کارگاه برنامه‌نویسی پیشرفته

دستور کار شماره هفت

اهداف

آشنایی با مهندسی نرم‌افزار[1]

آشنایی با کارت‌های CRC[2]

آشنایی با UML[3]

[1] Software Engineering
[2] Class-Responsibility-Collaboration Cards
[3] Unified Modeling Language Diagram

فهرست مطالب

آشنایی با مهندسی نرم‌افزار

معرفی مهندسی نرم‌افزار

همان‌طور که تا به حال متوجه شده‌اید، یکی از چالش‌های بزرگی که در پروژه‌های واقعی برنامه‌نویسی با آن سروکار داریم، مهندسی خود نرم‌افزار است. مهندسی نرم‌افزار یک روش مهندسی برای توسعه نظام‌مند نرم‌افزار است. در توسعه نرم‌افزار فعالیت‌های متعددی من جمله طراحی و پیاده‌سازی وجود دارد. در این قسمت تمرکز بر روی طراحی اجزای سامانه است.

با چالش‌های مهندسی نرم‌افزار و روش‌های حل آن بسیار مفصل‌تر در درس مهندسی نرم‌افزار آشنا خواهید شد. اما در این درس قصد داریم که مقداری با راه حل‌های ابتدایی برای این چالش‌ها آشنا شویم.

معرفی چند اصل در مهندسی نرم‌افزار

در نرم‌افزارهای بزرگ و صنعتی لازم است که نرم‌افزار طراحی شده، بتواند علاوه بر به‌روز ماندن، قابل نگهداری باشد. از همین رو در مهندسی نرم‌افزار چندین اصل برای این منظورها به وجود آمده که در اینجا دو تا از مهم‌ترین‌های آن‌ها را با هم مرور می‌کنیم:

- وابستگی کم[1]

وابستگی به معنای مقدار وابسته بودن اجزای مختلف نرم‌افزار به پیاده‌سازی داخل دیگر اجزا است. در طراحی نرم‌افزار سعی داریم نرم‌افزار را به شکلی طراحی کنیم که مقدار وابستگی را به کمترین حد ممکن برسانیم و اجزای نرم‌افزار به صورت مستقل مسئولیت‌های خود را انجام دهند.

- انسجام بالا[2]

انسجام به معنای تعداد وظیفه‌هایی است که هر بخش از نرم‌افزار مسئولیت انجام آن را دارد. اگر یک بخش از نرم‌افزار بیش از یک وظیفه‌ی مشخص داشته باشد، آنگاه نرم‌افزار اصطلاحاً انسجام کمی[3] دارد. در طراحی نرم‌افزار سعی می‌شود که هر یک از اجزای نرم‌افزار یک مسئولیت مشخص داشته باشد.

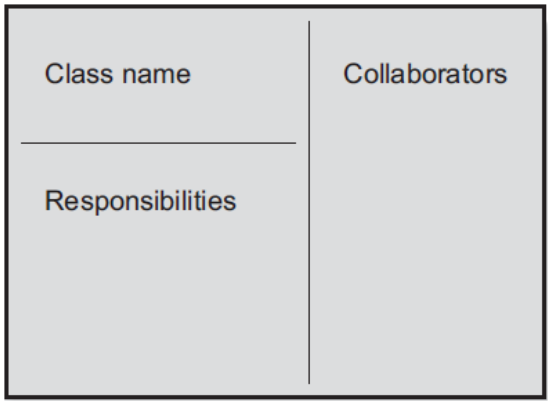
[1] Low Coupling
[2] High Cohesion
[3] Loose Cohesion

آشنایی با کارت‌های CRC

معرفی کارت‌های CRC

کارت‌های CRC یکی از روش‌ها در طراحی نرم‌افزار شیءگرا است. این روش با تحلیل متن پروژه با استفاده از دانش قبلی زبانی طراح سعی می‌کند روشی سامان‌یافته برای طراحی ساختار نرم‌افزار داشته باشد.

هر کارت CRC دارای سه بخش است که در آن‌ها، اسم کلاس، مسئولیت‌های آن و کلاس‌های همکار (کلاس‌هایی که این کلاس با آن‌ها ارتباط دارد) قرار می‌گیرند. شکل کلی یک کارت CRC به شکل زیر است:



(ساختار کلی یک کارت CRC)

مراحل ساخت کارت‌های CRC

برای ساخت کارت‌های CRC ابتدا با استفاده از روش اسم‌ها/فعل‌ها[1]، کلاس‌ها و وظایف آن‌ها را پیدا می‌کنیم و سپس با استفاده از آن‌ها، کارت‌ها را می‌سازیم. در این روش، ابتدا صورت مسئله‌ی داده شده را به دقت بررسی می‌کنیم و در آن، اسم‌ها و فعل‌ها را مشخص می‌کنیم. اسم‌ها نماینده‌ی کلاس‌ها هستند و فعل‌ها وظایف آن‌ها را نشان می‌دهند.

برای درک بهتر، می‌خواهیم با استفاده از این روش، سیستم رزرو بلیت سینما را شبیه‌سازی کنیم. صورت مسئله‌ی داده شده به شکل زیر است: (برای درک بهتر، زیر اسم‌ها خط کشیده شده است و همچنین فعل‌ها قرمز شده‌اند)

سیستم رزرو بلیت سینما باید رزروهای مربوط به چند سالن مختلف را ذخیره کند. هر سالن تعدادی صندلی در ردیف‌های مختلف دارد. مشتریان می‌توانند صندلی رزرو کنند که به آن‌ها شماره صندلی و شماره ردیف در سالن داده می‌شود. مشتری می‌تواند رزرو چندین صندلی مجاور را درخواست کند.

[1] Verb/Noun method

هر رزرو مربوط به یک نمایش خاص است (منظور نمایش یک فیلم در یک زمان خاص است). نمایش‌ها در یک ساعت، تاریخ و سالن مشخص برنامه‌ریزی می‌شوند. سیستم، شماره تلفن مشتری را ذخیره می‌کند.

پس از مشخص شدن کلاس‌ها و وظایف آن‌ها، برای هر کلاس، یک کارت CRC طراحی کرده و وظایف و کلاس‌های همکار آن را در آن کارت ثبت می‌کنیم. به این ترتیب، طرحی کلی از کلاس‌ها و وظایفشان و همچنین نحوه‌ی تعامل آن‌ها با یکدیگر معلوم می‌شود. در ادامه می‌خواهیم با استفاده از CRC کارت‌ها، نمودار کلاس را طراحی کنیم.

معرفی وب‌سایت

برای طراحی راحت‌تر کارت‌های CRC می‌توان از این وب‌سایت استفاده کرد.

دلیل به وجود آمدن UML Class Diagram

همان‌طور که دیدید، کارت‌های CRC تنها نام، وظایف و کلاس‌های همکار یک کلاس را نمایش می‌دهند. اما این‌ها تمام اطلاعات یک کلاس نیستند و علاوه بر آن‌ها نوع فیلدها و نوع پارامترهای متدها و سطح دسترسی فیلدها و متدها و نوع ارتباط کلاس‌ها نیز باید مشخص شود. این در حالی است که کارت‌های CRC توانایی نمایش این موارد را ندارند و به همین دلیل در صورت نیاز به نمایش جزئیات کلاس‌ها از نوع دیگری از نمایش کلاس‌ها استفاده می‌شود که UML Class Diagram نام دارد و شِمای بهتری از پروژه به توسعه دهندگان می‌دهد.

تعریف UML Class Diagram

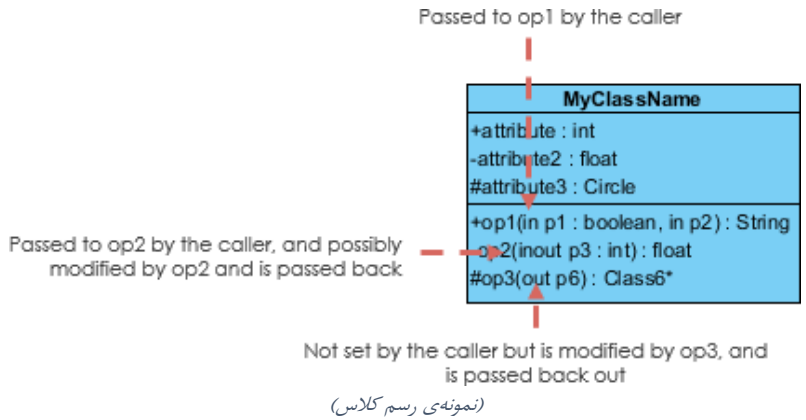
در واقع UML Class Diagram روشی برای رسم نموداری از کلاس‌های یک برنامه و روابط بین آن‌ها است. UML Class Diagram راهی برای رسیدن به شِمای کلی از کلاس‌های یک برنامه با توجه به دستورکار آن است. این روش علاوه بر فیلدهای مربوط به یک کلاس، رفتارهای (متدها) هر کلاس را نیز نمایش می‌دهد. همچنین این نوع نمودار روابط بین کلاس‌های مختلف و سطح دسترسی‌های مربوط به اعضای کلاس و نوع فیلدها و پارامترهای متدها را نیز نمایش می‌دهد.

نحوه‌ی رسم UML Class Diagram

برای رسم نمودار، ابتدا باید با نحوه‌ی رسم اجزای نمودار (کلاس‌ها و روابط بین آن‌ها) آشنا شویم:

نحوه رسم کلاس‌ها

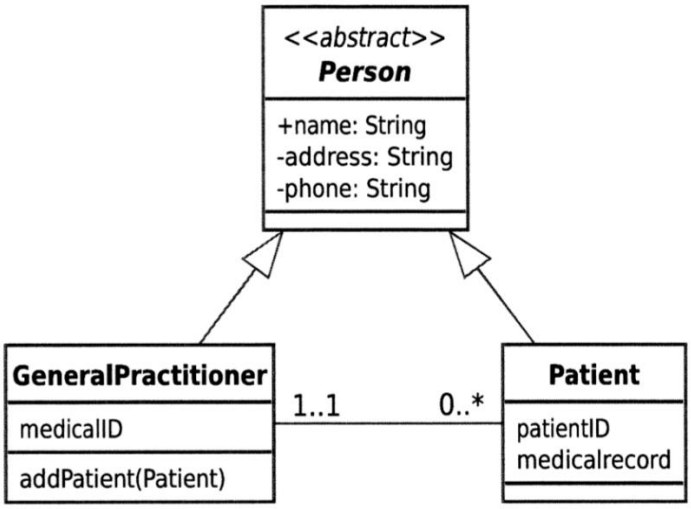
برای رسم کلاس‌ها باید مستطیلی کشید که دارای سه بخش است و از بالا به پایین مقادیر «نام کلاس» و «فیلدها» و «متدها» در آن قرار دارد:



(نمونه‌ی رسم کلاس)

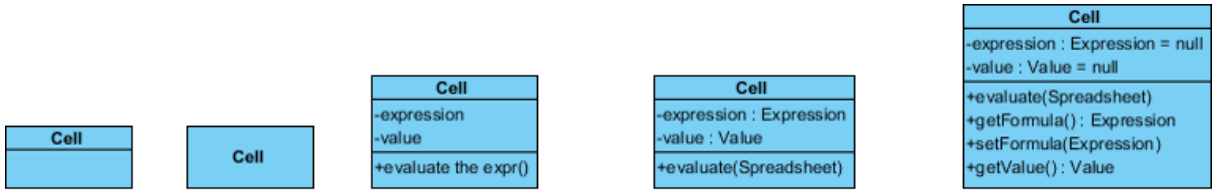
چند نکته:

- برای کلاس‌های انتزاعی[1] و اینترفیس‌ها[2] به ترتیب می‌توان از تگ‌های «abstract» و «interface» در قسمت نام کلاس‌ها استفاده کرد. همچنین نام کلاس انتزاعی به صورت کج[3] نوشته می‌شود.



- سطح دسترسی با علائم + و - و # نشان داده می‌شوند:
  - private: -
  - public: +
  - protected: #
- مسیر پارامترها در واقع به معنی تغییرات و برگردانده شدن آنها در متدها است که با کلمات زیر بیان می‌شوند:
  - In: آرگومان متد از طرف فراخواننده متد پاس داده می‌شود.
  - Inout: آرگومان از طرف فراخواننده به متد پاس داده می‌شود و پس از اعمال فرآیندهای متد، برگردانده می‌شود.
  - Out: پارامتر از متد به فراخوانندهی آن برگردانده می‌شود.

در مثال زیر کلاس Cell با توجه به پیشرفت کار دارای جزئیات بیشتر و کامل‌تری شده است:

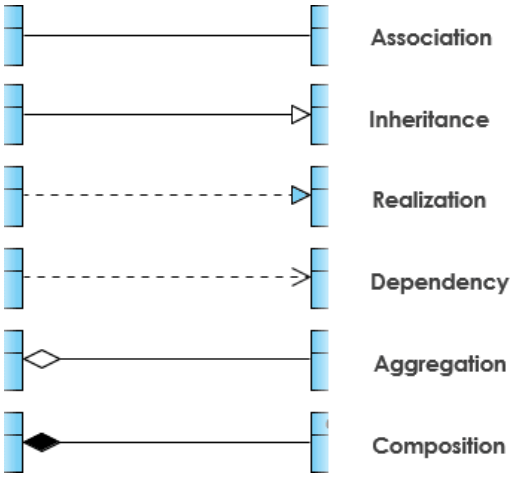


(نمودار UML کلاس Cell)

[1] Abstract Classes
[2] Interfaces
[3] Italic

روابط بین کلاس‌ها

نمودارهای بین کلاس‌ها بیانگر روابط بین آن‌ها می‌باشند که در تصویر زیر به مهم‌ترین آن‌ها اشاره شده است:



(نمودار بین کلاس‌ها)

توضیح نمودارها

- **Assosiation**: این رابطه به همکاری و ارتباط بین دو کلاس اشاره می‌کند (رابطه میان استاد و دانشجو).
- **Inheritance**: در این رابطه، کلاس ابتدای پیکان از کلاس انتهای آن ارث‌بری می‌کند (رابطه میان سگ و حیوان).
- **Realization**: این رابطه برای نمایش اینترفیس و کلاس پیاده‌سازی کننده‌ی آن است. کلاس انتهای پیکان بیانگر اینترفیسی است که باید در کلاس‌های طرف دیگر رابطه پیاده‌سازی شود (رابطه میان یک سرویس Search و یک اینترفیس SiteSearch).
- **Dependency**: در این رابطه کلاس ابتدای پیکان، به کلاس انتهای پیکان برای عملکرد صحیح خود نیازمند است (رابطه میان مشتری و تامین کننده).
- **Aggregation**: در این رابطه اشیاء کلاس در طرف لوزی، اجزایی تشکیل دهنده از اشیاء کلاس در طرف دیگر دارد. اجزا بدون وجود کل موجودیت معناداری دارند (رابطه میان استاد و دانشکده).
- **Composition**: رابطه کل به جز دارند، اما موجودیت اجزا بدون کل معنی ندارد (رابطه میان اتاق و ساختمان).

همچنین با توجه به نیاز و عدم نیاز و یا میزان پیشرفت کار می‌توان UMLها را به صورت ساده‌تر و با جزئیات کمتر و یا کامل‌تر و با جزئیات بیشتر طراحی کرد.

مطالعه‌ی بیشتر

برای مطالعه‌ی بیشتر می‌توانید به لینک Unified Modeling Language(UML) مراجعه کنید و همچنین برای طراحی راحت‌تر UML می‌توانید از لینک UML design کمک بگیرید.

انجام دهید: پیام رسان

تمرین کارت‌های CRC

برای یادگیری روش کارت‌های CRC می‌خواهیم نمونه‌ی ساده‌تری از یک پیام رسان را ایجاد کنیم که شرح آن به صورت زیر است:

در این نرم‌افزار، تعدادی کاربر وجود دارد. هر کاربر یک شماره‌ی شناسایی دارد که برای هر کاربر، متفاوت از دیگری است.

همچنین، در این نرم‌افزار سه نوع پیام وجود دارد، پیام متنی، پیام تصویری و پیام ویدئویی. هر پیام می‌تواند ویرایش شود و از طرفی، نویسنده‌ی هر پیام مشخص است و کسی غیر از نویسنده‌ی آن، نمی‌تواند آن را ویرایش کند. هر کاربر می‌تواند یک پیام را لایک[1] و یا دیس‌لایک[2] کند و برای هر پیام، تعداد لایک‌ها و دیس‌لایک‌های آن مشخص است.

علاوه بر آن، این پیام‌رسان انواعی از اجتماع‌های مختلف دارد که عبارتند از گروه‌ها و کانال‌ها. هر یک از اجتماع‌ها شامل تعدادی کاربر هستند و کاربرانی که عضو آن اجتماع نیستند، امکان دسترسی به محتوای آن را ندارند.

در هر گروه لیستی از پیام‌ها موجود است که هر کدام از اعضا می‌توانند پیام جدیدی به آن اضافه کنند ولی هیچ کسی نمی‌تواند هیچ یک از پیام‌ها را پاک کند.

هر کانال یک مدیر دارد و در آن همانند گروه، لیستی از پیام‌ها موجود است ولی تنها مدیر کانال می‌تواند پیام جدیدی به آن اضافه کند. از طرفی مدیر می‌تواند یک یا تعدادی از پیام‌ها را نیز پاک کند.

انجام دهید

حال با توجه به تعریفی که از نرم‌افزار به شما داده شد، از شما می‌خواهیم که با طراحی کارت‌های CRC مربوط به این تمرین، طرحی کلی از آن ارائه دهید.

[1] Like
[2] Dislike

انجام دهید: تاکسی اینترنتی

طراحی UML Class Diagram نرم‌افزار تاکسی اینترنتی

در این قسمت سعی داریم نمونه ساده‌ای از یک اپلیکیشن تاکسی اینترنتی طراحی کنیم. در این برنامه سه نوع کاربر وجود دارد:

- مسافر: دارای نام، شماره تماس، موقعیت مکانی، مانده‌ی اعتبار و شماره حساب است.
- راننده: علاوه بر داشتن مشخصات مسافر، کد ملی و لیستی از سفرهایی که انجام داده است را نیز دارد.
- مدیر: دارای نام، شماره تماس، مانده‌ی اعتبار، شماره حساب و کد ملی است و به لیست تمام مسافران و رانندگان دسترسی دارد.

در این برنامه هر مسافر می‌تواند با انتخاب مبدأ و مقصد سفر خود و انتخاب نوع تاکسی (راننده‌ی دلخواه یا خانم و تاکسی اقتصادی یا تشریفاتی) درخواست سفر کند. درخواست‌ها توسط مدیر بررسی و سازمان‌دهی می‌شوند. همچنین هر مسافر می‌تواند اعتبار حساب خود را افزایش دهد.

مدیر پس از بررسی راننده‌های دارای شرایط، سفر را به نزدیک‌ترین راننده می‌دهد. همچنین راننده و مسافر شماره تماس یکدیگر را برای هماهنگی از مدیر دریافت می‌کنند.

در پایان مسیر، مدیر مبلغ سفر را محاسبه و سپس از اعتبار حساب مسافر کم و به اعتبار حساب راننده اضافه می‌کند. همچنین اطلاعات سفر (شامل اطلاعات راننده و مسافر و مبدأ و مقصد و هزینه) به لیست سفرهای راننده اضافه می‌شود.

مدیر در پایان هر ماه، بخشی از درآمد راننده را از اعتبار حساب او کسر و به حساب خود می‌افزاید.

انجام دهید

حال با توجه به توضیحی که در رابطه با برنامه داده شد، از شما می‌خواهیم که نمودار UML این برنامه را طراحی کرده و ارائه دهید. همچنین دقت کنید که اصول ذکر شده را باید در طراحی خود رعایت نمایید.