به نام خدا

میانترم دوم-پاسخنامه

مهندسی نرم‌افزار – نیمسال اول ۴۰۲

- تاریخ آزمون: پنج‌شنبه ۱۴ / ۱۰ / ۴۰۲
- زمان شروع: ۹:۰۰ صبح
- زمان پایان: ۱۴:۰۰ ظهر

دانشکده مهندسی کامپیوتر – دانشگاه شریف

مدرس: دکتر مهران ریواده

بخش اول – سوالات تستی

۱-کدام یک از موارد زیر در رابطه با انتزاع (Abstraction) در طراحی نرم‌افزار درست نیست؟

a) انتزاع یکی از روش‌های اساسی برای ساده کردن سیستم‌ها یا مولفه‌های (Component) پیچیده است.
b) انتزاع سطوح متفاوتی دارد و هر چقدر از سطوح بالاتر به سطوح پایین‌تر برویم، جزئیات راه حل ارائه شده مساله، پنهان می‌ماند.
c) انتزاع رویه‌ای (Procedural Abstraction) دنباله‌ای از دستورالعمل‌ها است که عملکرد محدود و مشخصی دارند. برای مثال افزایش سرعت ماشین.
d) انتزاع داده‌ای (Data Abstraction) یک مجموعه نام‌گذاری شده از داده‌ها است که یک شی داده‌ای را تعریف می‌کند. مانند یک ماشین.

(صفحه ۲۳۲ – بخش ۱۲.۳.۱ – کتاب پرسمن ادیشن ۸)

۲- بر اساس اصول طراحی رابط کاربری Mandel برای کاهش بار حافظه کاربر، کدام یک از گزینه‌های زیر نادرست است؟

a) نشانه‌های بصری: رابط باید شامل نشانه‌های بصری برای کمک به کاربران در شناسایی اقدامات و ورودی‌های گذشته باشد تا بار حافظه کوتاه‌مدت آن‌ها کاهش یابد.
b) پیش‌فرض‌های معنادار: تنظیمات پیش‌فرض اولیه باید برای کاربر متوسط منطقی باشد، با انعطاف‌پذیری برای سفارشی‌سازی فردی و گزینه‌ای برای بازنشانی به تنظیمات پیش‌فرض اصلی.
c) میان‌برهای شهودی: طراحی رابط باید شامل میانبرهای شهودی، مانند میانبرهای حافظه که مستقیماً به عملکردهای خود مرتبط هستند، برای سهولت در به خاطر سپردن باشد.
d) نمایش اولیه جزئیات: در حالی که اطلاعات به صورت سلسله مراتبی سازماندهی می‌شود، رابط باید ابتدا سطح جامعی از جزئیات را برای ارائه زمینه کامل به کاربران قبل از ورود به وظایف خاص نمایش دهد.

(صفحه ۳۱۹ و ۳۲۰ – بخش ۱۵.۱.۲ – کتاب پرسمن ادیشن ۸)

۳- یک طراح اپلیکیشن‌های موبایل، در یکی از پروژه‌های خود برای یک دکمه از اپلیکیشن در دو صفحه متفاوت، نام‌گذاری متفاوتی قرار داده است در صورتی که هر دو دکمه یک عملکرد (Functionality) دارند، این طراح کدام یک از اشتباهات رایج در طراحی UI را انجام داده است؟

a) Overdesigning
b) Verbiage
c) Nonstandard Interaction
d) Inconsistency

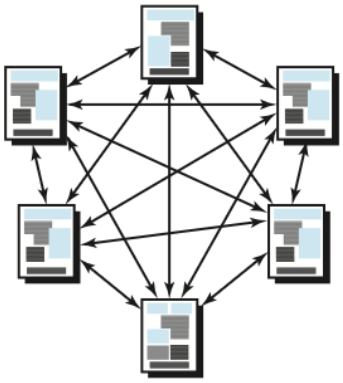
(صفحه ۴۰۱ – بخش ۱۸.۲.۴ – کتاب پرسمن ادیشن ۸)

۴- کدام یک از جملات زیر به درستی یکی از اشتباهات رایجی را که هنگام استفاده از طراحی مبتنی بر الگو در توسعه نرم‌افزار رخ می‌دهد، توصیف می‌کند؟

a) الگوها در طراحی نرم‌افزار همیشه بی‌نقص هستند و نیازی به تطبیق برای فضاهای مشکل خاص ندارند.
b) جستجو برای نظر شخص دوم یا بازبینی کار طراحی توصیه نمی‌شود زیرا ممکن است باعث سردرگمی و پیچیدگی در فرآیند طراحی شود.
c) یکی از اشتباهات رایج انتخاب الگو بدون درک کامل از مشکل و زمینه (Context) مشکل می‌باشد که باعث می‌شود الگوی انتخاب شده، به صورت کامل به همه نیازها پاسخ ندهد.
d) طراحی مبتنی بر الگو به طور ذاتی تمام پارامترها و زمینه‌های (Context) احتمالی یک مشکل را در نظر می‌گیرد، بنابراین تحلیل عمیق مشکل قبل از انتخاب الگو غیر ضروری است.

(صفحه ۳۵۹ – کتاب پرسمن ادیشن ۸)

۵- کدام گزینه در مورد ساختار زیر که به آن Pure web هم گفته می‌شود، برای Web Appها صحیح است؟



a) از جهات بسیاری شبیه معماری‌هایی است که برای سیستم‌های شیءگرا وجود دارند.
b) در این ساختار، هر صفحه وب می‌تواند کنترل را به هر صفحه دیگری بسپارد.
c) این ساختار گرچه انعطاف‌پذیری قابل توجهی در Navigation را فراهم می‌کند اما برای کاربر گیج‌کننده است.
d) همه موارد صحیح هستند.

(صفحه ۳۸۳ – بخش ۱۷.۷.۱ کتاب پرسمن ادیشن ۸)

بخش دوم – سوالات تشریحی

۶- چه تفاوتی بین الگوهای معماری با Styleهای معماری وجود دارد؟ مختصراً شرح دهید.

پاسخ:

۱. دامنه یک الگو گستردگی کمتری نسبت به Style دارد و بیشتر بر یک جنبه از معماری تمرکز دارد تا کلیت معماری
۲. الگو یک قانون را بر معماری تحمیل می‌کند و نحوه‌ی برخورد نرم‌افزار با برخی جنبه‌های عملکردی خود را در سطح زیرساخت توصیف می‌کند. (مانند همروندی)
۳. الگوهای معماری معمولاً مسائل رفتاری خاص را در چارچوب معماری بررسی می‌کنند (مانند این موضوع که برنامه‌های Real-time چگونه با همگام‌سازی یا وقفه‌ها روبرو می‌شوند).

(صفحه ۲۵۸ – بخش ۱۳.۳ - کتاب پرسمن ادیشن ۸)

۷- سناریویی را در نظر بگیرید که در آن کاربر در حال تعامل با یک برنامه موبایل جدید است که برای مدیریت امور مالی شخصی طراحی شده است. این برنامه به کاربران امکان می‌دهد هزینه ها را پیگیری کنند، بودجه را تنظیم کنند و گزارش های مالی را مشاهده کنند. با این حال، کاربران برخی از مشکلات را هنگام استفاده از برنامه گزارش کرده‌اند. بر اساس اصول طراحی Bruce Tognozzi، مشخص کنید کدام اصل(ها) ممکن است در این سناریو نقض شده باشد و دلایل آن را بیان کنید.

مشکلات گزارش شده:

- برنامه اقدامات مربوطه را پیشنهاد نمی‌کند یا مراحل بعدی کاربر را پیش بینی نمی‌کند، مانند پیشنهاد تنظیم بودجه بر اساس الگوهای هزینه‌های گذشته.
- رابط برنامه با عملکردهای بیش از حد در صفحه اصلی به هم ریخته است، که تمرکز روی یک کار واحد مانند وارد کردن هزینه‌های روزانه را دشوار می‌کند.
- همینطور کاربران جدید گزارش کرده‌اند که درک نحوه پیمایش در برنامه و استفاده از ویژگی‌های آن مشکل دارند.
- ساختار Navigation گیج کننده است بطوریکه برخی از عملکردها که در زیر چندین لایه از منوها مدفون شده‌اند و یافتن آن‌ها را سخت می‌کند.

پاسخ: اصل‌های نقض شده عبارتند از:

- Anticipation: اگر برنامه اقدامات مربوطه را پیشنهاد نکند یا مراحل بعدی کاربر را پیش بینی نکند، اصل پیش بینی را نقض می‌کند.
- Focus: رابط درهم و برهم با عملکردهای بسیار زیاد در صفحه اصلی، اصل تمرکز را نقض می کند، زیرا حواس کاربران را از وظایف اصلی آن‌ها منحرف می‌کند.
- Learnability: مشکل گزارش شده در درک Navigation و ویژگی‌های برنامه نشان دهنده نقض اصل یادگیری است.
- Visible Navigation: ساختار Navigation گیج کننده و توابع غیرقابل یافتن نشان‌دهنده نقض اصل ناوبری قابل مشاهده است.

(صفحه ۳۳۷ تا ۳۴۰ – بخش ۱۵.۵ – کتاب پرسمن ادیشن ۸)

۸- چه زمانی از Component Wrapping استفاده می‌کنیم؟ تکنیک‌های مورد استفاده در آن را مختصر توضیح دهید.

پاسخ: در حالت ایده‌آل، مهندسی دامنه، کتابخانه‌ای از Componentها ایجاد می‌کند که می‌توانند به راحتی در معماری برنامه ادغام شوند. به راحتی ادغام شدن به این معنی است که روش‌های مدیریت منابع سازگار برای همه Componentها در کتابخانه پیاده‌سازی شده‌اند، فعالیت‌هایی مانند مدیریت داده‌ها برای همه Componentها وجود دارد و رابط‌های درون معماری و با محیط خارجی به‌طور مداوم پیاده‌سازی شده‌اند. در واقع، حتی پس از اینکه یک مؤلفه برای استفاده در معماری برنامه واجد شرایط شد، ممکن است در یک یا چند قسمت از موارد ذکر شده تداخل ایجاد شود. گاهی اوقات برای جلوگیری از این تداخل‌ها از یک تکنیک تطبیقی به نام Component wrapping استفاده می‌شود. تکنیک‌های مورد استفاده در آن عبارتند از:

- وقتی یک تیم نرم‌افزاری به طراحی داخلی و کد یک Component دسترسی کامل دارد (اغلب اینطور نیست مگر اینکه از Componentهای COTS منبع باز استفاده شود)، white-box wrapping اعمال می‌شود. white-box wrapping جزئیات پردازش داخلی Component را بررسی می‌کند و تغییراتی در سطح کد برای حذف هرگونه تضاد ایجاد می‌کند.
- زمانی که کتابخانه‌ی Component یک component extension language یا API ارائه دهد، gray-box wrapping اعمال می‌شود که امکان حذف یا پوشاندن تداخل‌ها را فراهم می‌کند.

- تکنیک black-box wrapping به معرفی پیش‌پردازش و پس‌پردازش در رابط Component برای حذف یا پوشاندن تداخل‌ها نیاز دارد.

(صفحه ۳۱۰ – بخش ۱۴.۷.۲ - کتاب پرسمن ادیشن ۸)

۹- سناریوهای زیر را در نظر بگیرید. هر سناریو ممکن است اصول طراحی نرم‌افزار شامل اصل OCP، اصل LSP، اصل DIP، اصل ISP را نقض کرده باشند و یا دارای Coupling بالا یا Cohesion پایین باشند. بررسی نمایید هر یک از این سناریوها چه مشکلی دارند و چرا؟

۱. یک کلاس Animal متدی به نام makeSound دارد. کلاس Dog و کلاس Cat هر دو از Animal به ارث می‌برند. نوع جدیدی از حیوانات به نام Fish اضافه می‌شود، ولی متد makeSound برای آن استفاده نمی‌شود.
۲. یک کلاس رابط کاربری مسئول مدیریت ورودی‌های ماوس، ورودی‌های صفحه کلید، رندر کردن گرافیک و مدیریت پیام‌های شبکه است.
۳. کلاس دسترسی به پایگاه داده یک سیستم نرم‌افزاری از مصرف‌کنندگان می‌خواهد که مدیریت تراکنش، مدیریت اتصال و مدیریت خطا را پیاده‌سازی کنند، حتی اگر فقط به اجرای یک عملیات خواندن ساده نیاز داشته باشند.
٤. یک سیستم پردازش پرداخت به یک درگاه پرداخت خاص وابسته است و هر تغییری که در درگاه پرداخت ایجاد شود، تأثیر مستقیمی بر سیستم پردازش پرداخت خواهد داشت.

پاسخ:

۱. نقض اصل جایگزینی لیسکوف (LSP) است. LSP بیان می‌کند که زیرکلاس‌ها باید برای کلاس‌های پایه‌ی خود بدون تغییر در صحت برنامه قابل جایگزینی باشند. از آنجایی که "Fish" نمی‌تواند "makeSound" را به درستی پیاده‌سازی کند، LSP را نقض می‌کند.
۲. مشکل اینجا Cohesion پایین است. این کلاس مسئولیت‌های زیادی دارد که ارتباط تنگاتنگی با یکدیگر ندارند، که نشان می‌دهد طراحی کلاس متمرکز نیست و ممکن است بیش‌تر از آنچه که باید کار کند. (اشاره به SRP هم درست است، ولی از آن‌جایی که در کتاب نیامده، ما به Cohesion پایین اشاره کردیم.)
۳. این سناریو نقض اصل جداسازی رابط (ISP) را نشان می‌دهد. این اصل بیان می‌کند که هیچ کلاینتی نباید مجبور شود به متدهایی که استفاده نمی‌کند وابسته باشد. وقتی کلاینت‌ها نیاز به پیاده‌سازی روش‌های غیر ضروری دارند یعنی که رابط به اندازه کافی تفکیک نشده است.
٤. در این سناریو، اصل وارونگی وابستگی (DIP) نقض می‌شود. ماژول‌های سطح بالا، سیستم پردازش پرداخت، به ماژول‌های سطح پایین، درگاه پرداخت، وابسته‌اند. این اصل می‌گوید انتزاع‌ها نباید به جزئیات بستگی داشته باشند و جزئیات باید به انتزاعات وابسته باشند. در واقع، سیستم پردازش پرداخت باید طوری طراحی شود که با یک رابط درگاه پرداخت انتزاعی کار کند، به طوری که تغییرات در درگاه پرداخت خاص، بدون تأثیر بر بقیه سیستم انجام شود.

(صفحه ۲۹۲ تا ۲۹۸ – بخش ۱۴.۲ - کتاب پرسمن ادیشن ۸)

۱۰- در طراحی Pattern-Based زمانی که تعداد Design patternهایی که می‌خواهید از بین آن‌ها انتخاب کنید زیاد می‌شود، مرتب‌سازی به یک ضرورت تبدیل می‌شود. چه روشی برای این مرتب‌سازی و انتخاب پیشنهاد می‌کنید؟ شکل کلی روش خود را توضیح دهید.

پاسخ:

برای کمک به سازماندهی ارزیابی‌ها از الگوهای کاندید، مایکروسافت یک جدول مرتب‌سازی الگو پیشنهاد می‌کند که شکل کلی آن در جدول زیر نمایش داده شده است. همانطور که در جدول نمایش داده شده است، لیست خلاصه شده‌ای از مساله‌ها به صورت مرتب شده بر اساس داده یا محتوا، معماری، سطح کامپوننت و مشکلات رابط کاربری در سمت چپ جدول قرار گرفته و چهار نوع الگو (پایگاه داده – کاربرد – پیاده‌سازی و زیرساخت) در سطر بالای جدول لیست شده‌اند. نام الگوهای کاندید در هر یک از سلول‌های جدول بیان شده است. برای پر کردن جدول زبان‌ها و مخازن الگوهایی که هر مساله را حل می‌کنند، جست‌و جو می‌شود. زمانی که یک یا چند الگوی کاندید پیدا شد، در سلول مطابق با سطر مساله و ستون مربوط به نوع الگو وارد می‌شود و بر روی نام آن لینکی به توضیحات کامل آن زده می‌شود.

	Database	Application	Implementation	Infrastructure
Data/Content				
Problem statement ...	PatternName(s)		PatternName(s)	
Problem statement ...		PatternName(s)		PatternName(s)
Problem statement ...	PatternName(s)			PatternName(s)
Architecture				
Problem statement ...		PatternName(s)		
Problem statement ...		PatternName(s)		PatternName(s)
Problem statement ...				
Component-level				
Problem statement ...		PatternName(s)	PatternName(s)	
Problem statement ...				PatternName(s)
Problem statement ...		PatternName(s)	PatternName(s)	
User interface				
Problem statement ...		PatternName(s)	PatternName(s)	
Problem statement ...		PatternName(s)	PatternName(s)	
Problem statement ...		PatternName(s)	PatternName(s)	

در صورتی که پیشنهاد صحیح دیگری برای این سوال بیان شود، قابل قبول است.

(صفحه ۳۵۸ – بخش ۱۶.۲.۴ - کتاب پرسمن ادیشن ۸)

موفق باشید

تیم آموزش مهندسی نرم‌افزار