دستور کار ۷

معرفی مهندسی نرم افزار

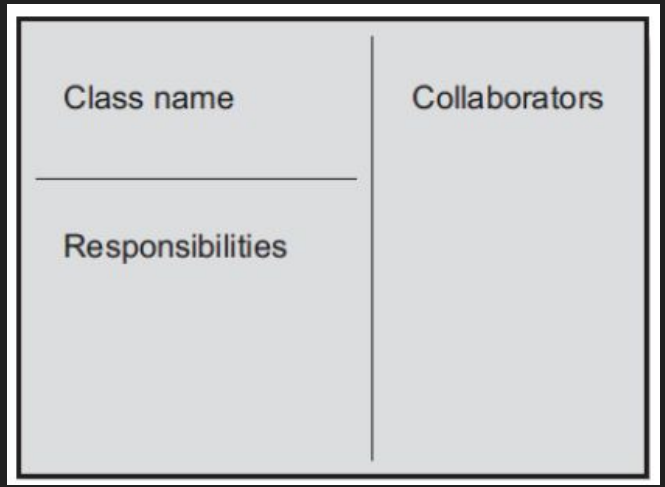
همانطور که تا به حال متوجه شده اید، یکی از چالش های بزرگی که در پروژه های واقعی برنامه نویسی با آن سروکار داریم، مهندسی خود نرم افزار است. مهندسی نرم افزار یک روش مهندسی برای توسعه نظام‌مند نرم‌افزار است. در توسعه نرم افزار فعالیت های متعددی من جمله طراحی و پیاده‌سازی وجود دارد. در این قسمت تمرکز بر روی طراحی اجزای سامانه است. با چالش های مهندسی نرم افزار و روش های حل آن بسیار مفصل در درس مهندسی نرم افزار آشنا خواهید شد. اما در این درس قصد داریم که مقداری با راه حلهای ابتدایی برای این چالش ها آشنا شویم

معرفی چند اصل در مهندسی نرم افزار

- وابستگی کم: وابستگی به معنای مقدار وابسته بودن اجزای مختلف نرم افزار به پیاده سازی داخل دیگر اجزا است. در طراحی نرم افزار سعی داریم نرمافزار را به شکلی طراحی کنیم که مقدار وابستگی را به کمترین حد ممکن برسانیم و اجزای نرم افزار به صورت مستقل مسئولیت های خود را انجام دهند.
- انسجام بالا: انسجام به معنای تعداد وظیفه هایی است که هر بخش از نرم‌افزار مسئولیت انجام آن را دارد. اگر یک بخش از نرم افزار بیش از یک وظیفه ی مشخص داشته باشد، آنگاه نرم افزار اصطلاحاً انسجام کمی دارد. در طراحی نرم افزار سعی میشود که هر یک از اجزای نرمافزار یک مسئولیت مشخص داشته باشد

آشنایی با کارت های CRC

کارتهای CRC یکی از روشها در طراحی نرم افزار شی گرا است. این روش با تحلیل متن پروژه با استفاده از دانش قبلی زبانی طراح سعی می کند روشی سامان یافته برای طراحی ساختار نرم افزار 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:



راه ساخت کارت‌های CRC

برای ساخت کارتهای CRC ابتدا با استفاده از روش اسم ها/فعل ها ، کلاس‌ها و وظایف آنها را پیدا میکنیم و سپس با استفاده از آنها، کارتها را میسازیم. در این روش، ابتدا صورت مسئله داده شده را به دقت بررسی می‌کنیم و در آن، اسم‌ها و فعل‌ها را مشخص می‌کنیم. اسم‌ها نمایندهی کلاس ما هستند و فعلها وظایف آنها را نشان میدهند

روش اسم‌ها/فعل‌ها

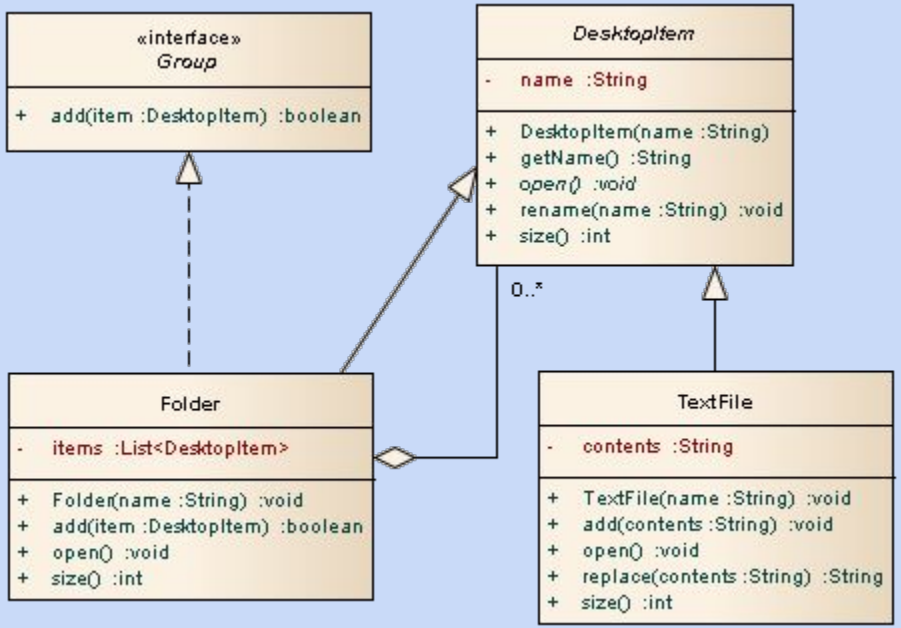
سیستم رزرو بلیت سینما باید رزروهای مربوط به چند سالن مختلف را ذخیره کند. هر سالن تعدادی صندلی در ردیف های مختلف دارد. مشتریان می توانند صندلی رزرو کنند که به آنها شماره صندلی و شماره ردیف در سالن داده می شود. مشتری میتواند رزرو چندین صندلی مجاور را درخواست کند.

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 می شود 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 دستور کار 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 دسترسی های مربوط به اعضای کلاس و نوع فیلدها و پارامترهای متدها را نیز نمایش میدهد


«interface»
Group
+ add(item :DesktopItem) :boolean
DesktopItem
- name :String
+ DesktopItem(name :String)
+ getName() :String
+ open() :void
+ rename(name :String) :void
+ size() :int
0..*
Folder
- items :List<DesktopItem>
+ Folder(name :String) :void
+ add(item :DesktopItem) :boolean
+ open() :void
+ size() :int
TextFile
- contents :String
+ TextFile(name :String) :void
+ add(contents :String) :void
+ open() :void
+ replace(contents :String) :String
+ size() :int

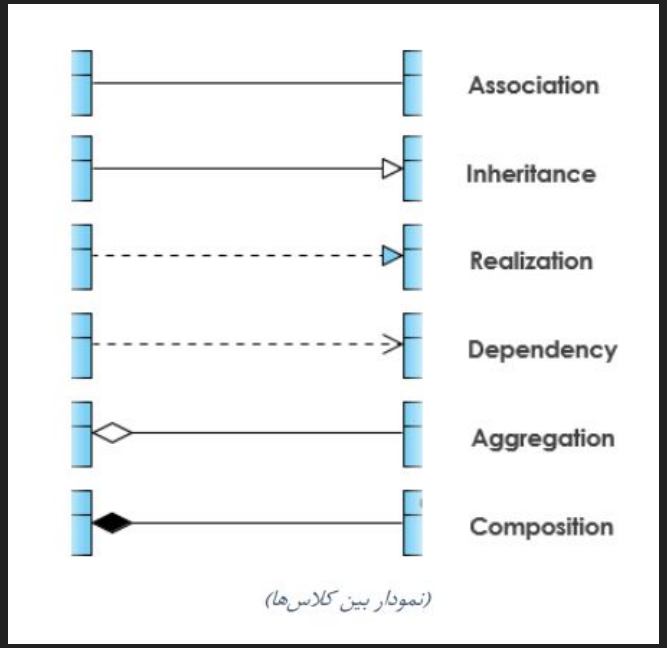
cardinality of associations between classes

- 1 = exactly 1
- 0..* or “” = are the same
- 1..* = at least 1 or more
- 1..n = at least 1, at most n

سطح دسترسی با علائم + و - و # نشان داده می‌شوند.

- private: -
- public: +
- protected: #

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



(نمودار بین کلاس‌ها)

توضیح نمودار ها

- Association 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 ارث بری میکند (رابطه میان سگ و حیوان).
- Realization این رابطه برای نمایش اینترفیس و کلاس پیاده سازی کنندهی آن است. کلاس انتهای پیکان بیانگر اینترفیسی است که باید در کلاسهای طرف دیگر رابطه پیاده سازی 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)