به نام خدا

تمرین دو

نیم‌سال ۱۴۰۲۱

توضیحات

- لطفاً پاسخ‌ها را به صورت تایپ شده در قالب فایل PDF، حداکثر تا ساعت ۲۳:۵۹ تاریخ تعیین‌شده در صفحه‌ی درس‌افزار درس بارگذاری نمایید.
- ذکر **نام و نام خانوادگی** به همراه **شماره دانشجویی** همه‌ی اعضای گروه، هم‌چنین **شماره‌ی تیم** در فایل PDF پاسخ‌ها ضروری است. در صورتی که نام هر یک از اعضای گروه در فایل پاسخ‌ها نباشد، به منزله عدم همکاری آن عضو در گروه و نارضایتی سایر هم‌گروهی‌ها محسوب شده و نمره تمرین برای آن فرد لحاظ نخواهد شد.
- در صورت ارسال پاسخ‌ها به صورت دست‌نویس تضمینی در تصحیح آن وجود نخواهد داشت.
- هدف درس مهندسی نرم‌افزار آشنایی شما با دنیای نرم‌افزار و افزایش مهارت تحلیل شماست. استفاده از ربات‌های هوشمند مانند ChatGPT برای پاسخ‌دهی به سوالات، مغایر با اهداف گفته شده است؛ از این رو توصیه می‌کنیم که برای پاسخ‌دهی به تمرین‌ها از این ربات‌ها استفاده نکنید.
- تمرین از **۱۱۰ نمره** است و **۱۰ نمره** امتیازی دارد. نمرات امتیازی هر تمرین فقط می‌تواند برای جبران نمرات ازدست‌رفته‌ی سایر تمرین‌ها استفاده شود و به بخش‌های دیگر درس مانند آزمون‌ها منتقل نمی‌شود.
- سیاست ارسال با تاخیر برای این تمرین به صورت زیر است:
  - تا ۲۴ ساعت نمره‌ای کسر نمی‌شود.
  - پس از ۲۴ ساعت، به ازای هر ساعت تاخیر ۱ درصد نمره کسر می‌شود.
  - این سیاست برای هر یک از تمارین درس برقرار است.
- چنانچه یک نفر از اعضای هر گروه پاسخ تمرین را در درس‌افزار درس بارگذاری کند، کافی است.
- پاسخ‌ها را به زبان **فارسی** بنویسید. در صورتی که ترجمه‌ی کلمه‌ای ناملموس می‌شد، واژه‌ی اصلی را به‌صورت پانویس اضافه کنید.
- **توجه کنید که پوشایی و دقت پاسخ‌های شما، ملاک ارزیابی است.**

موفق باشید

تیم آموزش مهندسی نرم‌افزار

sharif.software.engineering@gmail.com

سوال ۱ (۵ نمره)

یک قانون سرانگشتی در فاز تحلیل این است که «افراد تیم ایجاد در فاز تحلیل[1] باید بر نیازمندی‌هایی تمرکز کنند که در حوزه‌ی مسئله[2] و کسب‌وکار[3] قرار دارد».

۱. چه نوع نیازمندی‌هایی در این حوزه‌ها نیستند؟

۲. مثال بزنید.

پاسخ:

نیازمندی‌های فنی و technical؛ مثلاً موارد مرتبط با پایگاه داده یا تکنولوژی‌هایی که قرار است در فاز طراحی و پیاده‌سازی[4] استفاده کنیم.

هم‌چنین در فاز تحلیل در مورد نیازمندی‌های غیروظیفه‌ای نیز صحبت نمی‌کنیم. نیازمندی‌هایی مانند سرعت پاسخ به درخواست‌ها[5]، دسترس‌پذیری[6] نرم‌افزار، رابط کاربری[7] و تجربه‌ی کاربری[8] ...

---

[1] Analysis
[2] Problem Domain
[3] Business Domain
[4] Implementation
[5] Response Time
[6] Accessibility
[7] User Interface (UI)
[8] User Experience (UX)

سوال ۲ (۱۰ نمره)

۱. معماری یک خانه یا ساختمان را در نظر بگیرید و با معماری نرم‌افزار مقایسه کنید.
۲. رشته‌های معماری ساختمان و معماری نرم‌افزار چه شباهت‌هایی دارند؟ چه تفاوت‌هایی دارند؟

پاسخ بخش ۱:

شباهت‌ها:

- طراحی معماری هر دو، در ابتدای فرآیند ایجاد آن‌ها انجام می‌شود.
- با گذشته زمان، هزینه‌ی تغییر معماری در هر دو زیاد و زیادتر می‌شود.
- نسبت آدم‌هایی که می‌توانند یک ساختمان/نرم‌افزار را ایجاد/پیاده‌سازی کنند، از آدم‌هایی که می‌توانند معماری یک ساختمان/نرم‌افزار را طراحی کنند، بیشتر است (در صنعت نرم‌افزار، تعداد کدنویسان بیشتر از معماران است).
- برای طراحی معماری ساختمان/نرم‌افزار، الگوهای از پیش آماده‌ای وجود دارد (برای مثال در صنعت ساختمان، الگوهایی برای طراحی ساختمان‌های آموزشی (مانند مدرسه‌ها یا دانشگاه‌ها) یا ساختمان‌های درمانی (مانند بیمارستان‌ها) یا ساختمان‌های مسکونی وجود دارد).
- در طراحی معماری ساختمان/نرم‌افزار، به جنبه‌های غیروظیفه‌ای[9] نیز توجه می‌شود (برای مثال در طراحی معماری ساختمان، به دسترس‌پذیری[10] بخش‌های مختلف ساختمان از جمله پارکینگ، رسیدن یا نرسیدن نور خورشید به یک واحد، منظره‌ی جلوی پنجره‌های یک واحد یا ... اهمیت داده می‌شود).
- هر چه ساختمان/نرم‌افزار بزرگ‌تر باشد، طراحی معماری آن نیز سخت‌تر می‌شود.
- «خلاقیت» در طراحی معماری ساختمان/نرم‌افزار نقش پررنگی دارد.

تفاوت‌ها:

- پس از ایجاد ساختمان/نرم‌افزار، تغییر معماری ساختمان تقریباً غیرممکن است، اما تغییر معماری نرم‌افزار امکان‌پذیرتر است؛ چرا که به‌صورت کلی می‌توان گفت نرم‌افزار نرم‌تر از ساختمان است.

پاسخ بخش ۲:

شباهت‌ها:

- هر دو دانشجو دارند :/ (قبول داریم که سوال بخش ۲، سوال خوبی نبود)

تفاوت‌ها:

- دانشجویان رشته‌ی معماری ساختمان، پیشینه‌ی هنری و انسانی بیشتری تا دانشجویان رشته‌ی معماری کامپیوتر دارند. دانشجویان رشته‌ی معماری کامپیوتر جنبه‌ی مهندسی و ریاضی بیشتری

[9] Non-functional
[10] Accessibility

دارند.

- تعداد دانشجویان رشته‌ی معماری ساختمان بیشتر از دانشجویان رشته‌ی معماری نرم‌افزار است؛ چرا که معماری ساختمان حوزه‌ی مشهودتر، کاربردی‌تر و واضح‌تری از علم است (بشر به قدمت بودنش به ساختمان برای زندگی‌کردن نیاز داشته است، اما کمتر از ۱۰۰ سال است که نرم‌افزار پا به زندگی بشر گذاشته

توجه کنید که جواب‌های صحیح دیگر نیز قابل قبول است.

سوال ۳ (۲۰ نمره)

تفاوت فعالیت‌های تحلیل[11] و طراحی[12] سیستم‌های نرم‌افزاری را توضیح دهید. اطمینان حاصل کنید که در توضیحات خود به موارد زیر بپردازید:

- ارتباط آن دو با یک مساله و راه‌حل آن
- اهداف و تمرکز هر یک
- سطح انتزاع[13] هر کدام
- تقدم و تاخر هر یک از این دو فعالیت
- تفاوت مدل‌سازی ذیل هر فعالیت

پاسخ:

ارتباط آن دو با یک مساله و راه‌حل آن

در جریان‌کاری تحلیل به سوال چیستیِ سیستم پاسخ خواهیم داد در حالی که در جریان کاری طراحی به چگونگی عملکرد آن پاسخ خواهیم داد. تحلیل بر قلمروی مسئله احاطه دارد در حالی که طراحی بر قلمرو راه‌حل.

اهداف و تمرکز هر یک

اهداف و تمرکز فعالیت‌های تحلیل:

مدلسازی سیستم با یک رویکردِ «همانطور که هست»[14]. در بررسی‌های این جریان کاری ما شهودی کاملی به هر آنچه در سیستم می‌گذرد، منتزع از مسائل نرم‌افزاری، خواهیم رسید.

تکمیل و تدقیق نیازمندی‌ها. برای درک و مدلسازی کامل‌تر از ما وقعِ سیستم، غالبا نیازمندی‌های موجود کم می‌آورند. به این منظور نیازمندی‌های کنونی سیستم را تدقیق کرده و نیازمندی‌های جدیدتری نیز استخراج می‌کنیم که به ما در مدل کردن و فهم کامل سیستم کمک کند.

غالبا مدل‌ها و مستندات تهیه شده در این جریان کاری به حدی منتزع از جزییات هستند که می‌توان به عنوان یک پروپوزال از سیستم با ذی‌نفعان به اشتراک گذاشته شوند و برای «تصمیم رو به جلو»[15] مورد استفاده قرار گیرند.

اهداف و تمرکز فعالیت‌های طراحی:

مدلسازی سیستم با یک رویکردِ «آن طور که باید باشد»[16]. در بررسی‌های این جریان کاری ما شهود کاملی نسبت به سیستم **نرم‌افزاری** که در حال ایجاد آن هستیم پیدا خواهیم کرد.

---

[11] Analysis
[12] Design
[13] Abstraction
[14] as-is
[15] Go Forward Decision
[16] to-be

تکمیل و تدقیق مدل‌های تحلیل. به مدل‌های تحلیلی که پیشتر ساختیم (با هر زبان مدل‌سازی، بر فرض UML یا حتی مستندات متنی) جزییات نرم‌افزاری می‌افزاییم. برای مثال ملاحظات مربوط به پایگاه داده، رابط کاربری[17]، کارگزارها[18] و ... همگی به مدل‌های پیشین اضافه می‌شوند.

تهیه نقشه‌راهی برای پیاده‌سازی. فعالیت‌های طراحی به حدی راه‌حل نرم‌افزاری را شفاف می‌کنند که مرحله بعد یعنی پیاده‌سازی به سادگی با استفاده از مستندات طراحی، یا حتی با استفاده از ابزارهای خودکارِ تبدیلِ مدل به کد، انجام خواهد شد.

سطح انتزاع هر کدام و تقدم و تاخر هر یک از این دو فعالیت

از لحاظ سطح انتزاع و تقدم-تاخر می‌توان ترتیب زیر را برای هر تکرارِ ایجاد نرم‌افزار در نظر گرفت:

نیازمندی←تحلیل←طراحی←پیاده‌سازی و تست

همانطور که در اهداف و تمرکزهای جریان‌های کاری تحلیل و طراحی ذکر شد، طراحی در واقع تدقیقِ[19] تحلیل و بالطبع تحلیل انتزاعی[20] از طراحی است. یک چنین ترتیبی از حل مسئله در حوزه نرم‌افزار از این بابت مورد پسند مهندسین است که مانند خود روش‌های برنامه نویسی، از سطوح انتزاعی استفاده می‌کند و شبیه به تعریف چند کلاس که از همدیگر ارث‌بری می‌کنند و یک دیگر را کامل می‌کنند می‌باشد. در واقع مهندس می‌تواند مرحله مرحله نیازمندی را محقق کند و در هر مرحله پاسخ پیشین خود را کامل‌تر سازد (به جای اینکه پرش بلندی از نیازمندی به پیاده‌سازی بزند).

تفاوت مدل‌سازی ذیل هر فعالیت

از منظر مدل‌سازی نیز مدل‌های تحلیل - همانطور که پیش‌تر مطرح شد - از جزییات نرم‌افزاری کاملا مبرا هستند. این موضوع اما در مورد مدل‌های طراحی کاملا بر عکس است. مدل‌های طراحی می‌توانند به حدی دارای جزییات باشند که فعالیت پیاده‌سازی تقریبا کار بدیعی نداشته باشد جز کد زدن مابه ازای مدل‌های طراحی. مدل‌سازی با استفاده از نمودار کلاسی UML به عنوان مثال در نظر بگیرید. کلاس‌های تحلیل معمولا کلاس‌های ماوقع سیستم هستند؛ مفاهیمی خواهند بود که حتی یک مالک محصول[21] غیر نرم‌افزاری نیست با شنیدن اسم آن‌ها حسی از وجودشان خواهند داشت. بسیاری از این کلاس‌ها دقیقا از خود دامنه‌ی کسب‌وکار استخراج شده‌اند و در مرحله بعد - یعنی طراحی - معادلا برای بسیاری از آن‌ها یک کلاسِ ORM نیز قرار داده می‌شود تا نمونه‌هایشان[22] در پایگاه داده ذخیره شوند.

جزییات زیادی به این مدل‌ها در فعالیت‌های طراحی اضافه خواهد شد و به عبارتی تدقیق می‌شوند. تعداد کلاس‌های موجود در نمودار کلاسی چند برابر می‌شود چرا که تعداد زیادی کلاس جعلی[23] تحت اثر الگوهای طراحی شکل می‌گیرند. به علاوه وظایف حیطه نرم‌افزار (مانند کار با UI یا پایگاه داده) به کلاس‌های پیشین اضافه می‌شود که به دلیل حفظ تک مسئولیت در سیستم خود این وظایف به سلسه‌مراتب[24] هایی موازی با

---

[17] User Interface (UI)
[18] Servers
[19] Refinement
[20] Abstraction
[21] Product Owner - PO
[22] Instance
[23] Fabricated
[24] Hierarchy

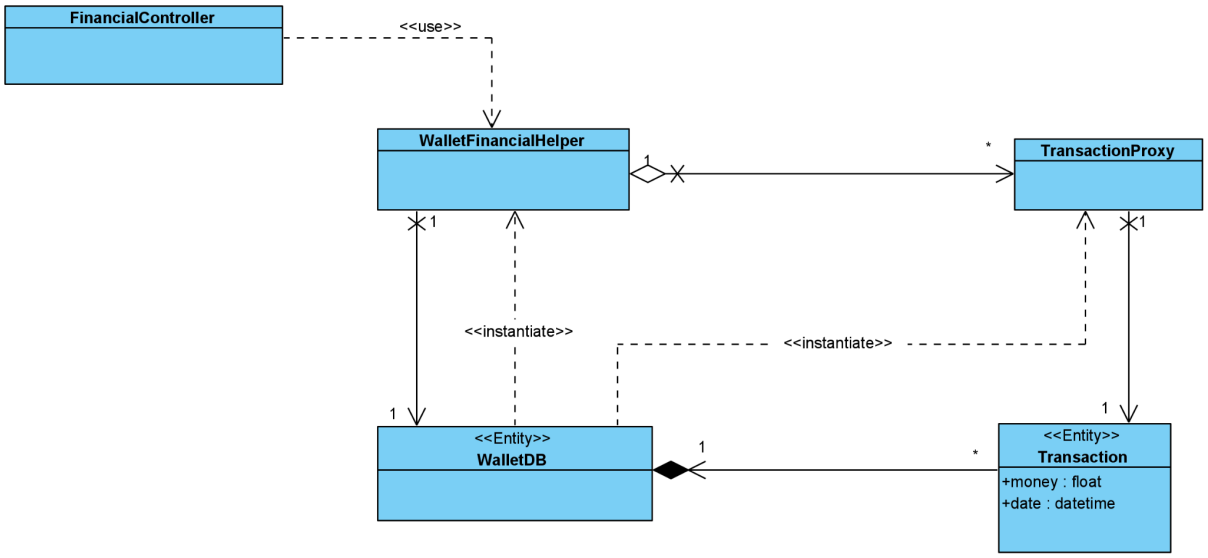
کلاس‌های پیشین واگذار[25] خواهند شد. تعدادی عملیات و صفت جدید به نمودار اضافه می‌شود و عملیات‌ها و صفاتی که از مرحله تحلیل وجود داشتن نیز جزییاتی از قبیل سطح دسترسی (private, protected, public) و نوع[26] دریافت خواهند کرد.

[25] Delegate
[26] Type

سوال ۴ (۲۵ نمره)

در یک فروشگاه تحت وب، کسب و کار مربوطه تنها عمل واسط را بر عهده داشته و تعداد زیادی مشتری را به تعداد زیادی انباردار متصل می‌کند. در واقع هم مشتری و هم انباردار در این سیستم دارای حساب و کیف پول می‌باشند، و هر خرید پول را مستقیم از کیف پول مشتری به کیف پول خریدار منتقل می‌کند (برای سادگی انتقال پول بدون هیچ هزینه‌ای اتفاق می‌افتد).

فرض کنید مدل زیر مابه‌ازای بخشی از کد زیرسیستم backend این فروشگاه می‌باشد؛ به چنین مدلی، مدل طراحی می‌گویند. به منظور تسهیل سوال، بسیاری از جزئیات (داده و عملیات کلاس‌ها) حذف شده‌اند و تمرکز مدل بر کلاس‌ها و روابط بین آن‌ها است.

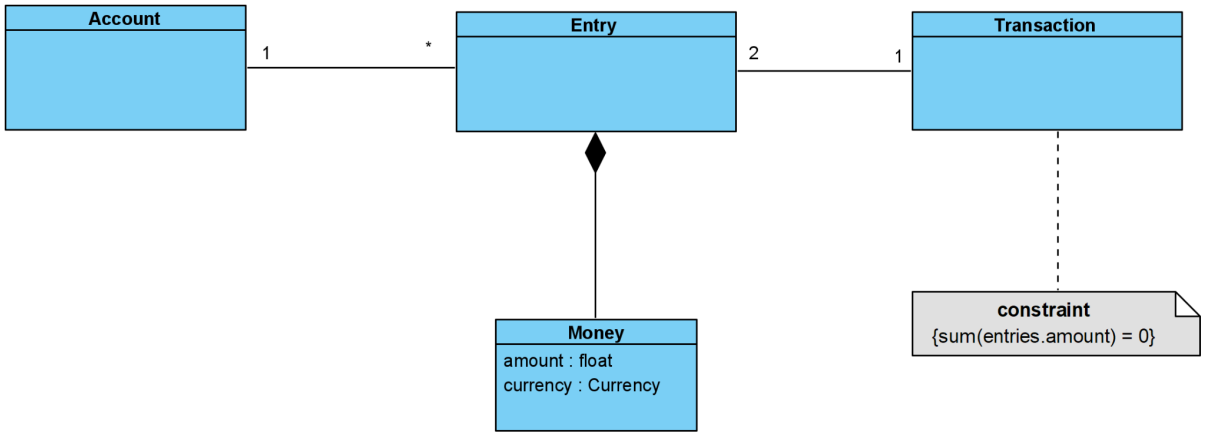


۱. مدل تحلیل متناظر با مدل طراحی فوق از دست رفته است. آن را شما تعبیه کنید. (**راهنمایی:** مدل تعبیه شده توسط شما باید بسیار ساده‌تر و کوچک‌تر از مدل فوق باشد)

۲. در فصل‌های ۸ تا ۱۱ کتاب پرسمن، بارها به الگوهای تحلیل -بالاخص الگوهای تحلیل فاولر[27]- اشاره شده است. می‌توانید کتاب فاولر را از اینجا دریافت کنید.

الگوی زیر «موجودی و حسابداری - تراکنش»[28] نام دارد (فصل ۶ کتاب الگوهای تحلیل فاولر).

[27] Fowler's Analysis Patterns
[28] Inventory and Accounting - Transaction



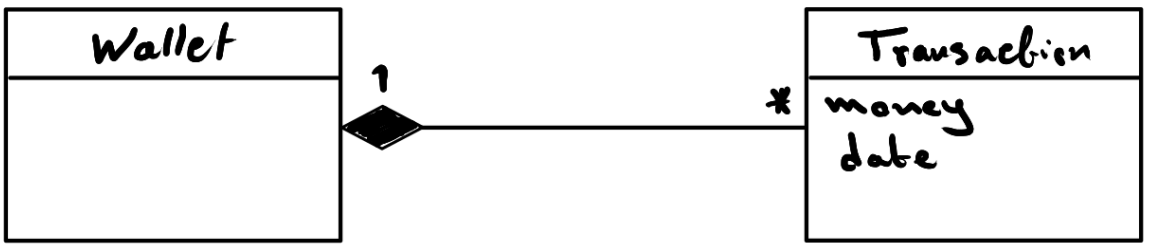
از این الگو برای غنی کردن مدل تحلیلی که در بخش ۱ فراهم آوردید، استفاده کنید. در واقع باید به نحوی مدل بخش ۲ را تغییر دهید که این الگو در آن نهادینه شده باشد (مانند استفاده از الگوهای طراحی در هنگام کد زدن).

۳. دو مورد از بهبودهایی را که این الگو به ارمغان می‌آورد، توجیه کنید.

پاسخ:

۱. (۱۰ نمره: ۷ نمره مدلسازی، ۳ نمره توضیحات)

پیش‌تر در سوال ۳ از این تمرین تفاوت مدلسازی ذیل جریان‌کاری تحلیل و طراحی را تشریح کردیم. مدلسازی در فضای تحلیل از جزییات نرم‌افزاری مبری است و تمرکز آن بر بصری‌سازی دامنه‌ی مسئله است. به همین ترتیب انتظار می‌رود مدل تحلیلی که از روی مدل طراحی فوق می‌کشیم تنها دارای کلاس‌هایی باشد که در قلمروی مسئله دارای موضوعیت هستند. کلاس‌هایی که وقتی از آن‌ها اسم می‌بریم، برای فردی با دانش نرم‌افزاری اندک اما دارای درک کافی از دامنه‌ی کسب‌وکار معنادار هستند.



کلاس TransactionProxy در مدل طراحی، یک بسته‌بند[29] دور کلاس دیتابیسیِ Transaction بود. تمام بسته‌بندها (پروکسی، آداپتور، دکوراتور و نما[30]) کلاس‌های جعلی هستند که طراحی را راحت‌تر و خواناتر

[29] Wrapper
[30] Facade

می‌کنند و وجودشان در کلاس‌های تحلیل فاقد موضوعیت است. چنین کلاس‌هایی در کسب‌وکار وجود ندارند و فقط در پیاده‌سازی نرم‌افزاری با توجیحات مهندسی وارد می‌شوند.

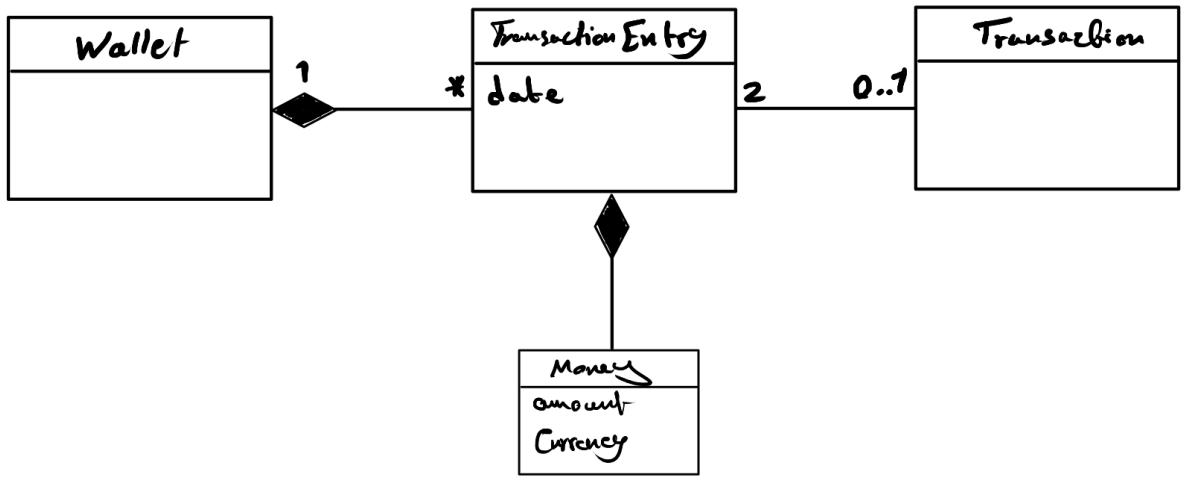
کلاس WalletFinancialHelper یک واگذاری[31] از بخشی از رفتارِ کلاس WalletDB است. در واقع طراح به دلیل حفظ اصل تک‌مسئولیتی[32] ترجیح داد کلاس WalletDB را منسجم[33] نگه دارد و برای بخشی از رفتار ریزدانه‌ی مالی آن یک کلاس Helper قرار دهد. وجود این کلاس نیز در فضای مسئله فاقد موضوعیت است و انتظار می‌رود مسئولیت[34]های آن در خود ماهیتِ کیف‌پول مشهود باشد.

کلاس FinancialController تحت اثرِ الگوهای طراحیِ GRASP[35] قرار داده شده است و وظیفه آن این است که مرز مشخصی بین View (درگاه ارتباط با UI) و Model (محل قرارگیری منطق کسب‌وکار) وجود داشته باشد و در واقع این دو مهم در هم بافته[36] نشوند - و تحت تاثیر این پیچش، تغییر و خوانایی کد آسیب ببیند. چنین ملاحظاتی برای بار دیگر تنها در طراحی معنادار است و در فضای مسئله که UI فاقد موضوعیت است، حضور چنین کلاس‌هایی را نمی‌بینیم.

لازم به ذکر است در یک سناریوی واقعی کلاس‌های درون نمودار کلاسیِ طراحی، دارای تعداد زیادی عملیات[37] (و صفت)[38] خواهند بود که در نمودار تحلیل معادل آن، به ازای هر یک یا چند عملیات (و صفت) تنها یک عملیات (صفت) به صورت ادغام شده قرار خواهد داشت.

در مورد مدل تحلیل رسم شده در تصویر بالا، ما حتی جهت رابطه بین Transaction و Wallet هم رسم نکردیم. معمولا مقوله «جهت» در مواردی که اشاره‌گرهای سطح زبان مورد توجه ما است حائز اهمیت می‌شود که باز مربوط به فضای طراحی می‌باشد.

۲. (۵ نمره: مدلسازی)



[31] Delegation
[32] Single Responsibility Principle
[33] Cohesive
[34] Responsibility
[35] **G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns - GRASP
[36] Intertwine
[37] Operation (AKA Method)
[38] Attribute (AKA Field)

تحت اثر الگوی تحلیل مزبور زین پس TransactionEntry (در مدل Entry نام دارد) بر موجودی کیف پول ما تاثیر خواهد داشت. Transaction در مدل ما تنها نگهدارنده‌ی دو TransactionEntry کنار یکدیگر است؛ ورودی اول مقداری پول از یک کیف‌پول (نوعا خریدار) کم کرده و ورودی دوم آن مقدار پول را به کیف‌پول مقصد (نوعا انباردار) منتقل می‌کند. در مورد اینکه تاریخ تراکنش در TransactionEntry قرار گیرد یا Transaction اگرچه می‌توان همچنان بحث نمود و تحت تاثیر شرایط مسئله تصمیم نهایی را گرفت؛ فعلا در این مدل، آن را در کلاس TransactionEntry قرار داده‌ایم.

یک مورد گوشه‌ای در شرایط مسئله وجود دارد که آن واریز پول به کیف‌پول است - در این حالت کاربر از درگاه بانکی مقدار پول از کارت بانکی خود که کسب‌وکار ما نسبت به آن دیدی ندارد خارج می‌کند و وارد کیف پول درون سیستم می‌کند. در این موارد ورودی به کیف‌پول یک طرفه است (پول از کیف‌پولی به کیف‌پول دیگر نمی‌رود). به همین منظور صلاح است شمارش[39] در سمت Transaction صفر یا یک باشد (که طبق قواعد UML نمایش آن اینطور خواهد بود: 0..1). **لازم به ذکر است اگر این مورد در پاسخ شما نادیده گرفته شد، بابت آن نمره‌ای از شما کسر نشده است.**

۳. (۱۰ نمره: ۸ نمره بهبود اول، ۲ نمره بهبود دوم)

بهبود اول مفهوم انتقال پول از یک حساب به حساب دیگر است که آن به خوبی با استفاده از این الگو مدل شده است. در مدل پیشین تحلیلمان کسر و افزایش پول از کیف‌پول‌های مختلف مشهود بود اما به خوبی معلوم نمی‌شد که وقتی پولی از حسابی کسر شده به کدام حساب افزوده شده است؟ ما ترجیح می‌دهیم مدل‌های تحلیلمان (که یک نمونه مهم از مدلسازی برای مهندسین نمودارهای کلاسی است) تا جای ممکن به دامنه کسب‌وکار نزدیک باشد. این نزدیکی، احتمال تاثیرگذاری کسب‌وکار در طراحی و پیاده‌سازی نهایی را بالا می‌برد. در ادامه می‌توان مانند خود دامنه کسب‌وکار یک سری صحت‌سنجی‌ها را در مورد وقایع انجام داد برای مثال اطمینان حاصل کرد به میزانی که پول از یک کیف پول خارج می‌شود حتما وارد حساب دیگر شود.

بهبود دوم استفاده همزمان از الگوی پول[40] در مدل‌هایمان است - این الگو به صورت عام‌تر تعداد[41] نام دارد. کلاس Money قرار داده شده ازین پس قابلیت پشتیبانی از واحدهای پول دیگر را به ما می‌دهد همچنین رفتارهای مرتبط با مسائل ساده‌ی مالی (مانند تبدیل واحدهای پول) ماژول تعریف شده‌ای برای خود دارد - یعنی همین کلاس Money.

---

[39] Cardinality
[40] Money
[41] Quantity

سوال ۵ (۲۰ نمره)

پنج مفهوم BPMN و CRC card و User Story و UML و DFD را از جنبه‌های زیر با یکدیگر مقایسه کنید:

- چه چیزهایی را مدل می‌کنند
- آن‌ها را چگونه مدل می‌کنند
- کجا/در چه زمانی استفاده می‌شوند
- تفاوت سطح انتزاع در مدل‌سازی

پاسخ:

BPMN

- چه چیزی را مدل می‌کند؟
  - فرآیندهای موجود در دامنه‌ی کسب و کار[42] را مدل می‌کند (جنبه‌ی رفتاری)، اما در مورد ساختار اجزای موجود در سیستم (جنبه‌ی ساختاری) یا وظایفی که هر جز در سیستم انجام می‌دهد (جنبه‌ی وظیفه‌ای) صحبتی نمی‌کند.
- آن را چگونه مدل می‌کند؟
  - خود BPMN نشانه‌گذاری[43] خاصی را تعریف می‌کند که به flowchart و بعضی از نمودارهای UML شباهت دارد. با این نشانه‌گذاری فرآیندهای کسب و کار را مدل می‌کنند.
- کجا/در چه زمانی استفاده می‌شود؟
  - تحلیل‌گرها[44] از BPMN و برای مدل‌سازی فرآیندهای موجود در دامنه‌ی کسب و کار استفاده می‌کنند. این مدل‌سازی اصلاً جنبه‌ی فنی و technical ندارد و در فاز تحلیل (قبل از ورود به دامنه‌ی راه‌حل و دامنه‌ی پیاده‌سازی) انجام می‌شود.
- سطح انتزاع مدل‌سازی؟
  - BPMN می‌تواند هم به‌صورت ریزدانه نقش[45]های دخیل در هر فرآیند ریزدانه‌ی کسب و کاری و جریان[46] اجرای آن را مشخص کند، و هم به صورت درشت‌دانه کلیت یک فرآیند درشت‌دانه را نمایش دهد.

CRC card

- چه چیزی را مدل می‌کند؟
  - جنبه‌ی وظیفه‌ای[47] ارتباط کلاس‌های مختلف موجود در یک سیستم را مدل می‌کند؛ به این صورت که مشخص می‌کند هر کلاس **چه** وظایفی را انجام می‌دهد و برای این وظایف به چه

---

[42] Business
[43] Notation
[44] Analysts
[45] Actor
[46] Flow
[47] Functional

کلاس‌های دیگری تعامل دارد. توجه کنید که کارت‌های CRC در مورد ساختار یک کلاس (جنبه‌ی ساختاری[48]) صحبتی نمی‌کند. هم‌چنین در مورد **چگونگی** انجام وظایف (جنبه‌ی رفتاری[49]) نیز صحبتی نمی‌کند.

- آن را چگونه مدل می‌کند؟
  - به صورت نوشتاری، با مشخص‌کردن یک کلاس، وظایف آن و کلاس‌های دیگری که برای انجام این وظایف با آن‌ها تعامل می‌کند
- کجا/در چه زمانی استفاده می‌شود؟
  - در فاز طراحی (دامنه‌ی راه حل[50]) و برای مدل‌کردن چیستی ارتباط بین کلاس‌های مختلف با هم.
- سطح انتزاع مدل‌سازی؟
  - کارت‌های CRC در سطح کلاس (ریزدانه) مدل‌سازی می‌کنند.

User Story

- چه چیزی را مدل می‌کند؟
  - می‌توان **نیازمندی‌های مشتریان** را در قالب داستان کاربر یا همان User Story بیان کرد؛ در اصل User Story چیزی جز جملاتی قالب‌مند برای بیان نیازمندی‌ها نیست.

    داستان‌های کاربر جنبه‌ی وظیفه‌ای نیازمندی‌ها را مدل می‌کنند؛ به این صورت که چیستی و چرایی نیازمندی‌ها را مشخص می‌کنند (جنبه‌ی وظیفه‌ای)، اما در مورد ساختار یا چگونگی پیاده‌سازی آن نیازمندی صحبتی نمی‌کنند (جنبه‌های ساختاری و رفتاری).
- آن را چگونه مدل می‌کند؟
  - هر نیازمندی را در قالب یک جمله به صورت زیر مدل می‌کند:
    - به عنوان ..(۱).. می‌خواهم ..(۲).. تا ..(۳)..
      - مورد (۱) نقشی را که این نیازمندی به آن نیاز دارد، بیان می‌کند
      - مورد (۲) چیستی نیازمندی را مشخص می‌کند
      - مورد (۳) چرایی و هدف پشت نیازمندی را مشخص می‌کند
- کجا/در چه زمانی استفاده می‌شود؟
  - داستان‌های کاربر اغلب در فاز تحلیل، برای جمع‌آوری نیازمندی‌های مشتریان استفاده می‌شوند، و در طول فرآیند ایجاد نرم‌افزار نیز از آن‌ها برای مدل‌سازی و پیاده‌سازی به کار گرفته می‌شوند.
- سطح انتزاع مدل‌سازی؟

---

[48] Structural
[49] Behavioral
[50] Solution Domain

- داستان‌های کاربر می‌توانند در سطوح انتزاع مختلف قرار بگیرند؛ از نیازمندی‌های در سطح درشت‌دانه‌ی معماری گرفته تا نیازمندی‌های ریزدانه‌ی در سطح یک خصوصیت/قابلیت[51] نرم‌افزار

UML

- چه چیزی را مدل می‌کند؟
  - UML زبان مدل‌سازی قدرتمندی برای مدل‌سازی جنبه‌های مختلف ساختاری و رفتاری یک سیستم است. UML در مدل‌سازی جنبه‌ی وظیفه‌ای ضعف دارد.
- آن را چگونه مدل می‌کند؟
  - UML با نمودارهای مختلفی که دارد، مانند نمودار فعالیت، نمودار بسته[52]، نمودار حالت[53]، نمودار مورد کاربرد[54]، ...، جنبه‌های ساختاری و رفتاری مختلف سیستم را مدل می‌کند.
- کجا/در چه زمانی استفاده می‌شود؟
  - در فاز تحلیل و طراحی، از UML برای مدل‌سازی استفاده می‌شود و از مدل‌های ساخته شده به زبان UML در تمام طول فرآیند ایجاد نرم‌افزار، برای مواردی از جمله پیاده‌سازی و کدنویسی استفاده می‌شود.
- سطح انتزاع مدل‌سازی؟
  - UML با داشتن انواع نمودارهای مختلف، می‌تواند سطوح انتزاع مختلف را مدل‌سازی کند؛ برای مثال با استفاده از نمودار بسته (package diagram) می‌توان در سطح درشت‌دانه و با استفاده از نمودار کلاس (class diagram) می‌توان در سطح کلاس و ریزدانه مدل‌سازی را انجام داد.

DFD

- چه چیزی را مدل می‌کند؟
  - نمودار DFD جریان داده را در فرآیندهای یک سیستم مدل‌سازی می‌کند (جنبه‌ی وظیفه‌ای). توجه کنید که DFD در مورد ساختار داده‌ها در سیستم صحبتی نمی‌کند (جنبه‌ی سیستمی)، هم‌چنین در مورد چگونگی انتقال داده‌ها یا چگونگی ارتباط و ترتیب عملیات‌ها (جنبه‌ی رفتاری) نیز صحبتی نمی‌کند.
- آن را چگونه مدل می‌کند؟
  - این دسته از نمودارها با کمک علائم و نشانه‌گذاری‌های خاصی که تعریف کرده‌اند، جریان انتقال داده در فرآیندهای سیستم را مدل‌سازی می‌کنند.
- کجا/در چه زمانی استفاده می‌شود؟

---

[51] Feature
[52] Package Diagram
[53] State Diagram
[54] Use Case Diagram

- از DFD در فاز تحلیل و طراحی، برای فهم بهتر نیازمندی‌های مشتری استفاده می‌شود. از آن‌جایی که DFD به جریان داده‌ها می‌پردازد و نمودار UML در آن ضعف دارد، می‌توان گفت DFD و UML تکمیل‌کننده‌ی یکدیگر در مدل‌سازی جنبه‌های مختلف یک سیستم هستند.

- سطح انتزاع مدل‌سازی؟
  - نمودارهای DFD را می‌توان هم در سطوح ریزدانه برای مدل‌سازی جریان انتقال داده بین کلاس‌ها استفاده کرد، و هم می‌توان در سطح درشت‌دانه برای مدل‌سازی جریان انتقال داده بین مولفه‌های مختلف سیستم به‌کار برد.

سوال ۶ (۴۰ نمره)

روش طراحی ویژگی‌رانه[55](نسخه سوم) را به دقت مطالعه کنید و تحلیل خود را از این روش بر اساس موارد زیر بیان کنید.

- **مستندات معماری:** شامل تصمیمات، عقلانیت[56]، دیدهای معماری[57]، راه‌حل‌های جایگزین، بازنمایی[58] و سایر موارد اشاره شده در کتاب
- **نگرانی‌های همه انواع ذی‌نفعان:** شامل کاربر نهایی، مشتری، تیم ایجاد، مدیر پروژه و ...
- **چگونگی کاربرد مفاهیم، اصول، الگوها و سبک‌های معماری**

توجه کنید که پوشایی و دقت پاسخ شما، ملاک مهم ارزیابی در این سوال است.

پاسخ:

پاسخ در این لینک نوشته شده است.

[55] Attribute-driven Design Method
[56] Rationality
[57] Architectural Views
[58] Representation