الگوهای طراحی نرم افزار

کارشناسی کامپیوتر

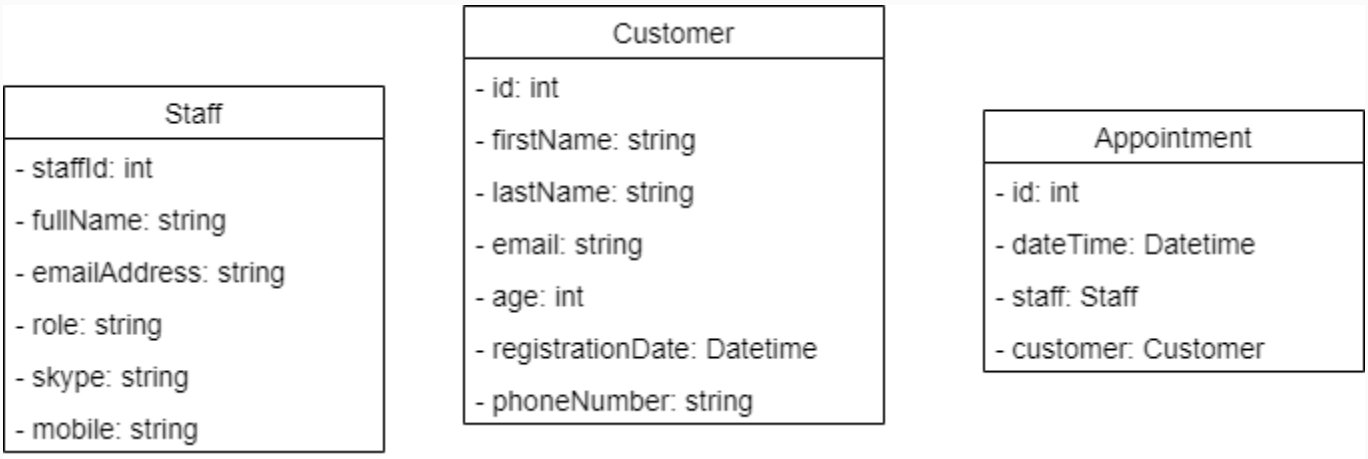
پائیز ۱۳۹۷

پروژه پایانی

پروژه پایانی به صورت **فردی** انجام می شود و مهلت تحویل آن، **۸ اسفند ۱۳۹۷** ساعت ۲۳:۵۹ است. در این تاریخ تمام **مستندات پروژه**، را در سایت آپلود می کنید.

در این پروژه یک اپلیکیشن با زبان جاوا پیاده سازی می کنیم که در واقع خود یک ساب سیستم از سیستمی بزرگتر (مثلا قسمتی از یک CRM) است که مدت زیادی بعد از انتشار نسخه اولیه سیستم اصلی، معرفی می شود.

در این سیستم از قبل سه کلاس داریم:



اکنون می خواهیم این امکان را ایجاد کنیم که مدیر سیستم بتواند بدون برنامه نویسی **قوانینی** را ایجاد کند که تحت **شرایط** خاصی، **عملیات** های مشخصی انجام شود.

هر کدام از این قوانین (Rule) از سه بخش اصلی تشکیل شده اند:

۱- رخداد (event)
۲- شرط (condition)
۳- عملیات ها (actions)

هر قانونی که تعریف می شود باید مشخص کند که چه رخدادی باعث می شود که آن قانون شروع به کار کند. شرایط، مشخص می کنند که آیا یک موجودیت (entity) در سیستم که رخداد پیرامون آن به وقوع پیوسته است، شرایط مورد نظر را دارد که عملیات روی آن انجام شود. عملیات ها نیز مشخص می کنند که چه کار یا کارهایی باید انجام شود.

رخداد هایی که داریم عبارتند از:

۱- یک موجودیت جدید به سیستم اضافه شده است
۲- یک موجودیت که از قبل در سیستم بوده آپدیت شده است
۳- یک موجودیت از سیستم حذف شده است

هر قانون حداقل و حداکثر باید یک رخداد داشته باشد.

هر قانون، یک یا چند شرط می تواند داشته باشد. این شرط ها می توانند گروه بندی شوند و بین آنها AND و OR قرار بگیرد. زمانی که شرط قانون TRUE شود، عملیات های تعریف شده برای آن قانون باید برای موجودیتی که با رخداد مرتبط است، اجرا شود.

هر کدام از شروط این قالب را دارند: (فیلد موجودیت) (عملگر) (یک مقدار ثابت)

فیلد های هر موجودیت را در دیاگرام کلاس ها می بینید. عملگر می تواند مساوی، نا مساوی باشد و برای تاریخ و عدد می تواند شامل بزرگتر و کوچکتر هم باشد.

عملیات های قابل انتخاب عبارتند از:

۱- مقدار دهی یکی از فیلد های موجودیت
۲- ارسال ایمیل به آدرس ایمیل موجودیت (فقط برای آنهایی که ایمیل دارند قابل انجام است)
۳- تلفن کردن به موجودیت (فقط برای آنهایی که شماره تلفن دارند قابل انجام است)

هر قانون می تواند حداقل یک عملیات یا بیشتر را داشته باشد.

برای مثال یک قانون تعریف می کنیم که اگر یک customer به سیستم اضافه شد که سن او بیشتر از ۵۰ سال است، ایمیلی برای وی ارسال شود. یا یک قانون که اگر customer آپدیت شد که از تاریخ مشخصی به قبل ثبت نام کرده و یا

شماره موبایل اش صفر است، ایمیلی برای وی ارسال شود که باید شماره موبایلش را آپدیت کند و فیلد شماره موبایلش به یک مقدار پیش فرض تغییر کند.

در برنامه ای که شما می نویسید توجه به چند نکته ضروری است:

۱- برای عملیات های ۲ و ۳ نیازی به پیاده سازی واقعی نیست صرفا نمایش یک متن یا نوشتن در یک فایل به نشانه انجام عملیات، کافی است.
۲- رابط کاربری برای ایجاد قوانین و ذخیره آنها در یک دیتابیس لازم نیست. بلکه برای تست برنامه، آنها را در یک دایرکتوری در کنار پروژه می توانیم قرار دهیم. که این می تواند به صورت یک فایل یا چند فایل با فرمت JSON، XML یا هر فرمت شناخته شده دیگری که می شناسید ذخیره شود. در ابتدای شروع برنامه همه این قوانین باید در برنامه لود شوند و منتظر وقوع رخداد هایی که برای آنها ثبت شده، باشند تا کار خود را انجام بدهند.
۳- رابط کاربری برای افزودن، آپدیت و حذف موجودیت ها و دیتابیس برای ذخیره آنها لازم نیست. یک موتور در برنامه شما باید داده ها و وقایعی که باید رخ بدهد را از یک فایل بخواند و کار سیستم را با آن شبیه سازی کند.
۴- مناسب است که یک فایل لاگ وجود داشته باشد که ترتیب رخ دادن وقایع و اتفاقاتی که برای موجودیت ها می افتد یا عملیات هایی که انجام می شود را به ترتیب در آن ذخیره کند که بتوانیم خروجی برنامه را ببینیم.
۵- موارد ۲، ۳ و ۴ امکاناتی است که برنامه شما برای نمایش کارکرد برنامه اضافه می کند، طراحی این قسمت ها و نحوه ارتباط آنها با زیرسیستم مستقل قوانین، نیز موضوع مهمی است و باید از اصول طراحی تبعیت کند.

آنچه که در این پروژه اهمیت دارد، **طراحی** شما است از همه امکاناتی است که این سیستم باید داشته باشد. پیاده سازی پروژه با آنکه ضروری است و ثابت می کند که طراحی شما دارد کار می کند، در رتبه بعدی اهمیت قرار دارد.

در طراحی نرم افزاری که می خواهید بسازید باید به استفاده از encapsulation، polymorphism و رعایت اصول طراحی نرم افزار که در این درس مطالعه کردیم، پایبند باشید.

در طراحی خود، از الگو های طراحی زیر که در درس خوانده اید، یا الگو های طراحی که دانشجویان ارائه کردند یا الگو های طراحی دیگری که نام آشنا هستند، استفاده کنید. البته این طور نباید تصور شود که باید از همه این الگو ها استفاده شود یا به زور در سیستم به کار رود. بلکه باید در جاهایی که واقعا با هدف و نیاز الگو های مورد نظر، تطابق دارد اعمال بشود و یا پیش بینی می شود که امکانات برنامه در آینده به آن سمت می رود و یا نیاز هایی است که ممکن است در آینده رخ بدهد و قابلیت توسعه برنامه باید تضمین شود.

الگو هایی که در این درس خواندیم عبارتند از:

1- Strategy

2- Observer
3- Abstract Factory
4- Factory Method
5- Composite
6- Command
7- Decorator

آنچه که باید ارسال شود:

۱- دیاگرام کلاس ها در قالب UML
۲- گزارشی از طراحی انجام شده، اصول طراحی و الگو هایی که به کار رفته، اینکه در کجا استفاده شده اند و موضوعات دیگری که در رابطه با طراحی مهم به نظر می رسند
۳- کد در زبان جاوا