

سوالات تستی

جواب سوالات تستی

1. B
2. D
3. D
4. A
5. D

سوال ۱

چه تفاوتی بین الگوهای معماری با style های معماری وجود دارد؟ مختصراً شرح دهید.

جواب سوال ۱

(Architectural Patterns) الگوهای معماری

الگوهای معماری، راه‌حل‌های تجربه شده و آزموده شده‌ای هستند که برای حل مشکلات معماری خاص در طراحی نرم‌افزار به کار می‌روند. این الگوها شامل دستورالعمل‌ها و رهنمودهایی برای توزیع مسئولیت‌ها در بین اجزای نرم‌افزار هستند. مثال‌هایی از الگوهای معماری شامل معماری سه لایه (Three-Tier Architecture)، مدل-نما-کنترلر (MVC)، و میکروسرویس‌ها (Microservices) هستند.

(Architectural Styles) Style های معماری

Style های معماری، بیشتر بر روی اصول و مفاهیم کلی در طراحی سیستم‌های نرم‌افزاری تمرکز دارند و کمتر به جزئیات پیاده‌سازی می‌پردازند. آن‌ها چارچوب کلی‌تری برای درک و بیان ساختار یک سیستم فراهم می‌کنند. مثال‌هایی از Style های معماری شامل سرویس‌گرا (SOA)، رویداد محور (Event-Driven)، و منشوری (Layered) هستند.

تفاوت‌های کلیدی:

الف) محدوده کاربرد: الگوهای معماری اغلب با جزئیات بیشتری برای حل مشکلات مشخص در نرم‌افزار تعریف می‌شوند، در حالی که Style های معماری مفاهیم کلی‌تر و چارچوب‌های فکری را ارائه می‌دهند.

- ب) جزئیات و دستورالعمل‌ها: الگوهای معماری معمولاً دستورالعمل‌های مشخص‌تری برای پیاده‌سازی دارند، در حالی که Style های معماری بیشتر به بیان اصول و مفاهیم پایه‌ای می‌پردازند.
- ج) انعطاف‌پذیری: Style های معماری اغلب انعطاف‌پذیری بیشتری برای تطبیق با شرایط مختلف دارند، در حالی که الگوهای معماری ممکن است در شرایط خاصی محدودیت‌هایی داشته باشند.
- د) تمرکز بر کیفیت‌های سیستم: Style های معماری تمرکز بیشتری بر کیفیت‌های کلی سیستم مانند قابلیت اطمینان، امنیت، و قابلیت استفاده دارند، در حالی که الگوهای معماری اغلب بر حل مسائل فنی و معماری تمرکز دارند.
- ه) تطبیق‌پذیری و مقیاس‌پذیری: در حالی که الگوهای معماری ممکن است در پیاده‌سازی‌های خاص محدودیت‌هایی داشته باشند، Style های معماری اغلب اجازه می‌دهند که سیستم‌ها با تغییرات تکنولوژیکی یا نیازهای تجاری به راحتی تطبیق یابند.

سوال ۲

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 هزینه‌ها را پیگیری کنند، بودجه را تنظیم کنند و گزارش‌های مالی را مشاهده کنند. با این حال، کاربران برخی از مشکلات را هنگام استفاده از برنامه گزارش کرده‌اند. بر اساس اصول طراحی Bruce Tognazzini، 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 گیج‌کننده است بطوریکه برخی از عملکردها که در زیر چندین لایه از منوها مدفون شده‌اند و یافتن آنها را سخت می‌کند.

جواب سوال ۲

بر اساس اصول طراحی Bruce Tognazzini، چندین اصل ممکن است در این سناریو نقض شده باشند:

- الف) قابلیت پیش‌بینی (Predictability): برنامه باید توانایی پیش‌بینی نیازهای کاربر و ارائه پیشنهادات مفید را داشته باشد. نبود این ویژگی در برنامه نشان‌دهنده نقض این اصل است.
- ب) سادگی رابط کاربری (Simplicity): رابط کاربری باید ساده و قابل فهم باشد. ازدحام عملکردها در صفحه اصلی موجب پیچیدگی و دشواری در استفاده می‌شود که نقض این اصل محسوب می‌شود.
- ج) قابلیت پیمایش (Navigability): کاربران باید بتوانند به راحتی در برنامه حرکت کنند و به ویژگی‌های مختلف دسترسی داشته باشند. گیج‌کننده بودن ساختار Navigation نشان‌دهنده نقض این اصل است.

سوال ۳

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

جواب سوال ۳

Component Wrapping یک تکنیک در مهندسی نرم‌افزار است که برای ادغام کامپوننت‌های موجود در یک سیستم جدید یا برای افزایش سازگاری بین کامپوننت‌های مختلف استفاده می‌شود. از Component Wrapping عمدتاً در موقعیت‌های زیر استفاده می‌شود:

- **انطباق با معماری‌های جدید:** زمانی که نیاز است تا کامپوننت‌های موجود در یک معماری جدید بدون تغییر عمده کد اصلی قرار گیرند.
- **پنهان‌سازی پیچیدگی:** برای مخفی کردن پیچیدگی‌های داخلی یک کامپوننت و فراهم کردن یک رابط ساده‌تر برای استفاده‌کنندگان.
- **افزایش قابلیت استفاده مجدد:** امکان استفاده مجدد از کامپوننت‌های قدیمی در محیط‌های جدید با استفاده از رابط‌های جدید.

تکنیک‌های مورد استفاده در Component Wrapping شامل:

- **Adapter Pattern:** استفاده از یک واسط یا آداپتور برای تبدیل رابط یک کامپوننت به رابطی دیگر که مورد نیاز است.
- **Facade Pattern:** ایجاد یک واسط ساده برای یک سیستم پیچیده، برای ساده‌سازی دسترسی به آن.

سوال ۴

سناریوهای زیر را در نظر بگیرید. هر سناریو ممکن است اصول طراحی نرم‌افزار شامل اصل

OCP (Open/Closed Principle)، اصل LSP (Liskov Substitution Principle)، اصل

DIP (Dependency Inversion Principle)، اصل ISP (Interface Segregation Principle) 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؟

الف) 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.

ب) یک کلاس رابط کاربری مسئول مدیریت ورودی‌های ماوس، ورودی‌های صفحه کلید، رندر کردن گرافیک و مدیریت پیام‌های شبکه است.

ج) 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.

د) 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.

جواب سوال ۴

در این سناریوها، مشکلات زیر ممکن است وجود داشته باشند:

الف) **نقض اصل LSP و اصل OCP:** در مورد کلاس Fish که متد makeSound را استفاده نمی‌کند، این نشان‌دهنده نقض اصل (LSP) Liskov Substitution است زیرا Fish نمی‌تواند به درستی جایگزین Animal شود. همچنین، این ممکن است نقض اصل (OCP) Open/Closed باشد زیرا برای اضافه کردن Fish نیاز به تغییر در کلاس Animal است.

ب) **نقض اصل ISP:** کلاس رابط کاربری که مسئولیت‌های زیادی دارد، نقض اصل Interface Segregation (ISP) است. بهتر است که وظایف به کلاس‌های تخصصی‌تر تقسیم شوند تا هر کلاس فقط مسئولیت‌های مرتبط با خود را داشته باشد.

ج) **Cohesion پایین و نقض اصل DIP:** کلاس دسترسی به پایگاه داده با مسئولیت‌های گسترده و اجبار مصرف‌کنندگان برای پیاده‌سازی تمام ویژگی‌ها، نشان‌دهنده Cohesion پایین است. این همچنین ممکن است نقض اصل (DIP) Dependency Inversion باشد، زیرا کلاس به جای استفاده از انتزاعات، مستقیماً به جزئیات پیاده‌سازی وابسته است.

د) **Coupling بالا:** سیستم پردازش پرداخت که به شدت به یک درگاه پرداخت خاص وابسته است، نمونه‌ای از Coupling بالا است. این وابستگی موجب می‌شود که هر تغییری در درگاه پرداخت تأثیر مستقیمی بر سیستم داشته باشد.

سوال ۵

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.

جواب سوال ۵

برای مرتب‌سازی و انتخاب در میان تعداد زیادی از design pattern‌ها در طراحی Pattern-Based، روش زیر پیشنهاد می‌شود:

الف) **تعیین نیازها و محدودیت‌ها:** ابتدا، نیازهای دقیق پروژه و هرگونه محدودیت مربوط به آن (مانند زمان، منابع، و محدودیت‌های تکنولوژیک) را شناسایی کنید.

ب) **دسته‌بندی الگوها:** الگوها را بر اساس دسته‌بندی‌هایی مانند ساختاری، رفتاری، و سازمانی تقسیم‌بندی کنید.

ج) **ارزیابی مطابقت الگوها:** برای هر دسته، الگوهایی که بیشترین مطابقت را با نیازهای پروژه دارند را ارزیابی کنید.

د) **تحلیل ترکیب‌پذیری:** بررسی کنید که چگونه الگوهای انتخاب شده می‌توانند به طور مؤثر با یکدیگر ترکیب شوند تا از تداخل کمتر و همکاری بیشتر بین آن‌ها اطمینان حاصل شود.

ه) **مدل‌سازی و ارزیابی:** پیاده‌سازی مدل اولیه با استفاده از الگوهای انتخاب شده و ارزیابی عملکرد آن در محیط‌های آزمایشی.

و) افزایش تکراری: بر اساس بازخورد و تحلیل‌ها، انتخاب الگوها را بهبود بخشیده و به صورت تکراری فرآیند را بهبود دهید.

این روش با تمرکز بر نیازهای خاص پروژه و انتخاب دقیق و مدبرانه‌ی الگوها، به تیم‌های طراحی اجازه می‌دهد تا راه‌حل‌های مؤثر و کارآمدی بیابند.
