطیل **طبقه‌بندی‌کننده** با کلمه کلیدی «component» نشان داده می‌شود.

مؤلفه WeatherServices

به صورت اختیاری، یک نماد **مؤلفه** می‌تواند در گوشه سمت راست بالای مستطیل مورد نظر، نمایش داده شود. اگر نماد مورد نظر نشان داده شود، ممکن است کلمه کلیدی «component» حذف شود.

مؤلفه UserServices

یک **مؤلفه** ممکن است توسط یک یا چند مصنوع تجلی یابد، و به نوبه خود، آن مصنوع ممکن است در محیط اجرای آن مستقر شود. در مشخصات استقرار ممکن است مقادیری تعریف شود که اجرای **مؤلفه** را پارامترپذیر کند.

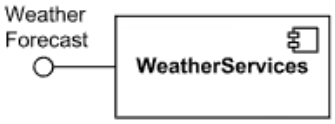
واسط‌ارائه‌شده

یک **واسط‌ارائه‌شده**[8] می‌تواند به یکی از روش‌های زیر محقق شود:

- به طور مستقیم توسط خود **مؤلفه**، محقق شده است

[8] Provided Interface

- توسط یکی از **طبقه‌بندی‌کننده‌های** محقق کننده **مؤلفه**، محقق شده است
- توسط یک **درگاه** عمومی **مؤلفه**، تهیه شده است

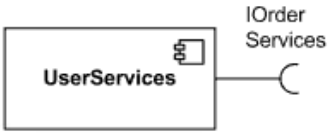


مؤلفه Weather Services واسط Weather Forecast را تهیه می‌کند (پیاده‌سازی می‌کند)

واسط‌موردنیاز

یک **واسط‌موردنیاز**[9] یکی از موارد زیر است:

- تعیین‌شده توسط **وابستگی نقطه‌به‌نقطه یک‌طرفه کاربرد** از خود **مؤلفه**
- تعیین‌شده توسط **وابستگی نقطه‌به‌نقطه یک‌طرفه کاربرد** از یکی از **طبقه‌بندی‌کننده‌های** محقق کننده **مؤلفه**
- تعیین‌شده توسط یک **درگاه** عمومی **مؤلفه**



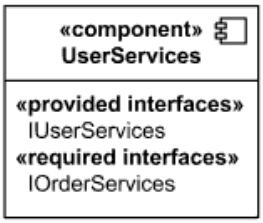
مؤلفه User Services به واسط‌موردنیاز IOrderServices

نمای‌خارجی‌مؤلفه

یک **مؤلفه** دارای یک **نمای‌خارجی‌مؤلفه**[10] یا نمای "جعبه سیاه" دارد که با استفاده از نمادهای **واسط** که از نماد **مؤلفه** بیرون می‌آیند و **مشخصات** و **عملکردهای** قابل‌مشاهده برای عموم را نشان می‌دهند، تعریف می‌شود.

به صورت اختیاری، **رفتاری** مانند یک پروتکل **ماشین‌وضعیت** ممکن است به یک **واسط**، **درگاه** و به خود **مؤلفه** متصل شود تا **نمای‌خارجی‌مؤلفه** را با نمایش **محدودیت‌های** دینامیکی در **توالی فراخوانی‌های عملیاتی**، با دقت بیشتری تعریف کند. سایر **رفتارها** نیز ممکن است با **واسط‌ها** یا **اتصال‌دهنده‌ها** برای تعریف "قرارداد" بین شرکت‌کنندگان در یک **همکاری** (مثلاً از نظر **مواردکاربرد**، **فعالیت** یا مشخصات **تعامل**) مرتبط باشند.

از طرف دیگر، **واسط‌ها** و/یا **عملیات‌ها** و **ویژگی‌ها** را می‌توان در بخش‌های یک نماد **مؤلفه** فهرست کرد (برای اینکه نوشتار در محل مورد نظر کنجانده شود، ابزارها ممکن است راهی برای فهرست‌بندی و مخفف کردن **ویژگی‌ها** و **رفتار مؤلفه‌ها** ارائه دهند).



نمای‌خارجی‌مؤلفه User Services شامل واسط‌ارائه‌شده IUserServices و واسط‌موردنیاز IorderServices

[9] Required Interface
[10] External View of Component

برای نمایش **علامت‌معرف**[11] کامل یک **واسط** مربوط به یک **مؤلفه**، واسط مورد نظر می‌تواند مانند مستطیل **طبقه‌بندی‌کننده** معمولی نمایش داده شود که می‌تواند برای نمایش جزئیات **عملیات** و **رویدادها** گسترش یابد.

کلیشه‌های استاندارد مؤلفه

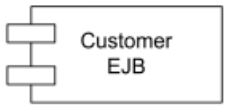
چندین **کلیشه** استاندارد UML وجود دارد که برای **مؤلفه‌ها** اعمال می‌شود:

نام کلیشه استاندارد	شرح
«BuildComponent»	مجموعه‌ای از عناصر تعریف شده برای فعالیت‌های توسعه سطح سیستم، مانند کامپایل و نسخه‌سازی.
«Entity»	یک **مؤلفه** اطلاعاتی پایدار که یک مفهوم کسب‌وکاری را نشان می‌دهد.  مؤلفه موجودیت Customer
«Implement»	**کلیشه** «Implement»، **کلیشه‌ای** برای تعریف **مؤلفه‌ای** است که برای آن **مشخصات** درونی در نظر گرفته نشده است. یا به عبارت دیگر این **مؤلفه** یک پیاده‌سازی برای یک **مؤلفه** با **کلیشه** «Specification» جداگانه است که به آن **وابستگی نقطه‌نقطه یک‌طرفه تحقق** دارد.
«Process»	**کلیشه** «Process»، یک **مؤلفه** مبتنی بر تراکنش را معرفی می‌کند.
«Realization»	**کلیشه** «Realization»، یک **طبقه‌بندی‌کننده** است که دامنه‌ای از **اشیاء** را مشخص می‌کند و همچنین اجرای فیزیکی آن **اشیاء** را تعریف می‌کند. به عنوان مثال، یک **مؤلفه** کلیشه‌شده توسط «Realization» تنها دارای **طبقه‌بندی‌کننده‌های** واقعی است که رفتار مشخص‌شده توسط **مؤلفه** «Specification» جداگانه را پیاده‌سازی می‌کنند. این با «ImplementationClass» متفاوت است، زیرا یک **کلاس** پیاده‌سازی، **تحقق کلاسی** است که می‌تواند **امکاناتی** مانند **ویژگی‌ها** و **متودهایی** داشته باشد که برای طراحان سیستم مفید است.
«Service»	**کلیشه** «Service»، یک **مؤلفه** عملکردی stateless است.  مؤلفه سرویس WeatherServices
«Specification»	**کلیشه** «Specification»، **طبقه‌بندی‌کننده‌ای** است که دامنه‌ای از **اشیاء** را بدون تعریف پیاده‌سازی فیزیکی آن **اشیاء** مشخص می‌کند. به عنوان مثال، یک **مؤلفه** کلیشه‌شده با «Specification» تنها دارای **واسط‌های‌ارائه‌شده و واسط‌های‌موردنیاز** است، و پیش‌بینی نشده که **طبقه‌بندی‌کننده‌ای** بخشی از آن را تعریف کند. این با **کلیشه** «type» تفاوت دارد، زیرا یک **کلیشه** «type» می‌تواند **امکاناتی** مانند **ویژگی‌ها** و **متودهایی** داشته باشد که برای تحلیلگران سیستم در حال مدل‌سازی، مفید باشد. **کلیشه** «Specification» و «Realization» برای مدل‌سازی **مؤلفه‌ها** با تعاریف مشخص و تحقق متمایز استفاده می‌شوند، که در آن یک **مشخصه** ممکن است چند تحقق داشته باشد.
«Subsystem»	**کلیشه** «Subsystem»، **مولفه‌ای** است که واحد تجزیه سلسله مراتبی را برای سیستم‌های بزرگ نشان می‌دهد و برای مدل‌سازی **مؤلفه‌های** مقیاس بزرگ استفاده می‌شود. تعاریف زیرسیستم‌ها ممکن است در حوزه‌های مختلف و متودهای نرم‌افزاری متفاوت باشد. انتظار می‌رود که پروفایل‌های دامنه و متود این عنصر را تخصصی کنند. یک زیرسیستم معمولاً به طور غیر مستقیم نمونه‌سازی می‌شود. یک زیرسیستم ممکن است دارای مشخصات و عناصر تحقق باشد.

[11] signature

سابقه

نماد **مؤلفه** به شکل یک مستطیل **طبقه‌بندی‌کننده** با کلمه کلیدی «component» در UML 2.0 معرفی شد. در نسخه‌های قبلی UML 1.x نماد **مؤلفه** یک مستطیل بود که دو مستطیل کوچک از یک طرف آن بیرون زده بود. به دلایل سازگاری با قبل، این نماد ممکن است همچنان در UML 2.5 استفاده شود.



نماد مؤلفه CustomerEJB در UML 1.x

در UML 1.4.2 **کلیشه** «entity» یک **کلاس** غیرفعال را نشان می‌دهد، یعنی **کلاسی** که **اشیاء** آن به خودی خود تعامل را آغاز نمی‌کنند. ولی «Entity» به یک **مؤلفه** اطلاعات پایدار در UML 2.0 تبدیل شد.

در UML 1.4.2 **کلیشه** «process» **طبقه‌بندی‌کننده‌ای** را مشخص می‌کند که یک جریان کنترل چندمرحله‌ای را نشان می‌دهد. ولی «Process» به یک **مؤلفه** مبتنی بر تراکنش در UML 2.0 تبدیل شد.

در UML 1.4.2 **کلیشه** «subsystem» نوع خاصی از **پکیج** است که برای نشان دادن یک واحد رفتاری در سیستم فیزیکی است و از این رو در **نمودارمدل** استفاده می‌شود و به عنوان یک واحد مشخصات برای **رفتار** عناصر موجود در **نمودارمدل** عمل می‌کند. ولی «Subsystem» به یک **کلیشه مؤلفه** تبدیل شد تا واحد تجزیه سلسله مراتبی را برای سیستم‌های بزرگ در UML 2.0 نشان دهد.

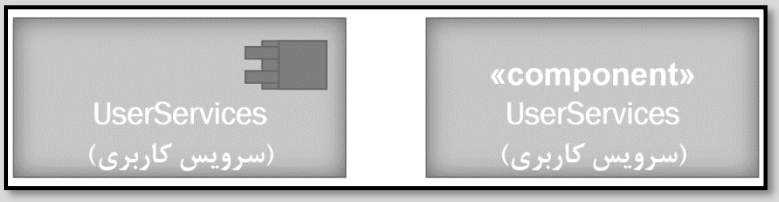
UML 2.0 همچنین **کلیشه‌های مؤلفه** استاندارد «BuildComponent»، «Implement» و «Service» را معرفی کرد.

خلاصه و چکیده – Component

مؤلفه، یک بخش ماژولار محصور شده است که بنا به **واسط‌های‌ارائه‌شده** و **واسط‌های‌موردنیاز** آن در محیط‌های مختلف قابل ارائه و شناسایی است. یعنی ماژول یا ماژول‌هایی را در یک محیط گردآوری کرده و توسعه می‌دهیم و سپس برای استفاده در هر محیطی، فقط کافی است که **واسط‌های** آن را تغییر دهیم.

رفتار هر **مؤلفه** به واسطه رفتار زمان اجرا **طبقه‌بندی‌کننده‌ها** یا **قطعاتی** که آن را تشکیل می‌دهند، تعریف شده و پیاده‌سازی می‌شود. **مؤلفه‌ها** می‌توانند درونی باشند که در این صورت اگر بدون **واسطارائه‌شده** باشند از دیدها پنهان خواهند بود حتی اگر **واسط‌موردنیاز** داشته باشند. این نوع **مؤلفه‌ها** بیشتر در طراحی برای **کپسوله‌کردن** ماژول‌ها استفاده می‌شوند و در درک بهتر موضوع نیز کمک می‌کنند.

نماد یک **مؤلفه** به شکل یک مستطیل است که یا **کلیشه** «component» و یا نماد کوچک شده **مؤلفه** در بالای نام **مؤلفه**، آن را نمایش می‌دهند.



مانند **نمودارمؤلفه**، **مؤلفه** نیز از **مصنوعات** و عناصری تشکیل شده و یا تجلی می‌یابد. این **مصنوعات** و عناصر شامل موارد زیر هستند:

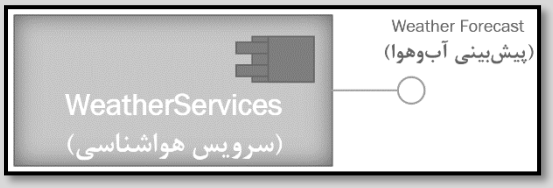
- مؤلفه محقق‌کننده
- قطعه محقق‌کننده
- اتصال‌دهنده
- درگاه

- واسط‌ارائه‌شده
- واسط‌موردنیاز

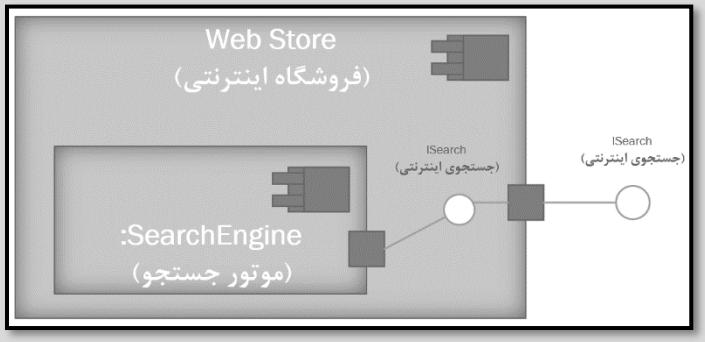
در این بخش فقط **واسط‌های‌ارائه‌شده** و **واسط‌های‌موردنیاز** توضیح داده می‌شود.

واسط‌های‌ارائه‌شده می‌توانند با یکی از روش‌های زیر تهیه شوند:

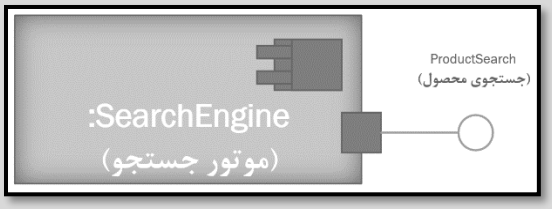
- به طور مستقیم توسط خود **مؤلفه**



- توسط یکی از **طبقه‌بندی‌کننده‌های** محقق کننده **مؤلفه**

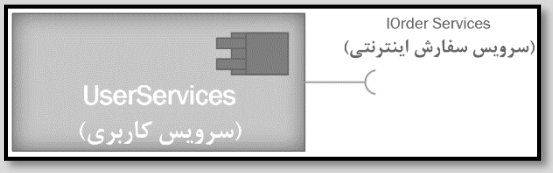


- توسط یک **درگاه** عمومی **مؤلفه**

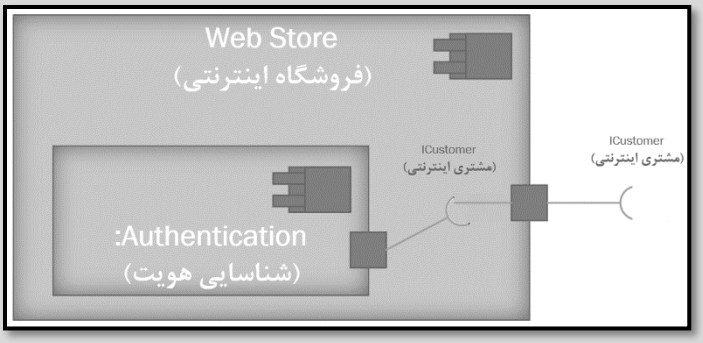


واسط‌های‌موردنیاز می‌توانند با یکی از روش‌های زیر تهیه شوند:

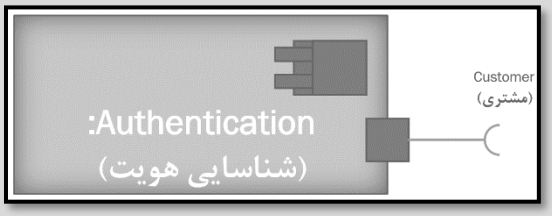
- توسط **وابستگی نقطه‌به‌نقطه یک‌طرفه کاربرد** از خود **مؤلفه**



- توسط **وابستگی نقطه‌به‌نقطه یک‌طرفه کاربرد** از یکی از **طبقه‌بندی‌کننده‌های** محقق کننده **مؤلفه**



- توسط یک **درگاه** عمومی **مؤلفه**



برخی مواقع بجای نمایش **واسط‌های** یک **مؤلفه** به صورت نماهای نمایش داده شده در بالا؛ می‌توانید **نمای‌خارجی‌مؤلفه** را به شکل زیر نمایش دهید:

برخی مواقع برای اینکه نمادهای **مؤلفه** و یا **نمودارهای‌مؤلفه** با توضیحات بیشتری همراه باشند این توضیحات را در قالب **کلیشه‌های** استاندارد نمایش می‌دهند. این **کلیشه‌ها** شامل موارد زیر هستند. توجه داشته باشید که اگر از این **کلیشه‌ها** استفاده می‌کنید، دیگر از **کلیشه** «component» استفاده نکنید.

- **کلیشه** «BuildComponent» که برای **مؤلفه‌هایی** استفاده می‌شود که در **نمودارهای‌مؤلفه** مربوط به فرآیند توسعه وجود خواهند داشت و در خصوص امکانات سیستم در حال توسعه طرحی را ارائه نمی‌کنند، بلکه فضای توسعه و تیم توسعه را شرح می‌دهند.

- **کلیشه** «Entity» که **مؤلفه‌ای** معرفی می‌کند که باید مقادیر اطلاعاتی ثابت، تهیه‌شده یا مورد نیاز را در خود نگاه دارد و یا در اختیار سایر **مؤلفه‌ها** قرار دهد. این **کلیشه** در یک **مؤلفه** نشان می‌دهد که این **مؤلفه** شامل **کلاس** غیرفعال (**کلاسی** که **شیء** آن به خودی خود نمی‌تواند آغازگر یک تعامل باشد) است. شاید به نظر برسد که این موضوع می‌تواند در قالب یک **قطعه** یا **طبقه‌بندی‌کننده** نیز وجود داشته باشد؛ بله همین‌طور است ولیکن **مؤلفه** باعث می‌شود که ارائه این اطلاعات توسط **واسط‌های** مشخصی ارائه شود.

- **کلیشه** «Implement» که **مؤلفه‌ای** معرفی می‌کند که طراح ،تعریف و یا **مشخصه** آن را به صورت مستقل ارائه نمی‌نماید. **مؤلفه‌هایی** که با این **کلیشه** معرفی می‌شوند، این انتظار را ایجاد می‌کنند که یک یا چند **مؤلفه** یا **طبقه‌بندی‌کننده** دیگر با **کلیشه** «Specification» باید **مشخصه‌های** آنها را معرفی کنند و این **مؤلفه** با این **کلیشه** باید فقط نوعی پیاده‌سازی برای آن محقق نماید.

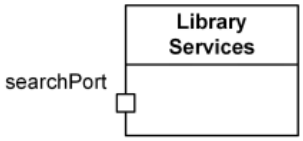
- **کلیشه** «Realization» که **مؤلفه‌ای** معرفی می‌کند که تنها دارای **طبقه‌بندی‌کننده‌های** واقعی است که رفتار مشخص‌شده توسط **مؤلفه** «Specification» جداگانه را پیاده‌سازی و محقق می‌کنند.

- **کلیشه** «Specification» که **مؤلفه‌ای** معرفی می‌کند که تنها دارای معرفی **واسط‌های‌ارائه‌شده و واسط‌های‌موردنیاز** است، و پیش‌بینی نشده که **طبقه‌بندی‌کننده‌ای** بخشی از آن را تعریف کند. این نوع **مؤلفه** فقط یک دید کلی بدون پیاده‌سازی از **امکانات** یک **مؤلفه** در اختیار قرار می‌دهد.
- **کلیشه** «Subsystem» که **مؤلفه‌ای** معرفی می‌کند که واحد تجزیه سلسله مراتبی را برای سیستم‌های بزرگ نشان می‌دهد و برای مدل‌سازی **مؤلفه‌های** مقیاس بزرگ استفاده می‌شود. این **مؤلفه‌ها** در **نمودارمدل** استفاده می‌شوند.
- **کلیشه** «Process» که یک **مؤلفه** مبتنی بر تراکنش را معرفی می‌کند که بنا به تعریف باید statfull باشد به عبارت دیگر یک جریان چندمرحله‌ای را معرفی می‌کند. از منظری می‌توان گفت که یک **مؤلفه** با **کلیشه** «Subsystem» می‌تواند شامل مجموعه‌ای از **مؤلفه** با **کلیشه** «Process» باشند.
- **کلیشه** «Service» که یک **مؤلفه** عملکردی stateless را معرفی می‌کند. از منظری می‌توان گفت که یک **مؤلفه** با **کلیشه** «Subsystem» می‌تواند از مجموعه‌ای از **مؤلفه** با **کلیشه** «Service» استفاده نماید.

درگاه

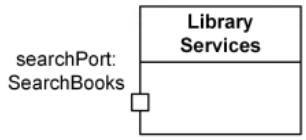
یک **درگاه**، نقطه تعاملی را مشخص می‌کند که از طریق آن یک **طبقه‌بندی‌کننده** می‌تواند با محیط خود، با سایر **طبقه‌بندی‌کننده‌ها** یا با بخش‌های داخلی خود ارتباط برقرار کند. **طبقه‌بندی‌کننده کپسوله‌شده** در UML به‌عنوان **طبقه‌بندی‌کننده ساخت‌یافته** با قابلیت داشتن **درگاه**، تعریف می‌شود و بنابراین **درگاه** یکی از **مشخصات** یک **طبقه‌بندی‌کننده کپسوله‌شده** است. **درگاه** به طور پیش‌فرض دارای **قابلیت‌نمایش** عمومی (public) است.

درگاه به شکل یک مربع کوچک نشان داده می‌شود که یا بر روی محیط مستطیل که نشان دهنده **طبقه‌بندی‌کننده** محصورشده است، به صورت همپوشانی شده قرار دارد، یا ممکن است در داخل آن نماد مستطیل نشان داده شود. نام **درگاه** در نزدیکی نماد آن قرار خواهد گرفت.



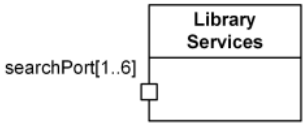
کلاس Library Services دارای درگاه searchPort است

در حالی که UML قرارداد نامگذاری را برای **درگاه‌ها** دیکته نمی‌کند، عقل سلیم حکم می‌کند که نام **درگاه‌ها** را از حروف کوچک شروع کنید، به عنوان مثال. "p"، "port12"، "searchPort". مشخص نیست که آیا هر **درگاه** باید یک نام داشته باشد یا خیر. اگر **درگاه** الزامی برای داشتن یک نام نداشته باشد، ممکن است در یک نمودار حذف شود. مشخصات UML 2.5 توضیح عجیبی ارائه می‌دهد که "هر تصویر از یک **درگاه** بی‌نام، نشان دهنده یک **درگاه** متفاوت از هر **درگاه** دیگری است." اگر نام یک **درگاه** به سادگی حذف شود، هر تصویری از آن **درگاه** باید همچنان همان **درگاه** باشد. نوع **درگاه** ممکن است بعد از نام **درگاه** نشان داده شود که با علامت دونقطه ":" از هم جدا شده است.



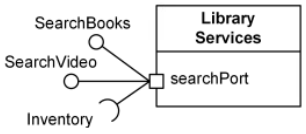
کلاس Library Services دارای درگاه searchPort با نوع SearchBooks است

تعدد یک **درگاه** (در صورت وجود) بعد از نام **درگاه** در براکت‌ها "[]" نشان داده می‌شود. هم نام و هم تعدد **درگاه**، اختیاری است.



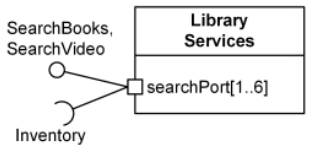
کلاس Library Services دارای 1 تا 6 درگاه searchPort است

یک **واسطارائه‌شده** ممکن است با استفاده از نمادی شبیه به "آب نبات چوبی" متصل به **درگاه**، نشان داده شود. یک **واسطموردنیاز** ممکن است با استفاده از نماد "سوکت" متصل به **درگاه**، نشان داده شود.



درگاه searchPort با واسط‌های‌ارائه‌شده SearchBooks و SearchVideo و واسطموردنیاز Inventory همراه است

درگاه searchPort، در **کلاس** Library Services کاملاً محصور شده است و می‌توان آن را بدون هیچ گونه آگاهی از محیطی که **کلاس** Library Services در آن جاسازی می‌شود، پیاده‌سازی کرد. اگر چندین **واسط** مرتبط با یک **درگاه** وجود داشته باشد، این **واسط‌ها** ممکن است با علامت کاما ”,“ در نزدیکی یک نماد **واسط** واحد، فهرست شوند.

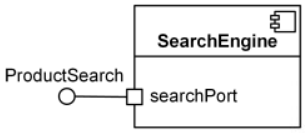


درگاه searchPort با واسط‌های‌ارائه‌شده SearchBooks و SearchVideo و واسط‌موردنیاز Inventory همراه است

درگاه همچنین ممکن است به عنوان یک نماد مربع کوچک نشان داده شود که روی محیط نماد مستطیل همپوشانی دارد و قسمتی را نشان می‌دهد که توسط آن **طبقه‌بندی‌کننده** تایپ شده است.

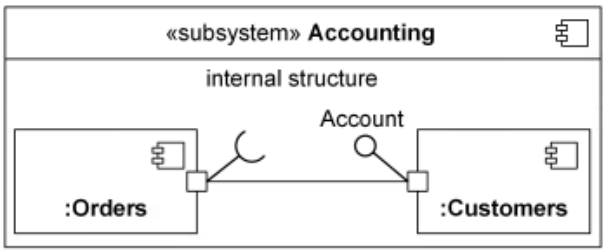
درگاه‌ساده

درگاه‌ساده، **درگاهی** است که دارای یک **واسط‌موردنیاز** یا **واسط‌ارائه‌شده** باشد. یک **درگاه‌پیچیده** دارای چندین **واسط‌ارائه‌شده** و/یا **واسط‌موردنیاز** است.



مؤلفه SearchEngine دارای درگاه‌ساده searchPort با واسط‌ارائه‌شده ProductSearch است

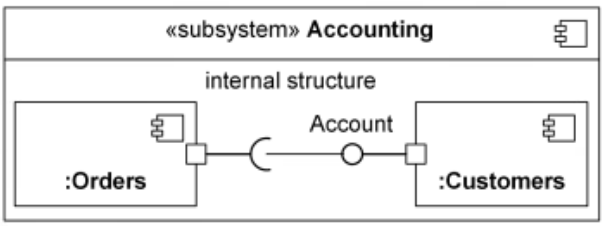
در یک **طبقه‌بندی‌کننده ساخت‌یافته**، UML چندین نماد جایگزین برای اتصال **درگاه‌های‌ساده** روی **قطعات** و **نقش‌ها** اجازه می‌دهد. تنها علامت اجباری برای اتصال **درگاه‌ها** در **ساختارداخلی**، زمانی است که **اتصال‌دهنده‌ها** مستقیماً به **درگاه‌ها** متصل می‌شوند. نمادهای اختیاری آب نبات چوبی و سوکت می‌تواند **واسط‌های‌ارائه‌شده** و **واسط‌های‌موردنیاز درگاه‌های** متصل را نشان دهند.



درگاه‌های‌ساده که مستقیماً توسط اتصال‌دهنده‌ها به هم متصل می‌شوند، نماد UML اجباری هستند.

قطعه مؤلفه Customers واسط‌ارائه‌شده Account را به قطعه مؤلفه Orders ارائه می‌دهد.

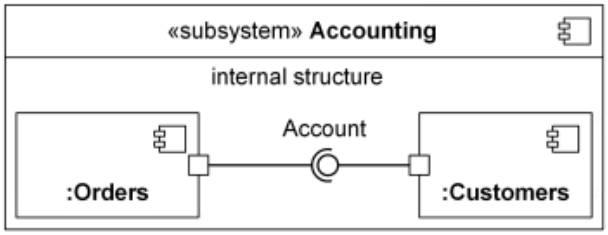
به عنوان یک گزینه، UML اجازه می‌دهد تا خط اتصال به جای **درگاه‌ها**، بین **واسط‌ارائه‌شده** و **واسط‌موردنیاز** متصل شود، همانطور که در زیر نشان داده شده است.



واسط‌ارائه‌شده و واسط‌موردنیاز توسط اتصال‌دهنده به هم وصل شده‌اند، نماد اختیاری UML.

قطعه مؤلفه Customers واسطارائه‌شده Account را به قطعه مؤلفه Orders ارائه می‌دهد.

اتصال‌دهنده برای سرهم‌کردن **واسطارائه‌شده** و **واسط‌موردنیاز** در **درگاه‌های‌ساده** اختیاری است. این نماد نباید برای اتصال **درگاه‌های‌پیچیده** یا **قطعات** بدون **درگاه** استفاده شود.



اتصال‌دهنده سرهم‌کننده واسطارائه‌شده و واسط‌موردنیاز درگاه‌های‌ساده، نماد اختیاری UML.

قطعه مؤلفه Customers واسطارائه‌شده Account را به قطعه مؤلفه Orders ارائه می‌دهد.

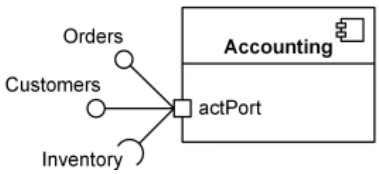
درگاه‌سرویس

یک **درگاه** ممکن است سرویسی را که یک **طبقه‌بندی‌کننده کپسوله‌شده** به محیط خود ارائه می‌دهد و یا سرویسی که یک **طبقه‌بندی‌کننده کپسوله‌شده** از محیط خود نیاز دارد، مشخص کند. هر **درگاه** به طور پیش‌فرض **درگاه‌سرویس** است که به طور پیش‌فرض مقدار **مشخصه** isService **درگاه** با true مشخص می‌شود.

واسط‌های‌ارائه‌شده یک **درگاه**، درخواست‌هایی را برای **طبقه‌بندی‌کننده** توصیف می‌کنند که سایر **طبقه‌بندی‌کننده‌ها** ممکن است از طریق این **درگاه** پاسخ دهند. **واسط‌های‌موردنیاز** یک **درگاه**، درخواست‌هایی را که ممکن است از طریق **درگاه طبقه‌بندی‌کننده** مورد نظر به محیط آن ارسال شود، توصیف می‌کنند.

هنگامی که **مشخصه** isService یک **درگاه** با false تنظیم می‌شود، به این معنی است که این **درگاه** متعلق به پیاده‌سازی **طبقه‌بندی‌کننده کپسوله‌شده** است و بخشی از قرارداد سرویس (SLA) یا عملکرد قابل‌مشاهده خارجی نیست. چنین **درگاه‌غیرسرویسی** را می‌توان بدون هیچ تأثیری بر قرارداد سرویس، اصلاح یا حذف کرد. اگر بتوانیم برفرض، **قابلیت‌نمایش** عمومی (public) پیش‌فرض **درگاه** را تغییر دهیم، مشخصات UML هیچ توضیحی برای اینکه چرا به این **مشخصه** نیاز است ارائه نمی‌دهد. **قابلیت‌نمایش** خصوصی (private) یک **مشخصه** مانع مشاهده آن توسط **قطعه** یا بخشی از هیچ عملکرد خارجی می‌شود.

UML هیچ علامت خاصی برای تشخیص بصری **درگاه‌های‌سرویس** از **درگاه‌های‌غیرسرویس** ارائه نمی‌دهد، در حالی که دارای قراردادی برای ارائه **مشخصات** Boolean به عنوان امکان **اصلاح‌کننده** است. به عنوان مثال، {ordered} به این معنی است که **مشخصه** 'isOrdered' مقدار true دارد، در حالی که {unordered} به این معنی است که **مشخصه** مورد نظر، مرتب نشده است. احتمالاً می‌توانیم از همان قرارداد برای **مشخصه** isService **درگاه** استفاده کنیم و آن را به صورت {service} یا {nonservice} قابل مشاهده کنیم.

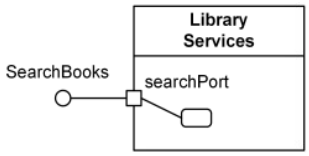


درگاه actPort به طور پیش‌فرض یک درگاه‌سرویس است که واسط‌های‌ارائه‌شده Orders و Customers و واسط‌موردنیاز Inventory را تهیه می‌کند

درگاه‌رفتار

درگاه‌رفتار، درگاهی است به گونه‌ای که درخواست‌هایی که به این **درگاه** می‌رسند به جای ارسال به برخی از نمونه‌های موجود، به **رفتار طبقه‌بندی‌کننده** صاحب **درگاه** ارسال می‌شوند. اگر هیچ **رفتاری** برای این **طبقه‌بندی‌کننده** تعریف نشده باشد، هرگونه ارتباطی که به **درگاه‌رفتار** می‌رسد از بین می‌رود. به طور پیش‌فرض، **درگاه‌ها**، **درگاه‌های‌رفتاری** نیستند.

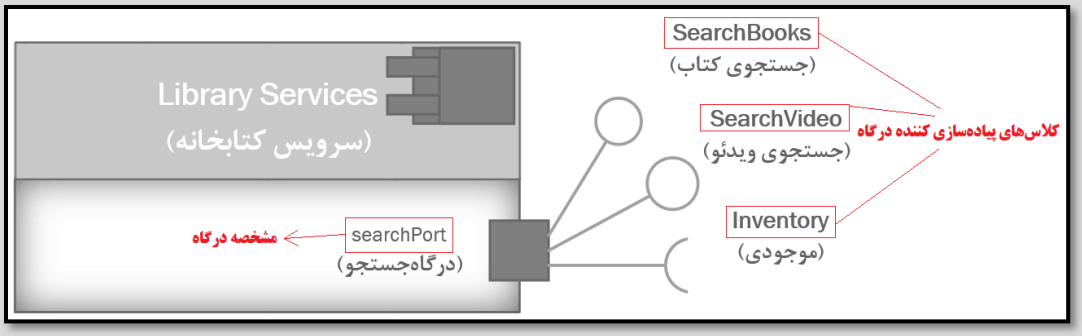
یک **درگاه‌رفتار** به عنوان یک **درگاه** ارائه می‌شود که توسط یک خط ثابت به یک نماد **وضعیت** کوچک که در داخل **طبقه‌بندی‌کننده** شامل **درگاه**، کشیده شده است، متصل می‌شود. نماد **وضعیت** کوچک، **رفتار طبقه‌بندی‌کننده** شامل **درگاه** را نشان می‌دهد.

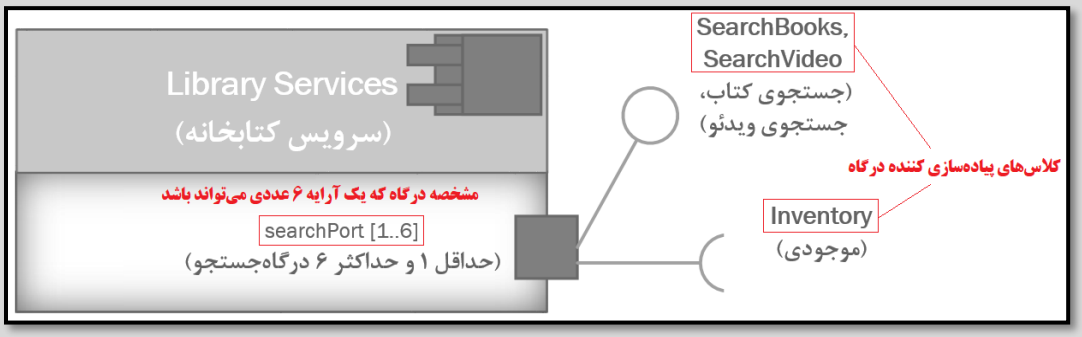


درگاه searchPort درگله‌رفتاری با واسط SearchBooks است

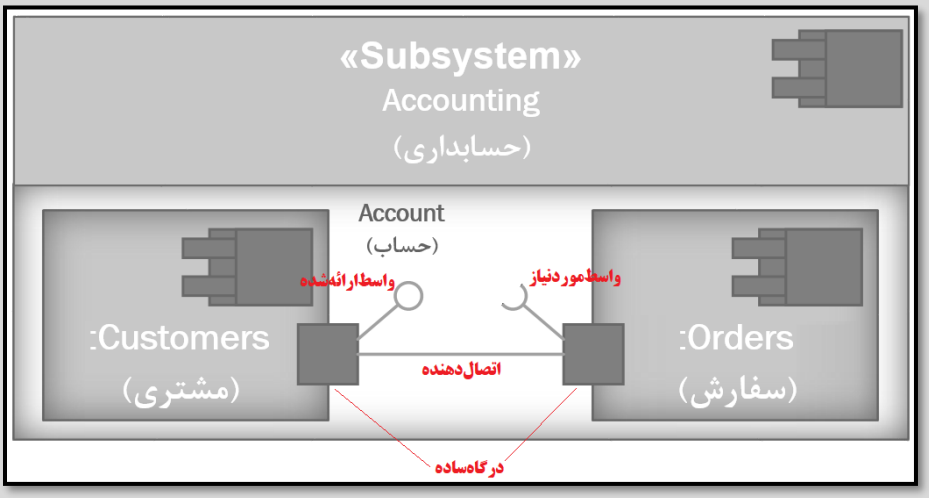
خلاصه و چکیده – Port

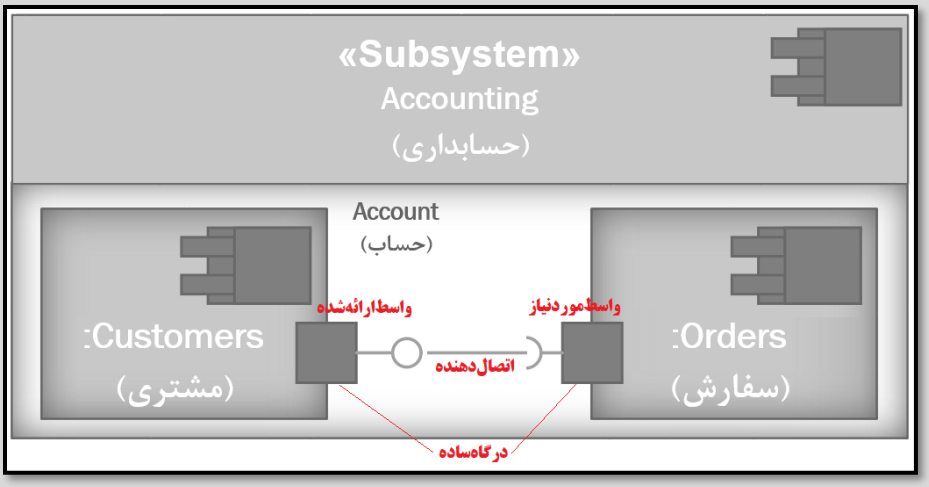
یک **درگاه** یک **مشخصه** از یک **طبقه‌بندی‌کننده** است که به واسطه آن می‌تواند با محیط اطراف خود ارتباط داشته باشد. این **مشخصه** می‌تواند دارای پیاده‌سازی‌هایی باشد که به آنها **واسط‌های‌ارائه‌شده** و **واسط‌های‌موردنیاز** می‌گویند. **مشخصه‌ای** به نام یک **درگاه** همواره باید با **قابلیت‌نمایش** عمومی (public) معرفی شود. با توجه به اینکه کدام سطح از طراحی مشخص شده است می‌توانید **درگاه‌ها** را به اشکال زیر معرفی کنید:

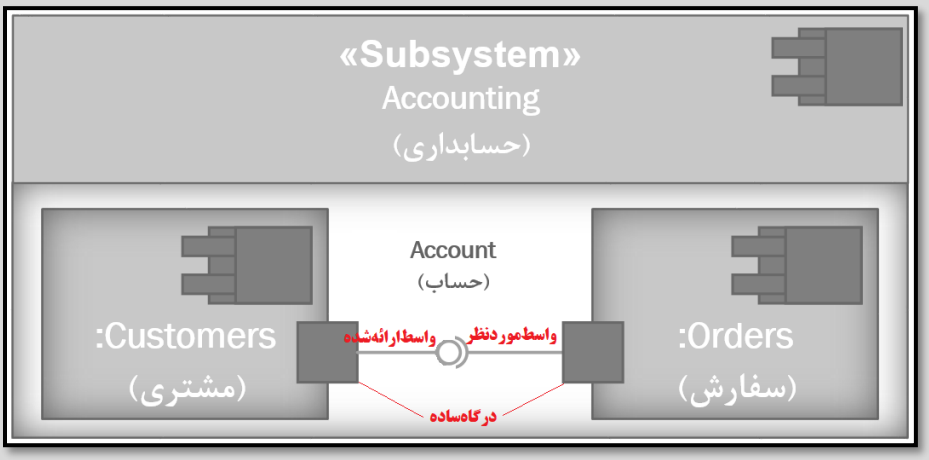




درگاهی که فقط با یک **واسط**، پیاده‌سازی شود؛ فارق از اینکه این **واسط** یک **واسط‌موردنیاز** باشد یا **واسط‌ارائه‌شده**، یک **درگاه‌ساده** است. **درگاه‌های‌ساده** می‌توانند با استفاده از **اتصال‌دهنده‌ها** به هم مرتبط شوند. این ارتباط می‌تواند به سه شکل زیر نمایش داده شود.







همانطور که در بخش‌های دیگر هم شرح داده شده است، **درگاه‌ها** فقط مختص **مؤلفه‌ها** نیستند، بلکه این عنصر UML برای **طبقه‌بندی‌کننده‌ها** نیز استفاده می‌شود. اگر **درگاهی** داشته باشیم که در یک **طبقه‌بندی‌کننده کپسوله‌شده** مورد استفاده قرار گیرد؛ یک **درگاه‌سرویس** نام دارد و احتمال دارد برای **طبقه‌بندی‌کننده** مورد نظر از **کلیشه** «Service» نیز استفاده شود. البته همانطور که قبلا گفته شد **مؤلفه‌ها** نیز تجلی از **سرویس‌ها** هستند؛ به همین دلیل **درگاه‌های** مربوط به **مؤلفه‌ها**، همه از نوع **درگاه‌سرویس** هستند. یک **درگاهی** در صورتی که مشخصه isService آن مقدار true داشته باشد، **درگاه‌سرویس** است. برای اینکه به صورت مستقیم در نمودار خود نشان دهید که **درگاه** مورد نظر یک **درگاه‌سرویس** یا **درگاه‌غیرسرویس** است می‌توانید از عبارت {service} و یا {nonservice} در کنار نام **درگاه** استفاده کنید. اگر نام **درگاهی** با عبارت {nonservice} همراه باشد به معنی این است که از خارج **طبقه‌بندی‌کننده کپسوله‌شده** یا **مؤلفه** قابل مشاهده نیست و بدون اینکه در کل مشکلی برای امکانات ارائه شده و یا قرارداد سرویس خللی وارد شود می‌توان این **درگاه** را حذف کرد.

واسط‌های‌ارائه‌شده یک **درگاه**، درخواست‌های **طبقه‌بندی‌کننده** را توصیف می‌کنند که سایر **طبقه‌بندی‌کننده‌ها** ممکن است از طریق این **درگاه** پاسخ دهند. **واسط‌های‌موردنیاز** یک **درگاه**، پاسخ‌های **طبقه‌بندی‌کننده** را توصیف می‌کنند که سایر **طبقه‌بندی‌کننده‌ها** ممکن است از طریق این **درگاه** دریافت کنند.

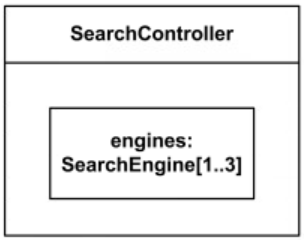
تا به حال در مورد **رفتارها** و **وقایع** بیرون **طبقه‌بندی‌کننده** یا **مؤلفه** در محل **درگاه** وابسته صحبت کردیم ولی در خصوص **رفتارها** و یا **وقایع** درون **طبقه‌بندی‌کننده** یا **مؤلفه** صحبتی نکردیم. اگر در نماد **طبقه‌بندی‌کننده** یا **مؤلفه** به همراه نماد **وضعیت** نمایش داده شود مانند تصویر زیر می‌تواند توضیح بیشتری در خصوص **درگاه** ارائه نماید. این **درگاه** را **درگاه‌رفتار** می‌گویند. همانطور که به نظر می‌رسد، **درگاه‌ها** به صورت پیش‌فرض **درگاه‌سرویس** خواهند بود ولی به صورت پیش‌فرض **درگاه‌رفتار** نخواهند بود.

قطعه

مشخصه‌ای (از ساختارهای داخلی) **زیرکلاسی** از **مشخصات** هسته است و مجموعه‌ای از نمونه‌ها را نشان می‌دهد که متعلق به یک نمونه دربرگیرنده یک **طبقه‌بندی‌کننده** هستند. همچنین یک عنصر قابل اتصال (از ساختارهای داخلی) است.

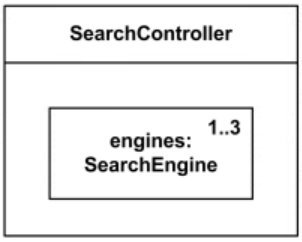
قطعه، مشخصه‌ای است که توسط **طبقه‌بندی‌کننده** با استفاده از ارتباط **ترکیب** وجود خواهد داشت. این بدان معناست که وقتی نمونه **طبقه‌بندی‌کننده** دربرگیرنده، از بین می‌رود، همه **قطعات** از بین می‌روند.

یک **قطعه** با تودرتوی گرافیکی یک نماد جعبه مانند با یک طرح کلی که نمایانگر **قطعه** در یک محفظه جداگانه در نماد **طبقه‌بندی‌کننده** است، نشان داده می‌شود.



طبقه‌بندی‌کننده SearchController دارای 1 تا 3 engines است که از قطعات SearchEngine تشکیل شده‌اند

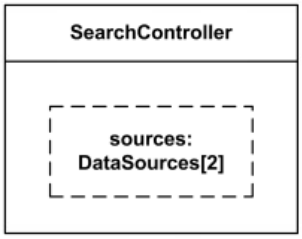
یک **قطعه** یا نماد جعبه مانند **مشخصه** فقط یک محفظه شامل نام دارد که دارای یک رشته حرفی مطابق نگارش تعریف شده برای **مشخصه** هسته‌ای **طبقه‌بندی‌کننده** است. ممکن است جزئیات در این کادر نشان داده شود که مقادیر خاصی را برای **مشخصات** مورد نظر، نشان دهد. تعدد یک **مشخصه** یا **قطعه** ممکن است به شکل علامت تعدد در گوشه سمت راست بالای کادر **مشخصه**، نشان داده شود.



طبقه‌بندی‌کننده SearchController دارای 1 تا 3 engines است که از قطعات SearchEngine تشکیل شده‌اند

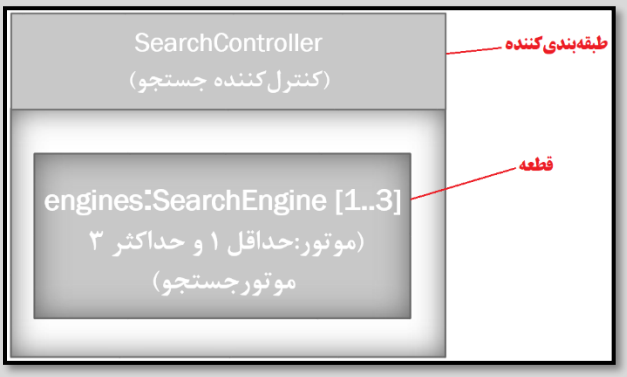
یک نماد **مشخصه** ممکن است حاوی فقط یک نام (بدون دو نقطه) در رشته نام خود نشان داده شود. این به معنای تعریف یک **کلاس** با نام ناشناس است که در **فضای‌نام کلاس** دربرگیرنده تودرتو است.

مشخصه‌ای که نمونه‌ای را مشخص می‌کند که با استفاده از ترکیب متعلق به نمونه **طبقه‌بندی‌کننده** دربرگیرنده نیست، با نماد جعبه تودرتوی گرافیکی با طرح کلی خط‌چین‌دار نشان داده می‌شود.



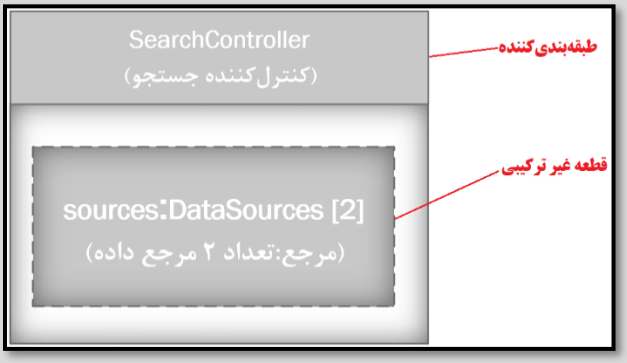
دو DataSource مشخصه sources است که قطعه‌ای از طبقه‌بندی‌کننده SearchController نیست

قطعه، بخشی از **ساختارداخلی** یک **طبقه‌بندی‌کننده** محسوب می‌شود که شاید به آن بتوان عنوان یک **زیرکلاس** داد ولی در حالت عمومی هر **قطعه**، بخشی از **مشخصات هسته‌ای** یک **طبقه‌بندی‌کننده** محسوب می‌شود که می‌تواند با بخش‌های دیگر آن ارتباط برقرار کند. این ارتباط قطعا از نوع ارتباط **نقطه‌به‌نقطه یک‌طرفه از نوع ترکیب** یا همان Composition خواهد بود. همانطور که می‌دانید هر دو انتهای مرتبط در دو طرف یک ارتباط **نقطه‌به‌نقطه یک‌طرفه از نوع ترکیب**، به صورت یک‌طرفه به هم وابستگی وجودی دارند یعنی **انتهای‌ترکیب‌شونده** وجود خود را از **انتهای‌ترکیب‌کننده** دارد و با نابودی **انتهای‌ترکیب‌کننده**، **انتهای‌ترکیب‌شونده** نیز نابود می‌شود.



در تصویر بالا، یک نماد **قطعه** در داخل یک **طبقه‌بندی‌کننده** قرار دارد و می‌تواند شامل 3 نمونه از **کلاس** SearchEngine باشد. هر یک از این نمونه‌ها که یکی از عناصر آرایه []engines خواهند بود یک **قطعه** درونی این **طبقه‌بندی‌کننده** محسوب می‌شوند. از نظر معناشناختی اگر آرایه از بین برود؛ عناصر آرایه نیز از بین می‌روند. همانطور که دیده می‌شود از این منظر نیز این قطعات (عناصر یک آرایه) نیز یک ارتباط **نقطه‌به‌نقطه یک‌طرفه از نوع ترکیب** با هم دارند. بعلاوه اگر **طبقه‌بندی‌کننده** نیز از بین برود این آرایه نیز از بین می‌رود.

بعضی مواقع ممکن است که **قطعه** مورد نظر بخشی از ترکیب کلی **طبقه‌بندی‌کننده** نباشد ولی در **طبقه‌بندی‌کننده** مورد استفاده داشته باشد. در این حالت اگر **طبقه‌بندی‌کننده** از بین برود این **قطعه** از بین نمی‌رود. ممکن است سوال پیش آید که چرا در این نمودار باید به این نوع **قطعه** اشاره شود؟ پاسخ این است که این **قطعات** در درک عملیات یا مفهوم ساختاری **طبقه‌بندی‌کننده** مورد نظر تأثیر دارند و بهتر است که به آنها اشاره شود. این **قطعات** را نیز به صورت یک مستطیل نشان می‌دهیم ولیکن به صورت خط‌چین‌دار ترسیم می‌شوند.



در تصویر بالا، **قطعه** غیر ترکیبی مرجع داده، برای کار این **طبقه‌بندی‌کننده** لازم است ولیکن به دلیل اینکه این **قطعه** دارای **کلاسی** است که به صورت Singleton طراحی شده است، باید به گونه‌ای استفاده شود که با از بین رفتن این **طبقه‌بندی‌کننده**، از بین نرود.

اتصال‌دهنده

اتصال‌دهنده، مشخصه‌ای است که پیوندی را مشخص می‌کند که ارتباط بین دو یا چند نمونه را که **نقش‌هایی** را در یک **طبقه‌بندی‌کننده ساخت‌یافته** ایفا می‌کنند، امکان‌پذیر می‌سازد. این پیوند ممکن است نمونه‌ای از یک ارتباط ساختاری باشد، یا ممکن است نشان‌دهنده امکان ارتباط رفتاری نمونه‌ها باشد، زیرا هویت آنها به واسطه ارسال پارامترهایی که در **متغیرها** یا **اسلات‌ها** نگهداری می‌شود، یا به این دلیل که نمونه‌های ارتباطی هستند، برای همان نمونه، شناخته شده است.

اتصال‌دهنده ممکن است با چیزی به سادگی یک اشاره‌گر یا با چیزی به پیچیدگی اتصال‌شبکه محقق شود. برخلاف ارتباط **وابستگی‌نقطه‌به‌نقطه**، که پیوندهای بین هر نمونه از **طبقه‌بندی‌کننده‌های** مرتبط را مشخص می‌کند، **اتصال‌دهنده‌ها**، پیوندهایی را بین نمونه‌هایی که فقط **قطعات** متصل را نمایش می‌دهند، مشخص می‌کنند.

یک **اتصال‌دهنده** با استفاده از نمادی شبیه به ارتباط **وابستگی‌نقطه‌به‌نقطه**، ارائه می‌شود. connector-label که یک مقدار اختیاری است از نگارش زیر پیروی می‌کند:

connector-label ::= [connector-name] [':' (association-name | association-class-name)]

در نگارش بالا connector-name نام **اتصال‌دهنده** است، association-name نام ارتباط **وابستگی‌نقطه‌به‌نقطه** و association-class-name نام **کلاس** این ارتباط **وابستگی‌نقطه‌به‌نقطه** است. ممکن است یک کلمه کلیدی یا **کلیشه** در براکت‌ها در بالا یا جلوی connector-name در یک نمودار قرار گیرد. یک رشته **مشخصه** ممکن است بعد یا زیر connector-name قرار گیرد.

اتصال‌دهنده‌های پیوند دهنده **مؤلفه‌ها** می‌توانند یکی از موارد زیر باشند:

- **اتصال‌دهنده نماینده**
- **اتصال‌دهنده اجتماع**

یک **اتصال‌دهنده نماینده**، نوع ویژگی **اتصال‌دهنده** مشتق شده است از یک **اتصال‌دهنده** با یک یا چند انتهای متصل به **درگاهی** که روی یک **قطعه** نیست و همچنین **درگاه‌رفتاری** نیست. و **اتصال‌دهنده‌هایی** که غیر از این باشند **اتصال‌دهنده اجتماع** هستند.

قرارداد **اتصال‌دهنده**، مجموعه‌ای از **رفتارهایی** است که الگوهای تعامل معتبر را برای کل **اتصال‌دهنده** مورد نظر، مشخص می‌کند.

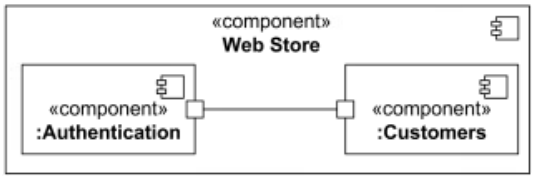
اتصال‌دهنده اجتماع

یک **اتصال‌دهنده اجتماع** یک اتصال بین دو یا چند **قطعه** یا **درگاه** روی **قطعات** است که تعریف می‌کند که یک یا چند **قطعه، سرویسی** را ارائه می‌دهند که سایر **قطعات** از آن، استفاده می‌کنند.

مفهوم زمان اجرا برای یک **اتصال‌دهنده اجتماع** به این صورت است که **سیگنال‌ها** در امتداد یک نمونه از یک **اتصال‌دهنده** حرکت می‌کنند. **اتصال‌دهنده‌های** متعددی که به و از قسمت‌های مختلف هدایت می‌شوند، یا **اتصال‌دهنده‌های** n-ary که در آن n> 2 باشد، نشان می‌دهد که نمونه‌ای که **سیگنال** را کنترل یا مدیریت می‌کند در زمان اجرا تعیین می‌شود.

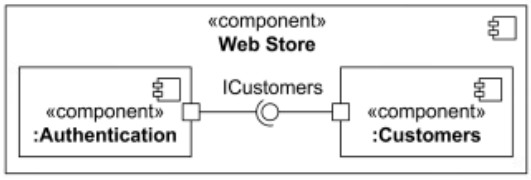
سازگاری **واسط** بین **درگاه‌هایی** که متصل هستند، یک **مؤلفه** موجود در یک سیستم را قادر می‌سازد تا با **قطعه‌ای** جایگزین شود که (حداقل) مجموعه‌ای از **سرویس‌ها** را ارائه می‌دهد. همچنین، در زمینه‌هایی که **مؤلفه‌ها** برای گسترش یک سیستم با ارائه **سرویس** موجود و افزودن عملکرد جدید استفاده می‌شوند، می‌توان از **واسط‌ها** برای پیوند در تعریف **مؤلفه** جدید استفاده کرد.

اتصال‌دهنده اجتماع به عنوان یک **اتصال‌دهنده** بین دو یا چند **قطعه** یا **درگاه** روی **قطعات** مشخص شده است.



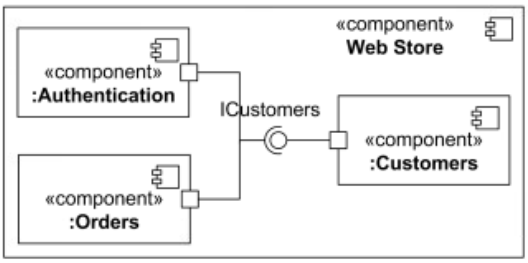
اتصال‌دهنده اجتماع بین درگاه‌های مؤلفه‌های Authentication و Customers

هنگامی که یک **اتصال‌دهنده اجتماع**، **درگاه‌های‌ساده** را به هم متصل می‌کند (**درگاه‌هایی** که یک **واسط‌ارائه‌شده** یا **واسط‌موردنیاز** واحد دارند)، ممکن است با یک اتصال "توپ و سوکت" بین یک **واسط‌ارائه‌شده** و یک **واسط‌موردنیاز** مشخص شود.



اتصال‌دهنده اجتماع بین درگاه‌های‌ساده مؤلفه‌های Authentication و Customers

نماد "توپ و سوکت" را نمی‌توان برای اتصال **درگاه‌های‌پیچیده** یا **قطعات** بدون **درگاه** استفاده کرد. در جایی که چندین **مؤلفه**، دارای **درگاه‌های‌ساده‌ای** هستند که **واسط‌های‌ارائه‌شده** یا **واسط‌های‌موردنیاز** یکسانی دارند، یک نماد **واسط** منفرد می‌تواند نشان داده شود و خطوطی از **مؤلفه‌ها** به آن نماد رسم شود. این حالت ارائه، چه **واسط** با استفاده از نماد "توپ و سوکت" نشان داده شود، یا با استفاده از یک نماد **واسط‌موردنیاز** یا نماد **واسط‌ارائه‌شده**، قابل اجرا است.



اتصال‌دهنده اجتماع که سه قطعه را مونتاژ می‌کند

اتصال‌دهنده نماینده

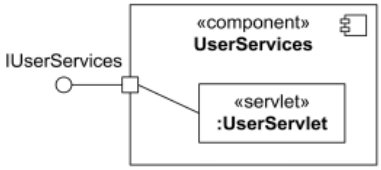
اتصال‌دهنده نماینده، اتصالی است که قرارداد خارجی یک **مؤلفه** (همانطور که توسط **درگاه‌های** آن مشخص شده است) را به تحقق آن **رفتار** مرتبط می‌کند. این نوع از **اتصال‌دهنده** نشان‌دهنده رویدادهای ارسالی (درخواست‌های عملیات و رویدادها) است: **سیگنالی** که به **درگاهی** می‌رسد که دارای یک **اتصال‌دهنده** به یک یا چند **قطعه** یا **درگاه‌های** روی **قطعات** است، برای رسیدگی به آن اهداف ارسال می‌شود.

اتصال‌دهنده نماینده، بیانیه‌ای است مبنی بر اینکه **رفتاری** که در یک نمونه **مؤلفه** موجود است، در واقع توسط خود آن **مؤلفه**، بلکه توسط یک یا چند نمونه که دارای قابلیت‌های "سازگار" هستند، تحقق می‌یابد. این موقعیت‌ها از طریق یک **اتصال‌دهنده** بین یک **درگاه** به **درگاه‌ها** یا **قطعات** سازگار، مدل‌سازی می‌شوند.

اتصال‌دهنده‌های نماینده، را می‌توان برای مدل‌سازی تجزیه سلسله مراتبی **رفتار** مورد استفاده قرار داد، جایی که سرویس‌ارائه‌شده توسط یک **مؤلفه** ممکن است در نهایت توسط **مؤلفه‌ای** که در سطوح تودرتو درون آن قرار دارد، تحقق یابد. کلمه **نماینده**، نشان می‌دهد که **پیام** مشخص و جریان **سیگنال** بین **درگاه‌های** متصل، احتمالاً در چندین سطح، رخ می‌دهد. باید توجه داشت که چنین جریان **سیگنالی** همیشه در همه محیط‌ها یا پیاده‌سازی‌های سیستم تحقق نمی‌یابد (یعنی ممکن است فقط زمان طراحی باشد).

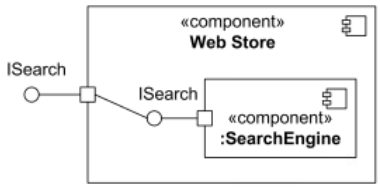
یک **درگاه** ممکن است به مجموعه‌ای از **درگاه‌های مؤلفه** فرعی واگذار شود. در آن صورت، این **درگاه‌های** فرعی باید به طور جمعی عملکرد نمایندگی شده **درگاه** واگذارنده را ارائه دهند. در زمان اجرا، **سیگنال‌ها** به **درگاه** مربوطه تحویل داده می‌شود. در مواردی که چندین **درگاه** هدف، **سیگنال** یکسانی را پشتیبانی می‌کنند، **سیگنال** به همه این **درگاه‌های** فرعی تحویل داده می‌شود.

یک نماد **اتصال‌دهنده‌های نماینده** به شکل یک **اتصال‌دهنده** از **درگاه** اختیاردهنده به **درگاه** یا **قطعه** نماینده، مشخص می‌شود.

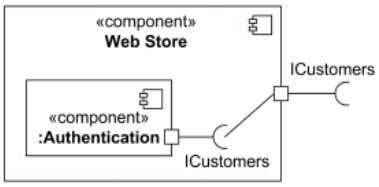


اتصال‌دهنده‌های نماینده از درگاه اختیاردهنده به قطعه UserServlet

اگر نمایندگی مورد نظر توسط یک **درگاه‌ساده** مدیریت شود، ممکن است **اتصال‌دهنده** به صورت اختیاری متصل به یک **واسط‌ارائه‌شده** یا **واسط‌موردنیاز** نشان داده شود.



اتصال‌دهنده‌های نماینده از درگاه اختیاردهنده به درگاه‌ساده SearchEngine



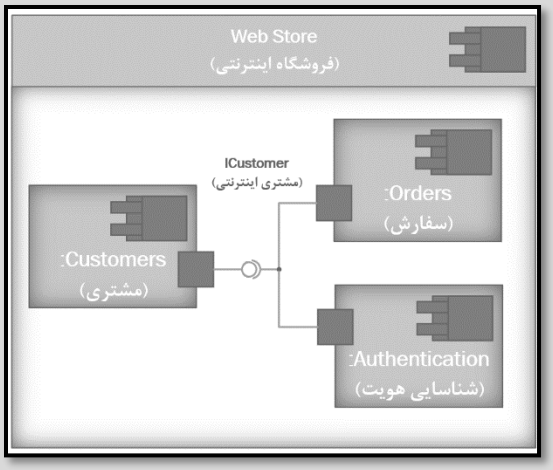
اتصال‌دهنده‌های نماینده از درگاه‌ساده مؤلفه Authentication به درگاه اختیاردهنده

خلاصه و چکیده – Connector

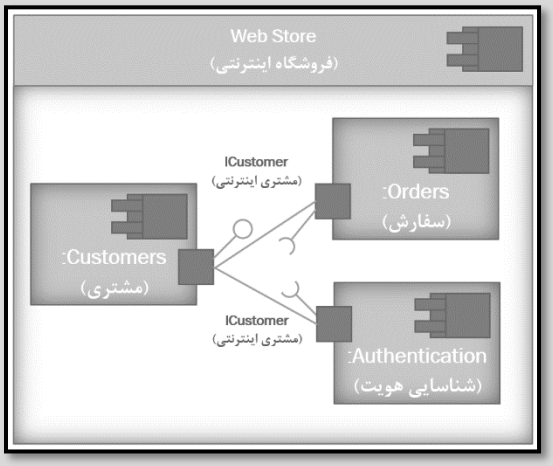
اتصال‌دهنده، یک ارتباط بین **قطعات** یک **طبقه‌بندی‌کننده** را معرفی می‌کند. این ارتباط می‌تواند یک ارتباط ساختاری و یا رفتاری باشد ولی در نمودارهای UML راهی برای تفکیک آنها وجود ندارد. شاید یکی از دلایل نیاز به ترسیم کل نمودارهای متودولوژی UML همین باشد که برای تشخیص بخشی از مفاهیم یک عنصر UML در یک نمودار باید به نمودارهای دیگر مراجعه نمود. مثلا این **اتصال‌دهنده** ممکن است نماینده یک اشاره‌گر باشد که مرجع یک **شیء** مشخص را نگهداری می‌کند (ارتباط ساختاری) و یا نماینده فراخوانی **متودی** از یک **شیء** دیگر (ارتباط رفتاری) باشد.

می‌توانید در زیر نماد **اتصال‌دهنده** از یک برچسب به عنوان معرفی کننده **اتصال‌دهنده** مورد نظر استفاده کنید. همچنین می‌توانید در بالای این برچسب از هر **کلیشه‌ای** که به **اتصال‌دهنده** مورد نظر مرتبط است، استفاده کنید. برای **مؤلفه‌ها** دو نوع **اتصال‌دهنده** وجود دارد:

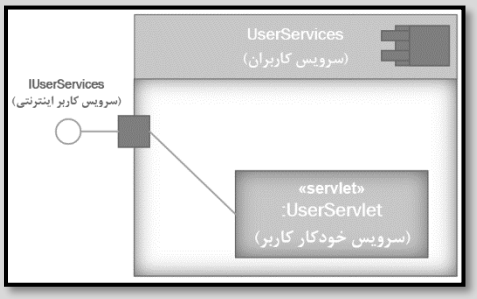
- **اتصال‌دهنده اجتماع** (Assembly Connector) که هر دو طرف این **اتصال‌دهنده** باید **درگاه** باشد (**درگاه‌ها**، عنصری از UML هستند که به **قطعات**، **سرویس‌ها** یا **مؤلفه‌ها** وابسته هستند). نکته‌ای که اینجا مطرح می‌شود، این است که فقط این **درگاه‌ها** نباید **درگاه‌رفتار** باشند.



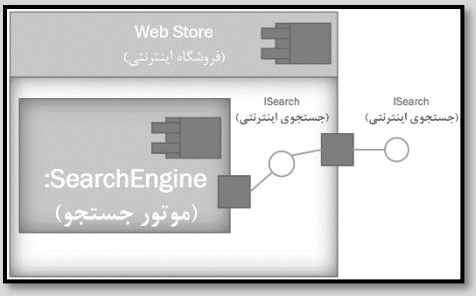
در تصویر بالا، نام **اتصال‌دهنده اجتماع**، ICustomer است. این **اتصال‌دهنده** به صورت ارتباط بین **واسط‌های‌موردنیاز** و **واسط‌ارائه‌شده درگاه‌های** مختلف نمایش داده می‌شود. دقت کنید که دوطرف این **اتصال‌دهنده، درگاه** مربوط به یک **مؤلفه** است که البته می‌توان تصویر بالا را به شکل زیر نیز ترسیم کرد.



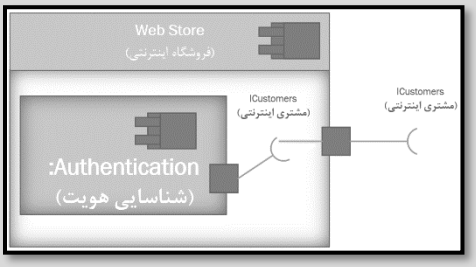
- **اتصال‌دهنده نماینده** (Delegation Connector) که **اتصال‌دهنده‌ای** است که یک‌طرف آن **درگاه** نیست و می‌تواند یک **طبقه‌بندی‌کننده**، یک **رفتار** (**وضعیت**)، یک **واسط‌موردنیاز** و یا یک **واسط‌ارائه‌شده** باشد. این نوع **اتصال‌دهنده** بیشتر برای **رفتارهایی** که به صورت سلسله مراتبی، طراحی و یا پیاده‌سازی می‌شوند، کاربرد دارد.



در تصویر بالا، طراحی و پیاده‌سازی **درگاه مؤلفه**، به یک **قطعه** درونی این **مؤلفه** به نام UserServlet سپرده‌شده است.



در تصویر بالا، **مؤلفه** فروشگاه اینترنتی، **واسط‌ارائه‌شده‌ای** در اختیار محیط پیرامون خود قرار می‌دهد که توسط یک **مؤلفه** درونی به نام موتور جستجو، تهیه شده است. **مؤلفه** موتور جستجو ممکن است در جای دیگری نیز استفاده شود به همین دلیل دارای **درگاه** و **واسط‌ارائه‌شده** است. این **اتصال‌دهنده** در واقع با نقش یک نماینده، این **واسط** را از **درگاه مؤلفه** موتور جستجو به **درگاه مؤلفه** فروشگاه اینترنتی انتقال می‌دهد.



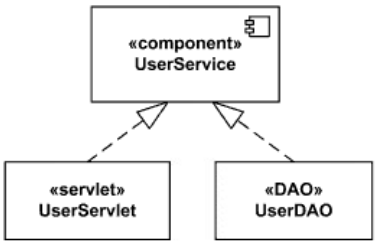
در تصویر بالا، **مؤلفه** فروشگاه اینترنتی، **واسط‌موردنیازی** در اختیار محیط پیرامون خود قرار می‌دهد که توسط یک **مؤلفه** درونی به نام شناسایی هویت، مورد استفاده قرار می‌گیرد. **مؤلفه** شناسایی هویت ممکن است در جای دیگری نیز استفاده شود به همین دلیل دارای **درگاه** و **واسط‌موردنیاز** است. این **اتصال‌دهنده** در واقع با نقش یک نماینده، این **واسط** را از **درگاه مؤلفه** شناسایی هویت به **درگاه مؤلفه** فروشگاه اینترنتی انتقال می‌دهد.

تحقق‌مؤلفه

تحقق‌مؤلفه، **وابستگی‌تحقق** تخصصی است که برای تعریف (اختیاری) **طبقه‌بندی‌کننده‌هایی** استفاده می‌شود که قرارداد ارائه‌شده توسط یک **مؤلفه** را بر حسب **واسط‌های‌ارائه‌شده** و **واسط‌های‌موردنیاز** آن محقق می‌سازند.

رفتار یک **مؤلفه** معمولاً ممکن است توسط تعدادی **طبقه‌بندی‌کننده** معرفی (یا پیاده‌سازی) شود. در واقع یک انتزاع، برای مجموعه‌ای از عناصر مدل تشکیل می‌دهد. در آن صورت، یک **مؤلفه** دارای مجموعه‌ای از **وابستگی‌های‌تحقق مؤلفه** به این **طبقه‌بندی‌کننده‌ها** است.

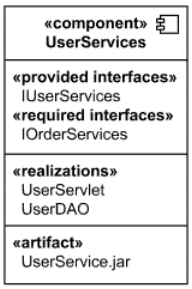
تحقق **مؤلفه** به همان شکل **وابستگی‌تحقق** نشان داده می‌شود، یعنی به شکل یک پیکان خط‌چین‌دار کلی از پیاده‌سازی **طبقه‌بندی‌کننده** تا **مؤلفه** تحقق یافته با مثلث توخالی به عنوان سر این پیکان.



مؤلفه UserService که توسط UserServlet و UserDAO ساخته شده است

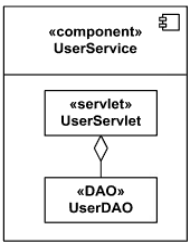
برای برنامه‌هایی که نیاز به مجموعه‌های مختلف تحقق برای یک **مؤلفه** واحد دارند، مجموعه‌ای از **کلیشه‌های** استاندارد در پروفایل استاندارد UML تعریف شده‌اند. به طور خاص، «Specification» و «Realization» برای این منظور تعریف شده است.

مؤلفه را می‌توان با استفاده از نمای داخلی یا نمای "جعبه شفاف" نشان داد که **مشخصات** خصوصی (private) آن را نشان می‌دهد و **طبقه‌بندی‌کننده‌ها** را معرفی می‌کند. این نما، نشان می‌دهد که چگونه **رفتار** بیرونی در داخل **مؤلفه**، تحقق می‌یابد. **طبقه‌بندی‌کننده‌های** واقعی را می‌توان در یک **مؤلفه** با **کلیشه** «realization» اضافی فهرست کرد. همچنین می‌توان از محفظه‌ها برای نمایش فهرستی از **قطعات** و **اتصال‌دهنده‌ها** یا هر مصنوع اجرایی، استفاده کرد.



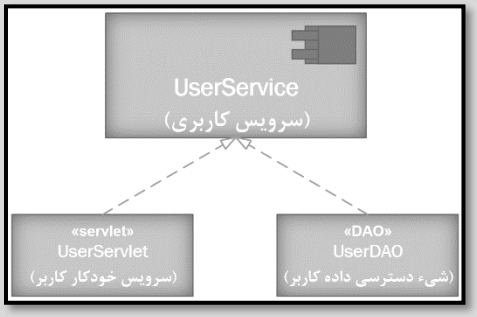
نمای جعبه شفاف مؤلفه UserServices که توسط UserServlet و UserDAO تحقق می‌یابد و توسط مصنوع UserService.jar قابل استفاده می‌شود

از طرف دیگر، **طبقه‌بندی‌کننده‌های** داخلی که **رفتار** یک **مؤلفه** را معرفی می‌کنند ممکن است به صورت تودرتو درون شکل **مؤلفه**، نمایش داده شوند.



نمای جعبه شفاف مؤلفه UserServices که توسط UserServlet و UserDAO تحقق یافته است

رفتار یک **مؤلفه**، ممکن است به واسطه **طبقه‌بندی‌کننده‌های** مختلف تعریف و یا پیاده‌سازی شود. در این حالت ارتباط بین **مؤلفه** مورد نظر و **طبقه‌بندی‌کننده‌های** مورد نظر رابطه **وابستگی نقطه‌به‌نقطه یک‌طرفه تحقق** خواهد بود.



اگر بخواهیم نمودار بالا را کامل کنیم و سپس در کادر شفاف، نمایش دهیم به شکل زیر عمل خواهیم نمود.