دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

درس مهندسی نرم‌افزار

دکتر مهران ریواده

امتحان پایان‌ترم | مصطفی قدیمی

سؤال ۱.

چه تفاوتی بین درس مهندسی نرم‌افزار و تحلیل طراحی سیستم‌ها وجود دارد؟ چه مباحثی در زمینه این درس باید در ادامه حیات مهندسی خود بیاموزید؟ آیا شما برای مهندسی نرم‌افزار گزینه‌ی مناسبی هستید؟ چرا؟

- الف) درس مهندسی نرم‌افزار و تحلیل و طراحی سیستم‌ها موضوعاتی هستند که در دو رشته‌ی مختلف درس داده می‌شوند. در مهندسی نرم‌افزار تمرکز بر روی تولید و توسعه نرم‌افزار است؛ این در حالی است که تمرکز درس تحلیل و طراحی سیستم‌ها بر روی ساخت یک سیستم اطلاعاتی است. یک سیستم اطلاعاتی نمی‌تواند بدون آن‌که نرم‌افزار آن را بسازیم، ایجاد شود. علت این‌که در زمینه مهندسی نرم‌افزار در مقایسه با تحلیل و طراحی سیستم‌ها درس‌های بیش‌تری ارائه می‌شوند، «کیفیت» در این حوزه است. در واقع در درس تحلیل و طراحی سیستم‌ها بیش‌تر با روش‌گان‌ها[۱] سروکار دارد. به زبان دیگر اگر به منابع[۲] و کتاب‌های این دو نگاهی اجمالی بیندازیم، متوجه می‌شویم که در کتاب‌های تحلیل و طراحی سیستم‌ها به دنبالی روشی برای طراحی مراحل [۳] سیستم‌های اطلاعاتی هستیم اما در کتاب‌های مهندسی نرم‌افزار بیش‌تر مدیریت پروژه نرم‌افزاری و ارزیابی کیفیت خروجی پروژه مد نظر است.
- ب) مهندسی نرم‌افزار شامل یک فرآیند، مجموعه‌ای از روش‌ها، تمرین‌ها و مجموعه‌ای از ابزارها است که به متخصصان اجازه می‌دهد، نرم‌افزارهای کامپیوتری با کیفیت بالا بسازند.

به همین دلیل، مهندس نرم‌افزار به مهارت‌های زیادی در حوزه‌های مختلف نیاز دارد که مهم‌ترین آن‌ها عبارتند از:

- چند کاری[۴]: به عنوان یک مهندسی نرم‌افزار نیاز است تا توانایی مدیریت کردن چندین پروژه را در محیط ددلاین محور را داشته باشیم. این مهارت را اگر به‌طور جزئی‌تر بخواهیم تحت نظر قرار دهیم شامل
  - سازمان‌دهی
  - اولویت‌بندی
  - ددلاین‌ها
  - مدیریت توقعات

  می‌شود.
- کار تیمی: ساخت سیستم‌های نرم‌افزاری عموما یک تلاش انفرادی است اما یک مهندس نرم‌افزار هنوز باید توانایی برقراری ارتباط با افراد و تیم‌های دیگر را داشته باشد و این ارتباط را به‌طور مرتب برقرار کند. به عنوان یک مهندس نرم‌افزار باید قادر باشیم تا نیازهای پروژه را بیان کنیم و اگر با چالش یا مشکلی مواجه شدیم، بحث کنیم. این بخش شامل
  - همکاری
  - سازش[۵]
  - گوش کردن به‌طور فعال
  - حل اختلاف
  - ارتباط شفاهی
  - ارتباط نوشتاری

---

[۱] methodology
[۲] resources
[۳] phases
[۴] Multitasking
[۵] compromising

می‌باشد.

- توجه به جزییات: در صنعت، روش‌ها و استانداردهای خاصی وجود دارد که یک مهندس نرم‌افزار باید به آن‌ها تسلط کامل بیابد. این موارد شامل اصول اولیه، مانند استفاده از سیستم‌های کنترل نسخه (مانند گیت‌هاب) است تا کار قدیمی خود را از دست ندهیم. البته این موارد شامل چیزهای پیچیده‌تری مانند تحلیل و نگهداری کد قدیمی‌تر یا تولید روش‌های استراتژیک برای ایجاد یک چارچوب کدزنی می‌شود. بنابراین مهارت‌های لازم در این قسمت شامل
  * مهارت‌های تحلیل
  * اشکال‌زدایی
  * مستندسازی فنی
  * فرمول‌ها
  * خلاقیت
  * تفکر بحرانی

  می‌باشد.
- زبان‌های برنامه‌نویسی: اکثر مهندسان نرم‌افزار به یک یا دو زبان برنامه‌نویسی تسلط و اشراف کامل دارند و این یک حوزه اصلی تخصص آن‌ها می‌شود.

- پ) با توجه موارد گفته شده در قسمت «ب»، بنده خود را یک گزینه‌ی مناسب برای مهندسی نرم‌افزار می‌دانم؛ زیرا

۱. تجربه‌ی کار در صنعت به عنوان توسعه‌دهنده و مدیر پروژه را دارم.
۲. با متدولوژی‌های گوناگونی مانند UP، Agile و... آشنا هستم.
۳. درس‌های زیادی را در دانشکده‌ی مهندسی کامپیوتر با موضوع مهندسی نرم‌افزار و نمره‌ی عالی در این زمینه گذرانده‌ام.
۴. تعداد زیادی از مهارت‌های عمومی لازم و نه فنی (نظیر کار تیمی، آشنایی با ددلاین‌ها و...) را دارا هستم.

سؤال ۲.

فرض کنید امروز در شرکتتان محصولی نرم‌افزاری ایجاد کرده و به مشتری فروخته‌اید. سی سال بعد وضعیت نرم‌افزار شما چگونه خواهد بود؟ آیا هنوز مشتری شما از نرم‌افزار استفاده می‌کند؟ اگر چنین است شما در طول این زمان چه کارهایی روی نرم‌افزار انجام داده‌اید؟ اگر از نرم‌افزار شما استفاده نمی‌شود به نظرتان چه دلایلی موجب این اتفاق شده است؟

- الف و ب) وضعیت نرم‌افزار در آینده، به عوامل زیادی بستگی دارد که از مهم‌ترین آن‌ها می‌توان به نوع قرارداد، متدولوژی مورد استفاده و نحوه‌پیاده‌سازی اشاره کرد. برای وضعیت نرم‌افزار نیز چندین حالت می‌توان داشت:

  ۱. نرم‌افزار به خوبی کار می‌کند و قابلیت اضافه کردن ویژگی[۶]های جدید به آن وجود دارد.
  ۲. نرم‌افزار کار می‌کند اما قابلیت کم یا زیاد کردن ویژگی به آن وجود ندارد. اصطلاحا به این سیستم، سیستم «مُرده» می‌گویند.
  ۳. نرم‌افزار کار نمی‌کند.

  نکته: نباید پشتیبانی را با نگهداری اشتباه کرد. در پشتیبانی تغییری در کد داده نمی‌شود و هدف از آن بالا نگه داشتن سیستم است. برای مثال موارد

  - logging
  - نسخه‌ی بازگردانی[۷]
  - نسخه‌ی پشتیبان[۸]
  - و...

  از جمله موارد پشتیبانی به حساب می‌آیند اما در نگهداری منبع کد دست‌کاری می‌شود و برای بالا نگه داشتن سیستم، تغییر روی آن اعمال می‌شود.

- پ) در صورتی که نرم‌افزار کار کند و قابلیت اضافه کردن ویژگی‌های جدید یا تغییر در آن وجود داشته باشد، این نرم‌افزار دارای قابلیت نگهداری[۹] دارد. [۱۰] در حالت کلی ۴ نوع نگهداری داریم:

  ۱. Perfective Maintenance: در این نوع از نگهداری به سیستم قابلیت‌هایی را اضافه کنیم.
  ۲. Corrective Maintenance: در این نوع از نگهداری مشکل[۱۱] یک قابلیت را برطرف می‌کنیم.
  ۳. Adaptive Maintenance: در نوع از نگهداری سیستم را با پلتفرم جدید تطبیق و تغییر می‌دهیم.
  ۴. Preventive Maintenance: در این نوع از نگهداری برای اجتناب از بروز یک مشکل در آینده، تغییری در سیستم ایجاد می‌کنیم.

  در روش‌های گوناگون، سیاست‌های متفاوتی برای ویژگی نگهداری وجود دارد که عبارتند از:

  ۱. سیاست مقیم[۱۲]. قراردادی نوشته می‌شود تا به صورت مداوم سیستم نگهداری شود.
  ۲. سیاست on-demand. این سیاست به این معنی است که هر موقع سیستم دچار اشکال شد و یا کارفرما ویژگی جدیدی را خواست، از سیستم نگهداری می‌شود.
  ۳. سیاست دوره‌ای[۱۳]. این سیاست به این معنی است که از سیستم در بازه‌های زمانی مشخص نگهداری می‌شود.

- ت) دلیل استفاده نشدن از نرم‌افزار ما، نداشتن قابلیت نگهداری و به اصطلاح «مُرده» بودن سیستم است. چند عامل ممکن است در از بین رفتن قابلیت نگهداری سیستم، تاثیر به سزایی داشته باشند؛ این عوامل عبارتند از:

---

[۶] feature
[۷] restore
[۸] backup
[۹] maintenance
[۱۰] دشواری این کار بسیار وابسته به نوع متدولوژی و نحوه پیاده‌سازی است؛ زیرا اگر Coupling بالا و Cohesion کم باشد، کار نگهداری نرم‌افزار بسیار سخت می‌شود.
[۱۱] bug
[۱۲] resident
[۱۳] periodical

۱. در صورتی که منبع کد به صورت نابهینه و بدون در نظر گرفتن الگوهای طراحی نرم‌افزار[۱۴] و bad smellها پیاده‌سازی شوند، تکثیر[۱۵] در کد بسیار بالا خواهد بود. این تکثیر باعث بالا رفتن coupling و در نتیجه پایین آمدن cohesion می‌شود.

۲. نوع قرارداد و متدولوژی در مرحله ایجاد و توسعه نرم‌افزار. ممکن است کارفرما با توجه به شرابطی که وجود دارد، پروژه و محدودیت‌های زمانی و مالی را طوری تعریف کند که ویژگی نگهداری برایش اهمیت کم‌تری داشته باشد.

۳. وجود کلاس God. به کلاسی کلاس God گفته می‌شود که ویژگی‌ها و عمل‌کردهای زیادی در آن پیاده‌سازی شود در حالی که آن کلاس قابل شکسته شدن به کلاس‌های کوچک‌تر باشد.

۴. وجود دید ترایا. [۱۶] اگر همه‌ی دیدها در روابط به صورت دوطرفه تعریف شود در حالی‌که به آن نیاز نیست، باعث به وجود آمدن این عارضه می‌شود.

همان‌طور که مشخص است، ویژگی نگهداری را می‌توان به عنوان مهم‌ترین ویژگی در ایجاد و توسعه نرم‌افزار محسوب کرد؛ به همین دلیل، در اکثر متدولوژی‌ها بیش‌ترین هزینه و زمان صرف محقق کردن این ویژگی می‌شود.

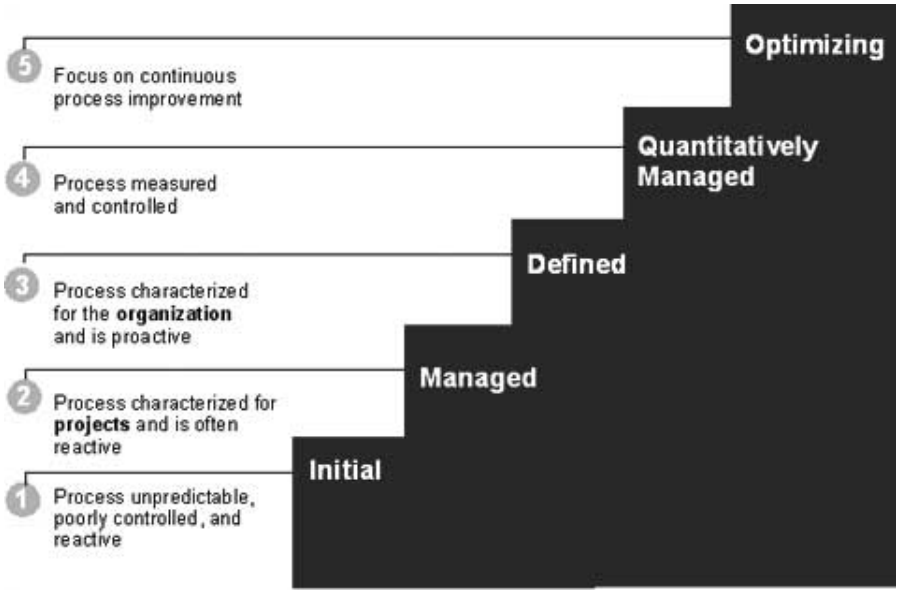
---

[۱۴] pattern design
[۱۵] duplication
[۱۶] Chain Message

سؤال ۳.

شرکت‌ها و به‌ویژه شرکت‌های نرم‌افزاری هم مانند انسان‌ها مراحل رشد و شکوفایی دارند. برای بررسی این بلوغ از روش‌هایی مانند CBA-IPI که در فصل سوم کتاب مطرح شده، استفاده می‌شود. لطفا با انجام تحقیق کوچکی مراحل مختلف بلوغ سازمان‌ها را در این روش به اختصار مشخص کنید.

سطح بلوغ یک تکامل خوش‌تعریف[۱۷] جهشی رو به جلو برای دست‌یابی به یک فرآیند نرم‌افزاری بالغ است. هر سطح بلوغ یک لایه‌ای را در پایه برای بهبود مداوم فرآیند فراهم می‌کند.



شکل ۱: مراحل بلوغ در سازمان‌ها

در مدل CMMI [۱۸] از پنج سطح بلوغ تشکیل شده است:

۱. سطح اولیه[۱۹]

۲. سطح مدیریت شده

۳. سطح تعریف شده

۴. سطح مدیریت شده از نظر کمّی[۲۰]

۵. سطح بهینه‌سازی

حال هر سطح از سطوح بالا را به‌طور دقیق‌تر بررسی می‌کنیم:

- **سطح اولیه**

در این سطح فرآیندها معمولا نامشخص هستند. سازمان معمولا محیط پایداری را فراهم نمی‌کند و موفقیت در این سازمان‌ها ببیش‌تر ستگی به رقابت افراد و اعضای و نه استفاده از فرآیندهای اثبات شده دارد. در این مرحله بعضی اوقات محصول یا سرویسی تولید می‌شود اما قیمت تمام‌شده‌ی آن‌ها معمولا بالاتر از بودجه‌ی اختصاص داده شده است.

- **سطح مدیریت شده**

در این سطح سازمان به همه‌ی اهداف کلی و خاص رسیده است. به بیان دیگر، پروژه‌های سازمان تضمین کرده‌اند که نیازمندی‌ها مدیریت می‌شوند و هم‌چنین فرآیندها نیز برنامه‌ریزی، انجام، اندازه‌گیری و کنترل می‌شوند.

در این مرحله تعهدات بین سهام‌داران ایجاد شده و در صورت نیاز مورد بازبینی و تجدید نظر قرار می‌گیرد.

---

[۱۷] well-defined
[۱۸] Integration Model Maturity Capability
[۱۹] Initial
[۲۰] Quantitatively Managed

- **سطح تعریف شده**

در این سطح فرآیندها به خوبی مشخص و درک شده و در قالب استانداردها، رویه‌ها[۲۱]، ابزارها و روش‌ها بیان می‌شوند. تفاوت اصلی و مهم بین سطح شماره‌ی «۳» و «۲» در محدوده استانداردها، توضیحات فرآیندها و رویه‌ها می‌باشد. در سطح «۲» استانداردها، توضیحات فرآیند و رویه‌ها ممکن است در هر نمونه خاص از فرآیند (مانند یک پروژه خاص) کاملا متفاوت باشند اما در مورد «۳» این تفاوت وجود ندارد و سازمان یک‌پارچه است.

- **سطح مدیریت شده از نظر کمّی**

در این سطح فرآیندهای فرعی که به‌طور قابل توجهی در عملکرد [۲۲]کلی فرآیند تاثیرگذارند، انتخاب می‌شوند. این فرآیندها با استفاده از روش‌های آماری و دیگر تکنیک‌های اندازه‌گیری کمّی کنترل می‌شوند. اهداف کمّی به عنوان معیاری برای مدیریت فرآیندها ایجاد می‌شوند. این اهداف براساس نیازهای مشتری، کاربران نهایی[۲۳]، سازمان‌ها و مجریان فرآیند هستند.

تفاوت اصلی این سطح با سطح شماره‌ی «۳» «قابلیت پیش‌بینی عمل‌کرد فرآیندها» است.

- **بهینه‌سازی**

در این سطح فرآیندها به‌طور مداوم بر اساس درک کمّی از دلایل مشترک تغییر ذاتی فرآیندها بهبود می‌یابند و بیش‌تر تمرکز آن بر روی پیشرفت ادامه‌پذیر عمل‌کرد فرآیند از طریق کارهای تکراری_افزایشی[۲۴] است.

این سطح در مقایسه با سطح «۴» در این مورد تفاوت دارند که در آن مشکلی برای کافی بودن عوامل برای دست‌یابی به نتایج اهداف وجود ندارد

نکته هر کدام از سطوح گفته شده ضروری هستند و نباید از آن‌ها پرش کرد.[۲۵] در صورت پرش کردن، هر چه سطح بالاتر باشد، احتمال رسیدن به موفقیت کم‌تر می‌شود.

---

[۲۱] procedure
[۲۲] performance
[۲۳] users end
[۲۴] incremental and iterative
[۲۵] skip

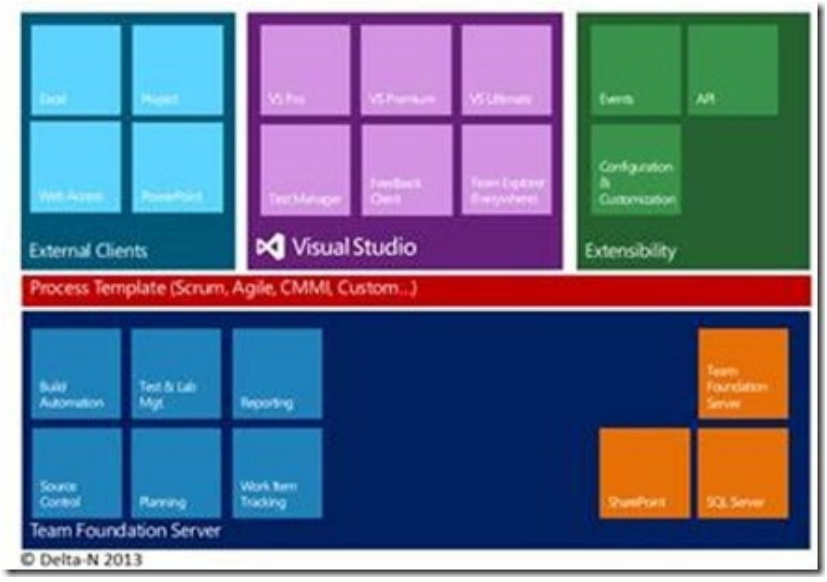
سؤال ۴.

Agile مفهومی است که این روزها در دنیای مهندسی نرم‌افزار خیلی مطرح می‌شود. شما با مفاهیمی مانند Scrum و XP در کلاس کمی آشنا شده‌اید. در شرکت‌هایی که از Scrum استفاده می‌کنند، معمولا از ابزاری مانند tfs برای مدیریت مراحل مختلف در فرآیند استفاده می‌شود. لطفا در خصوص این نرم‌افزار تحقیق کنید و نتایج آن را مختصرا گزارش دهید. تحقیقتان در حیطه مباحثی باشد که در کلاس مطرح شده است. یعنی ببینید موارد مطرح شده در کلاس در نرم‌افزار چگونه دیده شده است.

TFS یک ابزار مدیریت چرخه‌ی حیات نرم‌افزار [۲۶]است.
در واقع TFS مجموعه‌ای از ابزارهای مختلف تعاملی برای تولید نرم‌افزار را فراهم می‌کند که همه‌ی تیم ایجاد و توسعه نرم‌افزار از آن استفاده می‌کنند. این ابزارها شامل

- کنترل نسخه[۲۷]
- ابزارهای گزارش‌دهی
- مدیریت نیازمندی‌ها
- مدیریت توزیع‌ها[۲۸]
- و...

است. به طور کلی با توجه به مطالب گفته شده می‌توان این نتیجه را گرفت که از آن می‌توان به عنوان یک backend استفاده کرد. کاربرد اصلی آن در مستقرسازی[۲۹] است که مفهومی به نام DevOps در آن بسیار پررنگ است.



شکل ۲: نمای سطح بالای TFS

همان‌طور که در شکل شماره «۲» مشاهده می‌شود (بدون پرداختن به هر جنبه به‌طور مستقل)، TFS کاربردهای بسیار زیادی دارد و اساس بسیاری از موارد است. این اجازه را به ما می‌دهد تا منابع، مشکلات، برنامه‌ریزی و ساختن تست و... را به‌خوبی مدیریت کنیم. مهم‌ترین ویژگی TFS توسعه‌پذیر [۳۰] آن است که آن را فوق‌العاده می‌کند؛ به این صورت که هر چیزی که وجود ندارد، یا می‌تواند ساخته شود و یا می‌تواند تغییر کند. این دقیقا همان چیزی است که استفاده از TFS را برای سازمان‌ها مناسب می‌کند. ویژگی دیگر آن تنظیم‌پذیری [۳۱]و گستردگی آن است

[۲۶] Application Lifecycle Management (ALM)
[۲۷] version control
[۲۸] release management
[۲۹] deployment
[۳۰] extensibility
[۳۱] adjustability

که هم می‌تواند تبدیل به نقطه قوت و هم نقطه ضعف آن شود؛ زیرا این ابزارها خاص منظوره هستند و عمل‌کرد آن‌ها باید بسته به کاربردشان مقایسه و بررسی شود.
نحوه‌ی کار TFS بدین صورت است که با process templateها کار می‌کند. این قالب[۳۲]ها از بخش‌های مختلفی شامل

- مورد کاری[۳۳]
- گزارش‌ها
- مستندات
- نوع[۳۴]ها
- و...

است و اجباری در انتخاب فرآیند انتخاب شده ندارد و دست مهندس نرم‌افزار را باز می‌گذارد.
TFS می‌تواند در هر متدولوژی‌ای استفاده شود. بسته به نوع متدولوژی‌ای که از آن استفاده می‌کنیم، مورد کاری تعریف می‌شود. این انعطاف‌پذیری زیاد باعث می‌شود که ساختار همه‌ی آن‌ها یکسان باشد.

---
[۳۲] template
[۳۳] work item
[۳۴] type

سؤال ۵.

تصور کنید که پس از اتمام تحصیل تصمیم به راه‌اندازی شرکتی نرم‌افزاری با دوستانتان گرفته‌اید. ساختار این شرکت از لحاظ نیروی انسانی به چه شکلی است؟ چه نیروهایی دارد؟ می‌توانید برای پاسخ‌گویی بهتر حوزه فعالیت شرکت را به دل‌خواه پیش‌نهاد دهید. مثلا بگویید شرکت بناست در حوزه تولید نرم‌افزارهای مالی فعالیت کند لذا این نیروها را خواهد خواست.

- الف) ساختار تیم نیروی انسانی به یکی از ۴ شکل زیر است:

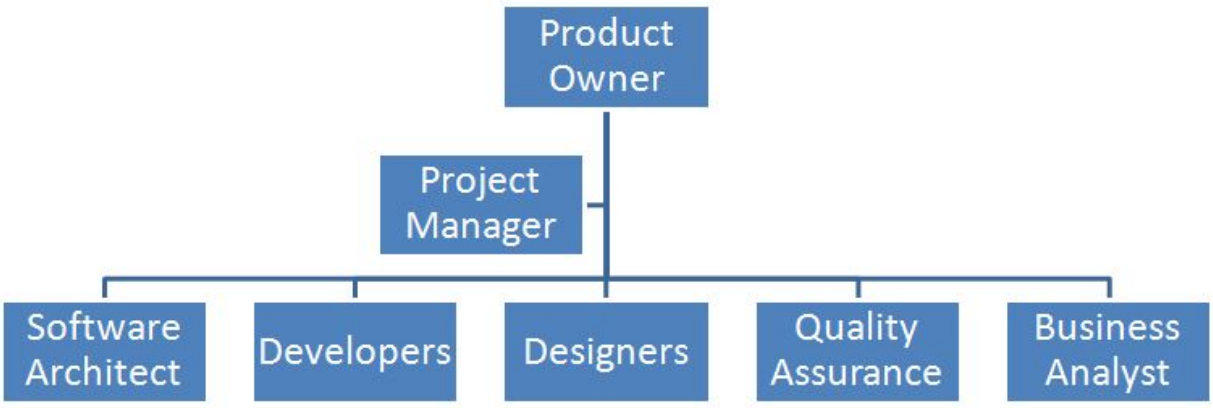
۱. الگوی بسته [۳۵] در این الگو، تیم به‌طور شکل سنتی و سلسله‌مراتبی دارد. این‌گونه تیم‌ها در هنگام تولید و ایجاد نرم‌افزار که شبیه کارهای گذشته باشد، به خوبی کار می‌کند. اما احتمال نوآورانه بودن آن در این الگوی بسته بسیار پایین است.

۲. الگوی تصادفی [۳۶] د در این الگو یک تیم به نوآوری و ابتکار فردی اعضای تیم وابستگی زیادی دارد. این الگو وقتی مزیت دارد که دست‌یابی به یک نوآوری یا موقعیت تکنولوژیکی مورد نیاز است. اما این متدولوژی هنگامی که عمل‌کرد منظم مطرح و ضروری می‌شود، عملکرد خوبی ندارد.

۳. الگوی باز[۳۷] در این الگو، سعی در ساختاربندی تیم به شکلی داریم که به برخی از کنترل‌های مرتبط با الگوی بسته دست یابد؛ هم‌چنین نوآوری‌های زیادی هنگام استفاده از الگوی تصادفی اتفاق می‌افتند. کارها با همکاری مشترک، ارتباطات سنگین و تصمیم‌گیری مبتنی بر تصمیم مشترک در مورد علائم تجاری تیم‌های با الگوی باز انجام می‌شود. ساختار تیم الگوی باز برای بسیاری از مسائل پیچیده مناسب است اما ممکن استبه اندازه سایر تیم‌ها کارآمد نباشد.

۴. الگوی هماهنگ [۳۸] این الگو متکی‌به تقسیم طبیعی یک مسئله است و اعضای تیم را برای داشتن ارتباط فعال با یک‌دیگر کمک می‌کند.

- در این سوال فرض می‌کنیم از متدولوژی چابک برای تشکیل روند تیم استفاده کنیم. استفاده از این متدولوژی باعث می‌شود تیم ساختار سلسله مراتبی داشته باشد و نقش‌ها در آن به شکل زیر است:



شکل ۳: ساختار تیم ایجاد و توسعه

- در این سوال فرض شده است که می‌خواهیم یک سامانه حمل‌ونقل بار بین شهری ایجاد کنیم. به همین منظور در متدولوژی Agile، نقش‌های مختلفی در تیم وجود دارد. برای مثال صاحب محصول[۳۹] همان مشتری است که با ما برای انجام پروژه در ارتباط است،

[۳۵] closed paradigm
[۳۶] random paradigm
[۳۷] open paradigm
[۳۸] synchronous paradigm
[۳۹] Product Owner

محدودیت‌های مالی و زمانی را تعیین می‌کند و مطابق پروپوزال او، باید برنامه‌های تعیین نیازمندی‌های پروژه استخراج شود و سپس برنامه‌ی مدونی به او داده شود.

سؤال ۶.

در فصل هفتم کتاب اصولی کاربردی معرفی شده است. ۵ اصل از این اصول که برایتان جالب است و فکر می‌کنید در آینده مورد استفاده قرار خواهید داد را نام برده و کاربرد آن را برای خودتان توضیح دهید.

از آن‌جایی که این اصول در ۵ دسته آورده شده‌اند، ترجیح می‌دهم برای پاسخ به این سوال و مطالعه‌ی همه‌ی اصول، از هر دسته، یک مورد را انتخاب کرده و توضیح دهم.

۱. **دسته ارتباط ــ اصل شنیدار**

باید سعی کنیم که تمرکز خود را به کلمات کسی که صحبت می‌کند، به جای فکر کردن به نحوه‌ی بیان جوابمان به آن صحبت‌ها، قرار دهیم.

۲. **دسته برنامه‌ریزی ــ اصل واقع‌گرایی**

باید این واقعیت را بپذیریم که آدم‌ها همه‌ی طول روز را کار نمی‌کنند و بازده صددرصد معنایی ندارد. همیشه اختلال‌هایی وارد روابط هر انسانی می‌شود. غفلت و ابهام از واقعیت‌های زندگی هستند و اتفاق نیز رقم می‌خورد. حتی بهترین مهندسین نرم‌افزار هم اشتباه می‌کنند. این واقعیات باید هنگام برنامه‌ریزی برای پروژه لحاظ شوند.

۳. **دسته مدل‌سازی ــ اصل هدف قرار دادن تولید نرم‌افزار و نه مدل**

چابکی[۴۰] به معنای رساندن نرم‌افزار به مشتری در سریع‌ترین زمان ممکن است. مدل‌ها ارزش ساخته شدن دارند اما اگر باعث کند شدن فرآیند نرم‌افزار بشوند باید از وسواس به خرج دادن در مورد آن‌ها پرهیز کرد.

۴. **دسته تولید و ساخت ــ اصل اعتبارسنجی**

- در صورت لزوم باید کد را بررسی کرده و مورد کنکاش قرار دهیم.
- unit test انجام دهیم و خطاهایی که بعد از این آزمون نمایان می‌شوند را برطرف کنیم.
- کد را بازآرایی[۴۱] کنیم.

۵. **دسته مستقرسازی ــ نرم‌افزارهای مشکل‌دار[۴۲] اول باید درست و سپس تحویل داده شوند.**

زمانی که از لحاظ زمانی تحت فشار هستیم، بعضی از شرکت‌های نرم‌افزاری نرم‌افزارهای با کیفیت کم و این وعده که مشکل‌ها را در نسخه‌ی بعدی برطرف می‌کنند، تحویل می‌دهند. این یک اشتباه بسیار بزرگی است. یک جمله‌ی معروف در نرم‌افزار هست که می‌گوید: «مشتری چند روزی را که به خاطر تحویل دادن نرم‌افزار با کیفیت بالا تاخیر می‌کنید، فراموش می‌کند اما هرگز مشکلات نرم‌افزار کم کیفیت را فراموش نمی‌کند و نرم‌افزار این موضوع را هر روز به آن‌ها یادآوری می‌کند.»

---

[۴۰] agility
[۴۱] refactor
[۴۲] buggy

سؤال ۷.

فرض کنید بعد از موضوع کرونا و ماجراهایش در کشور ما هم مجوز داروخانه صادر شده است. لذا دانشجویان کلاس مهندسی نرم‌افزار هم به این فکر افتادند که چنین نرم‌افزاری ایجاد نمایند. این نرم‌افزار نسخه تحت وب و اپلیکیشن خواهد داشت. نیازمندی‌های این سیستم را با روش‌های مختلفی که در فصل‌های مربوط به نیازمندی‌ها آموخته‌اید استخراج کرده و مستند کنید.

با توجه به این که این سوال به صورت خیلی کلی طرح شده است، فرضیات مربوط به این سوال و سوال بعدی را در این قسمت نوشته و توضیح خواهم داد و نمودارها و نیازمندی‌ها با لحاظ این فرضیات نوشته و کشیده خواهند شد. [۴۳]

- **نیازمندی‌ها[۴۴] و موارد کاربرد[۴۵]**

یک نرم‌افزار تحت وب و اپلیکیشن برای داروخانه طراحی می‌شود که در آن:

۱. بیماران می‌توانند در سامانه ثبت‌نام کنند.
۲. بیماران باید بتوانند وارد سامانه شوند.
۳. متصدیان داروخانه باید بتوانند وارد سامانه شوند.
۴. متصدیان داروخانه امکان اضافه کردن داروی جدید را دارند.
۵. دارو شامل جزییاتی مانند نام عمومی دارو، نام علمی دارو، نحوه‌مصرف، شکل و... هستند. این اطلاعات باید قابل مشاهده باشند.
۶. بیماران می‌توانند برگه‌های دفترچه بیمه و مربوط به داروهای خود را آپلود کنند.
۷. متصدیان داروخانه برگه‌های دفترچه بیماران را بررسی می‌کنند.
۸. امکان خرید آنلاین داروهای آزاد وجود دارد.
۹. امکان جست‌وجو در لیست داروها باید پیاده‌سازی شود.
۱۰. کاربران باید بتوانند از سیستم خارج شوند.

در بخش تحلیل نمودارهای زیادی وجود دارند که می‌توان آن‌ها را رسم کرد و باعث ارتباط بهتر با مشتری می‌شود. در واقع در بخش فنی تنها در محدوده قلمرو مسئله[۴۶] هستیم و کاری به جزئیات فنی مانند پایگاه‌داده و ... نداریم.

برخی نمودارهایی که در این بخش می‌توان رسم کرد عبارتند از:

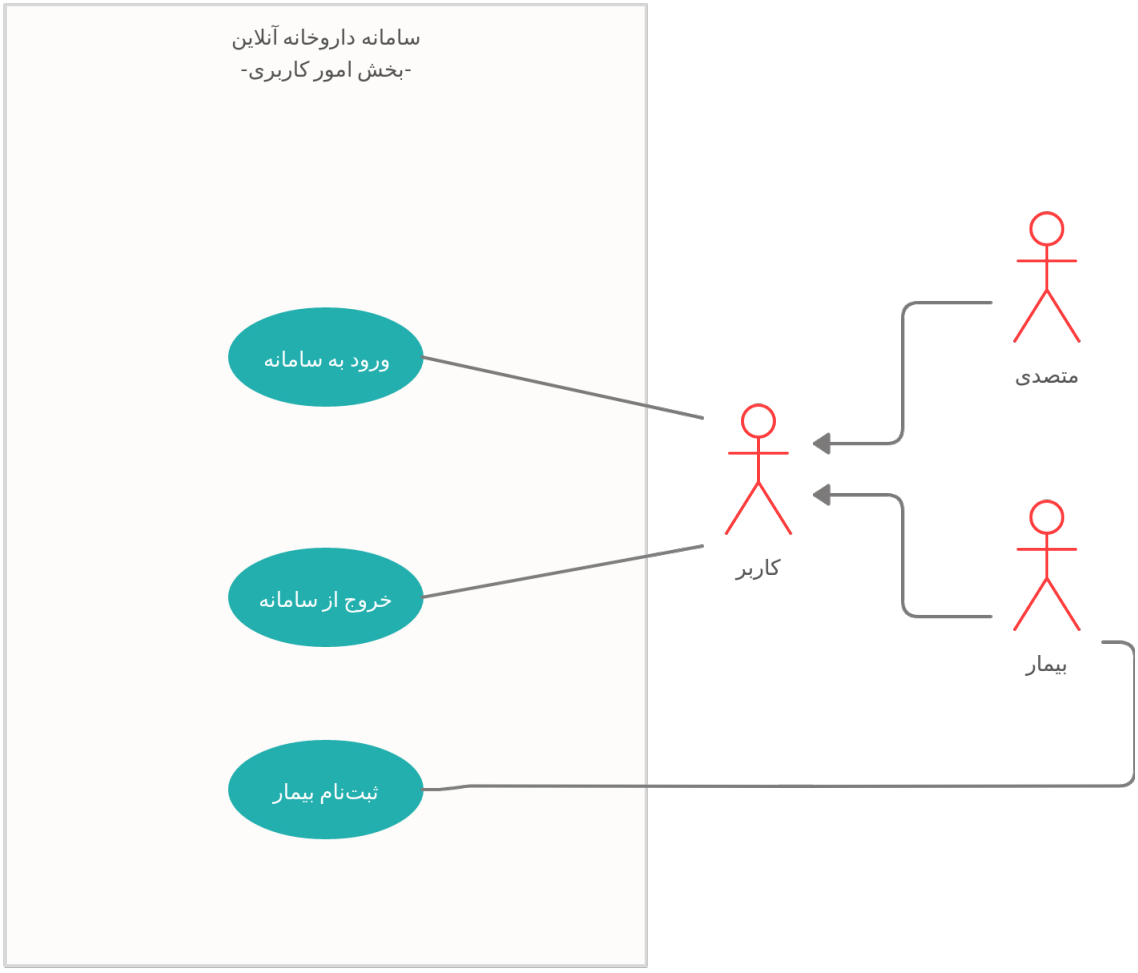
- نمودار مورد کاربرد
- نمودار CRC[۴۷]
- نمودار کلاس‌ها (بدون جزئیات فنی)
- و...

- **نمودار مورد کاربرد**

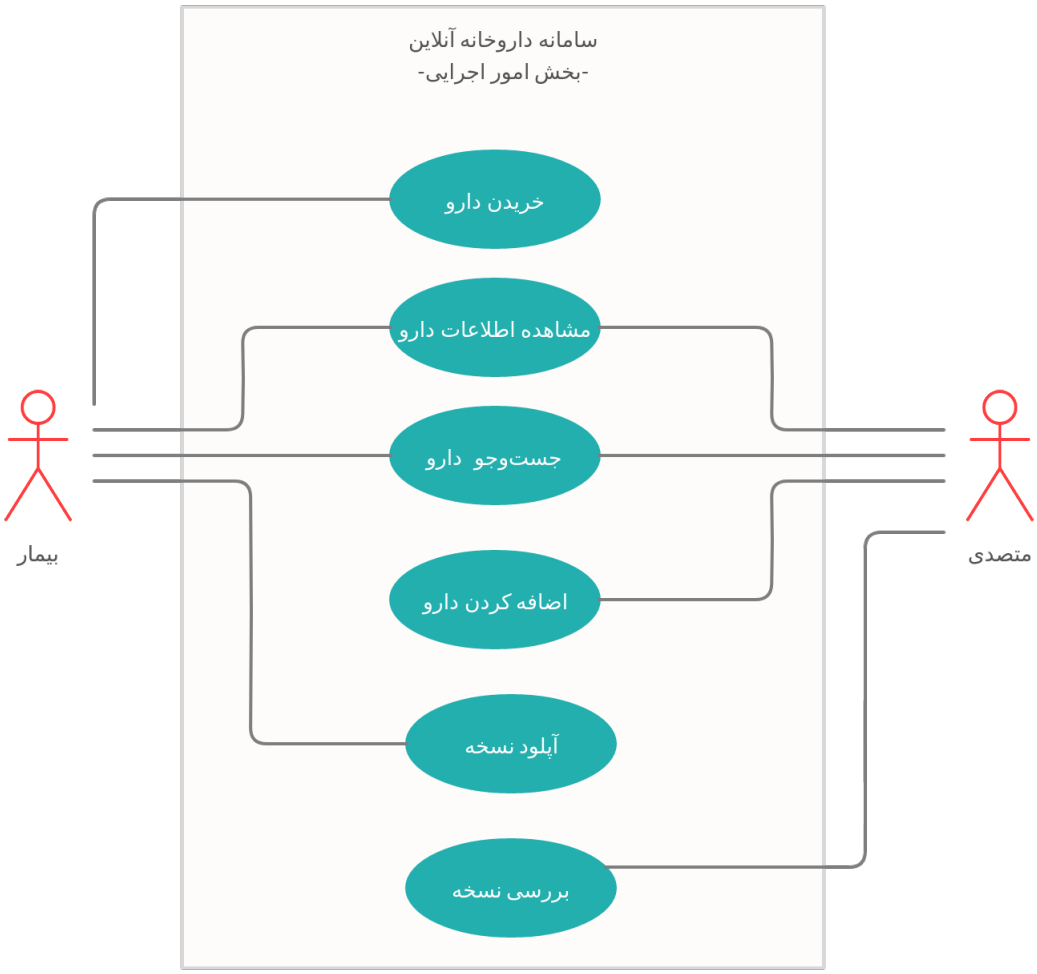
در این نمودار ارتباط عوامل خارجی که به آن‌ها کنش‌گر[۴۸] گفته می‌شود با موارد کاربرد نمایش داده شده است. موارد کاربرد حتما باید خاصیت atomic داشته باشند؛ به این معنی که قابل شکسته شدن به مورد کاربردهای کوچک‌تر نباشند.

---

[۴۳] از آن‌جایی که کشیدن مدل‌های تحلیلی پیچیده در زمان امتحان امکان‌پذیر نیست، تلاش شده است تا حد امکان ساده و کاربردی پیاده‌سازی شوند.
[۴۴] requirements
[۴۵] use-case
[۴۶] domain problem
[۴۷] در این نمودار رابطه‌ی بین مسئولیت‌ها و کلاس‌ها و نحوه‌ی همکاری آن‌ها نمایش داده می‌شود. منظور از
[۴۸] actor



شکل ۴: نمودار مورد کاربرد بخش امور کاربری



شکل ۵: نمودار مورد کاربرد بخش امور اجرایی

سؤال ۸.

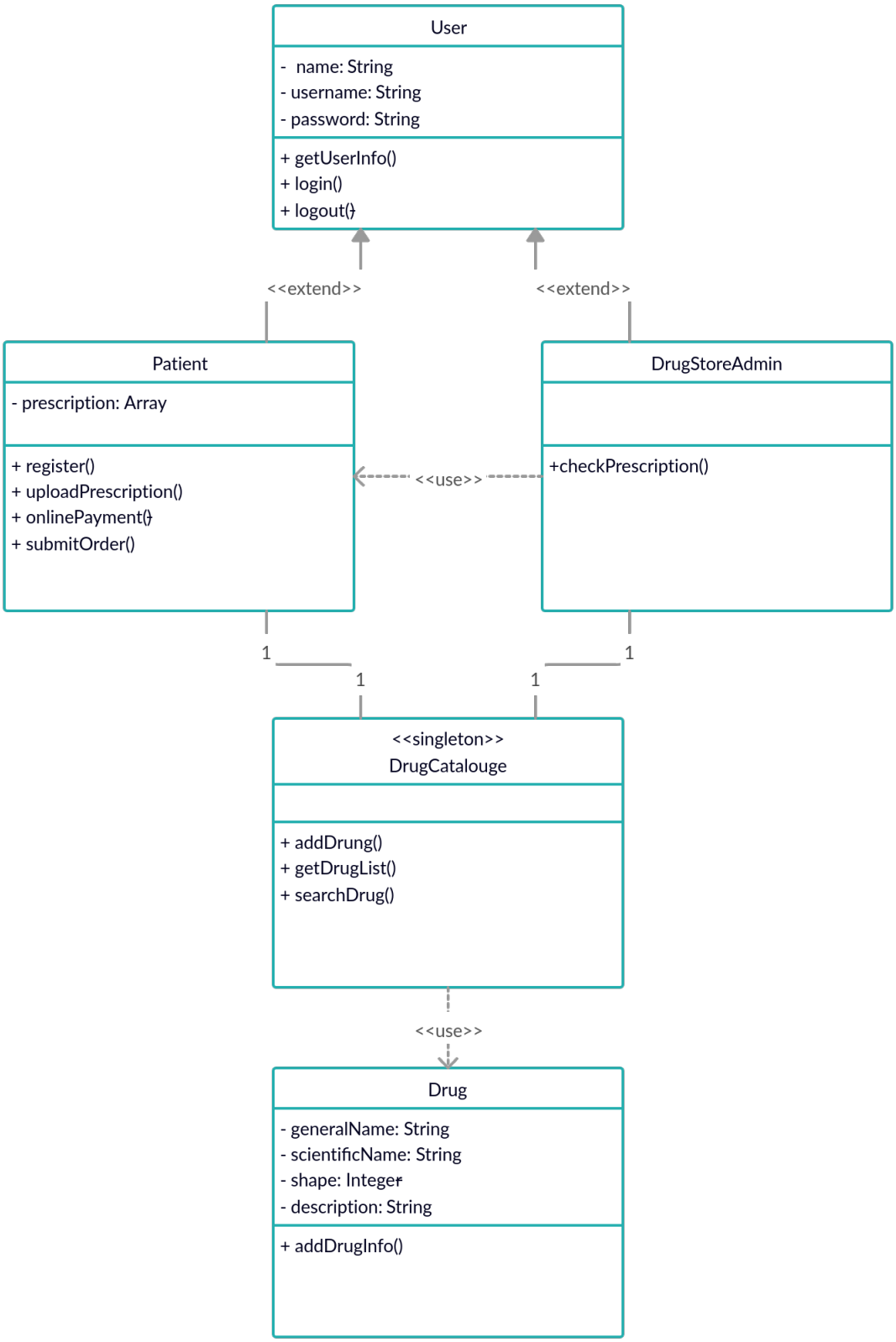
با توجه به مباحثی که در فصل‌های Design آموخته‌اید، نرم‌افزار را طراحی نمایید.
تفاوت اصلی بخش تحلیل و طراحی مربوط به قلمرو آن‌ها است. همان‌طور که در سوال «۷» گفتیم، محدوده‌ی تحلیل در قلمرو مسئله و محدوده‌ی طراحی در قلمرو جواب [۴۹] است. این بدان معنی است که جزئیات فنی نیز در این بخش مانند پایگاه‌داده، تکنولوژی‌های مورد استفاده و... در نظر گرفته می‌شوند.
نمودارهای زیادی در بخش طراحی وجود دارند:

- نمودار کلاس با جزئیات فنی
- نمودار مستقرسازی[۵۰]
- نمودار مؤلفه [۵۱]
- نمودار توالی [۵۲]
- و...

با توجه به محدودیت زمان امتحان، تصمیم گرفته‌ام تا نمودار کلاس را تا حد امکان ساده و کاربردی پیاده‌سازی کرده و وارد جزئیات بیش از حد نشوم.

- **نمودار کلاس**

[۴۹] solution domain
[۵۰] deployment diagram
[۵۱] component diagram
[۵۲] sequence diagram



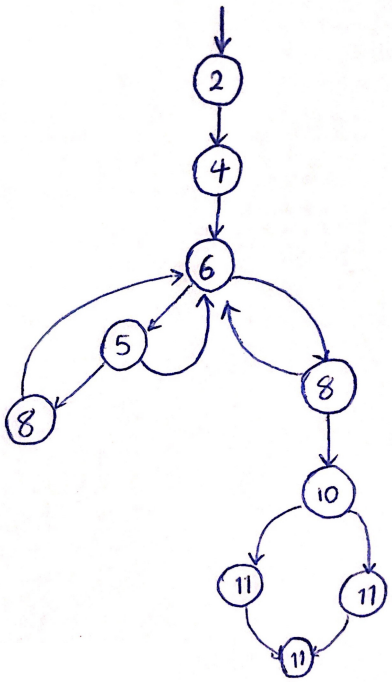
شکل ۶: نمودار کلاس بخش طراحی

سؤال ۹. برنامه‌ای بنویسید که آرایه‌ای را از ورودی گرفته و آن را به ترتیب صعودی یا نزولی به انتخاب کاربر مرتب کند.

```python
def sort(input_array, ascending=True):
    input_array_len = len(input_array) - 1
    outer: while True:
        is_swapped = False
        inner: for i in range(input_array_len):
            if input_array[i] > input_array[i + 1]:
                #swappng
                is_swapped = True
                input_array[i], input_array[i+1] = input_array[i+1], input_array[i]
        if not is_swapped:
            break outer
    return input_array if ascending else input_array.sort(reverse=True)
```

شکل ۷: کد برنامه

- **گراف ISP**



شکل ۸: گراف ISP این گراف براساس شماره‌ی خطهای کد ساخته شده است.

- **پوشش گره‌ها**[۵۳]

$$NodeCoverage = \{[2, 4, 6, 5, 8, 8, 10, 11, 11], [2, 4, 6, 8, 10, 11, 11]\}$$

- **بررسی راه و پوشش**

به دلیل وجود حلقه امکان‌پذیر نیست.

[۵۳] node coverage

- **پوشش یال‌ها**[۵۴]

$EdgeCoverage : [2, 4, 6, 5, 8, 6, 8, 10, 11, 11], [2, 4, 6, 5, 6, 8, 6, 8, 10, 11, 11]$

- Predicate شامل موارد زیر است:
  -

$input\_array[i] > input\_array[i + 1]$

  -

$is\_swapped$

  -

$ascending$

- عبارت‌ها [۵۵] همان موارد قسمت قبل هستند که به آن‌ها مقدار True و یا False اختصاص گرفته است. بنابراین در کل ۸ حالت داریم.

---

[۵۴] Coverage Edge
[۵۵] clauses