به نام خدا

آزمون میان‌ترم سوم + پاسخ

نیم‌سال دوم ۱۴۰۱-۱۴۰۲

۸ خرداد ۱۴۰۲

سوالات چهارگزینه‌ای (هر کدام ۱ نمره)

1. کدام یک از مفاهیم طراحی زیر، به سازماندهی مولفه[1]های نرم‌افزار در واحدهای منطقی و متمرکز که می‌توانند به طور مستقل ایجاد و نگهداری شوند، اشاره می‌کند؟
   a. Modularity
   b. Abstraction
   c. Encapsulation
   d. Inheritance

   پاسخ: الف

   منبع: اسلاید ۱۲ صفحه‌های ۷ و ۱۵ - دلیل: مفهوم modularity دقیقا به همین مفهوم اشاره دارد. با تجزیه یک سیستم به اجزای ماژولار، درک، اصلاح و استفاده مجدد از نرم‌افزار آسان‌تر می‌شود.

2. کدام یک از موارد زیر در مورد طراحی نادرست است؟
   a. محصولات حاصل از طراحی باید برای کسانی که کد می‌زنند، یک راهنمای قابل خواندن و قابل درک باشد.
   b. محصولات حاصل از طراحی باید تصویر کاملی از نرم‌افزار ارائه دهد که از منظر پیاده‌سازی به حوزه‌های داده، عملکردی و رفتاری می‌پردازد.
   c. در طراحی تنها کافی است که نیازمندی‌هایی را که صریحاً در مدل‌های تحلیل آمده‌اند، در نظر داشته باشیم.
   d. اجزایی که در مدل‌های طراحی وجود دارند، با توجه به اجزایی که قبلاً در مدل تحلیل به دست آورده بودیم، ساخته می‌شوند و با یکدیگر مرتبط هستند.

   پاسخ: ج

   منبع: اسلاید ۱۲ صفحه ۵ و ۶ - دلیل: در طراحی باید علاوه بر نیازمندی‌هایی که صریحا در مدل تحلیل آمده‌اند، به نیازمندی‌های ضمنی‌ای که مدنظر مشتری است توجه کنیم.

---

[1] Component

3. کدام یک از گزینه‌های زیر مزیت استفاده از مخفی‌سازی اطلاعات[2] را به شکل بهتری توصیف می‌کند؟
   a. وابستگی[3]ها در سطح کد را کاهش می‌دهد.
   b. سرعت اجرای کد را افزایش می‌دهد.
   c. قابلیت نگهداری و اصلاح‌پذیری را بهبود می‌بخشد.
   d. فرآیند دیباگ‌کردن را تسهیل می‌کند.

   پاسخ: ج

   منبع: اسلاید ۱۲ صفحه ۱۸ - دلیل: با پنهان‌کردن جزئیات پیاده‌سازی و تنها افشاکردن اطلاعات ضروری از طریق رابط‌های کنترل‌شده و سایر مواردی که در information hiding انجام می‌دهیم، در نهایت به نرم‌افزاری با کیفیت بالاتر دست پیدا می‌کنیم که در طول زمان نگهداری و اصلاح‌پذیری آن آسان‌تر است.

4. در میان مفاهیم طراحی زیر، کدام گزینه میزان مسئولیت یک ماژول یا مؤلفه و اینکه این مسئولیت چقدر واحد و خوب تعریف شده است را مشخص می‌کند؟
   a. Coupling
   b. Cohesion
   c. Abstraction
   d. Refactoring

   پاسخ: ب

   منبع: اسلاید ۱۲ صفحه ۲۱ - دلیل: از تعریف Cohesion دقیقا می‌توان به این مفهوم رسید. Cohesion بالا نشان می‌دهد که یک ماژول بر روی یک کار یا عملکرد خاص تمرکز می‌کند، با کمترین تعامل و وابستگی به ماژول‌های دیگر.

5. هدف از بازآرایی[4] یک نرم‌افزار چیست؟
   a. معرفی ویژگی‌ها و قابلیت‌های جدید به نرم‌افزار بدون تغییر در مخزن کد موجود
   b. بهینه‌سازی عملکرد نرم‌افزار با بازنویسی کل پایگاه کد از ابتدا
   c. جایگزینی زبان‌های برنامه‌نویسی منسوخ و چهارچوب[5]ها با نسخه‌ی مدرن‌تر آن‌ها
   d. بهبود ساختار داخلی، طراحی و خوانایی کد بدون تغییر رفتار خارجی آن

   پاسخ: د

   منبع: اسلاید ۱۲ صفحه ۲۴ - دلیل: با استناد به تعریف بازآرایی می‌توان به درست بودن گزینه ۴ پی برد.

---

[2] Information Hiding
[3] Coupling
[4] Refactoring
[5] Framework

6. چرا در ایجاد نرم‌افزار، طراحی معماری[6] اهمیت دارد؟
   a. تضمین می‌کند که نرم‌افزار از نظر بصری جذاب و از نظر زیبایی‌شناسی دلپذیر است.
   b. طرح و نقشه‌ای را برای سازماندهی و ساختاربندی اجزای نرم‌افزار و تعاملات آن‌ها ارائه می‌دهد.
   c. بر بهینه‌سازی عملکرد و سرعت اجرای نرم‌افزار تمرکز دارد.
   d. قیمت و برآورد هزینه‌ی پروژه ایجاد نرم‌افزار را تعیین می‌کند.

   پاسخ: ب

   منبع: اسلاید ۱۳ صفحه ۳ - دلیل: هدف اصلی طراحی معماری، ارائه یک طرح اولیه یا یک طرح سطح بالا برای سازماندهی و ساختاربندی اجزای نرم‌افزار و تعاملات آن‌ها است.

7. در فرآیند ایجاد نرم‌افزار، پیچیدگی معماری معمولاً چگونه ارزیابی می‌شود؟
   a. با اندازه‌گیری خطوط کد نوشته‌شده برای طراحی معماری
   b. با تجزیه و تحلیل تعداد الگوهای معماری اجرا شده در طراحی
   c. با در نظر گرفتن وابستگی بین اجزای داخل معماری
   d. با ارزیابی جذابیت زیبایی‌شناختی و پیچیدگی بصری نمودارهای معماری

   پاسخ: ج

   منبع: اسلاید ۱۳ صفحه ۲۰ - دلیل: با خواندن مطالب اسلاید به درستی این گزینه می‌توان پی برد.

8. کدام اصل طراحی بیان دارد که اجزای نرم‌افزار را باید بتوان بدون تغییر در پیاده‌سازی موجود آن‌ها گسترش داد؟
   a. Dependency Inversion Principle
   b. Interface Segregation Principle
   c. Single Responsibility Principle
   d. Open-Closed Principle

   پاسخ: د

   منبع: اسلاید ۱۴ صفحه ۵ - دلیل: تعریف OCP دقیقا همان صورت سوال می‌باشد. این اصل مفهوم Open to extension-Close to modification را بیان می‌کند.

9. از میان اصول طراحی زیر، کدام یک پیشنهاد می‌کند که ماژول های سطح بالا نباید مستقیماً به ماژول های سطح پایین وابسته باشند، بلکه هر دو باید به انتزاع‌ها وابستگی داشته باشند؟
   a. Dependency Inversion Principle
   b. Liskov Substitution Principle
   c. Interface Segregation Principle

[6] Architectural Design

d. Single Responsibility Principle

پاسخ: الف

منبع: اسلاید ۱۴ صفحه ۵ - دلیل: تعریف DIP دقیقا همان چیزی است که در صورت سوال آمده است.

10. کدام مورد در مورد مهندسی دامنه[7] صحیح است؟

a. فرآیند طراحی و پیاده سازی رابط کاربری یک سیستم نرم‌افزاری می‌باشد.
b. فرآیند انجام مصاحبه و جمع‌آوری دانش خاص در مورد یک دامنه می‌باشد.
c. فرآیند تعیین دامنه، دسته‌بندی موارد استخراج شده از دامنه، جمع‌آوری نمونه از دامنه و تحلیل هر کدام و ایجاد مدل تحلیل برای آن‌ها می‌باشد.
d. فرآیند مستندسازی تصمیمات و منطق نرم‌افزار می‌باشد.

پاسخ: ج

منبع: اسلاید ۱۴ صفحه ۲۳ - دلیل: با توجه به متن اسلاید می‌توان به درستی این گزینه پی برد.

11. کدام یک از موارد زیر، قوانین طلایی[8] طراحی رابط کاربری[9] را که Theo Mandel مطرح کرده است، نقض می‌کند؟

a. اطلاعات کلیدی‌ای که در صفحات دیگر برنامه به کاربر نمایش داده شده اما کاربر در صفحه جدید هم به آن‌ها نیاز دارد، دوباره به او در صفحه‌ی جدید نشان داده شود.
b. کاربر باید بتواند تنظیماتی را که قبلا در برنامه اعمال کرده، به راحتی تغییر دهد.
c. خوب است که در صفحات مختلف برنامه از پالت‌راهای رنگی متفاوتی استفاده شود تا کاربر بتواند تفاوت این صفحات را از هم تشخیص دهد.
d. از میانبر[10]هایی که در برنامه‌های دیگر مرسوم است، در برنامه استفاده شود تا کاربر به حفظ‌کردن میانبرهای جدید برای کار با برنامه، نیاز نداشته باشد.

پاسخ: گزینه ۳

طبق اصل Make the interface consistent، باید از یک پالت رنگی در کل برنامه استفاده شود و تغییرات رنگی در روندهای مختلف برنامه به صورت جزئی تغییر کند. (صفحات ۴ تا ۷ اسلاید سری ۱۵)

12. کدام یک از اصول زیر کمک می‌کند تا بتوانیم در طراحی رابط کاربری، ثبات[11] داشته باشیم؟

---

[7] Domain Engineering
[8] Golden Rules
[9] User Interface
[10] Shortcut
[11] Consistencty

a. با قرار دادن عناوین برای پنجره‌ها، استفاده از کدگذاری رنگی[12] و...، به کاربر اجازه دهیم تا معنای فعالیتی را که در حال انجام دادن آن است، بفهمد.
b. لایه‌هایی مختلف و پیچیده را برای تعامل کاربر با برنامه پیاده‌سازی کنیم تا کاربر بتواند تمام کارهایی را که با برنامه می‌کند، ببیند.
c. مدل‌های تعاملی فعلی در برنامه را در به‌روزرسانی‌های اساسی، برای انتقال حس نو بودن تغییر دهیم.
d. در خانواده‌ای از برنامه‌ها، رابط کاربری هر کدام از برنامه‌ها را طبق اصول و قوانینی متفاوت طراحی کنیم تا قشرهای مختلف با سلیقه‌های متفاوت بتوانند از برنامه‌هایی متفاوت استفاده کنند.

پاسخ: گزینه ۱

گزینه ۲ غلط چون پیچیده نباید باشد. گزینه ۳ غلط به دلیل اینکه کاربران باید در طول عمر برنامه به یک صورت یکسان با برنامه تعامل داشته باشند. گزینه ۴ غلط زیرا تمام برنامه‌های یک خانواده باید از یک اصول پیروی کنند. (صفحه ۷ اسلاید ۱۵)

13. کدام یک از موارد زیر جزو مشکلات و دغدغه‌های طراحی رابط کاربری به حساب نمی‌آیند؟

a. بومی‌سازی رابط‌های کاربری برای کاربران نقاط مختلف دنیا
b. کاهش زمان انجام یک عملیات برای کاربر
c. افزایش دسترس‌پذیری برنامه
d. افزایش سرعت واکشی و نمایش اطلاعات

پاسخ: گزینه ۴

سه مورد دیگر طبق صفحه ۱۵ اسلاید ۱۵، از مشکلات طراحی به حساب می‌آیند.

14. کدام یک از گزینه‌های زیر جزو مواردی که برای داشتن یک رابطه کاربری با زیبایی بصری باید رعایت شوند، نیستند؟

a. مطالب، ناوبری[13]ها و عملکرد[14]ها در صفحه از لحاظ مکانی گروه‌بندی شوند.
b. برای سایت‌ها و برنامه‌های به زبان فارسی، المان‌های صفحات باید از بالا راست صفحه شروع و تا پایین چپ آن چیده شوند.
c. باید اندازه و وضوح نمایشگرهای مختلف در طرح‌بندی‌ها و چیدمان‌های مختلف در نظر گرفته شود.
d. حتی‌الامکان باید از داشتن فضای خالی (سفید) اجتناب کرد.

پاسخ: گزینه ۴

طبق اصل don't be afraid of white space، استفاده از فضای خالی حتی می‌تواند موجب زیبایی شود. (صفحه ۲۵ اسلاید ۱۵)

---

[12] Color Coding
[13] Navigation
[14] Function

15. یک نوآموز طراحی رابط کاربری می‌خواهد تا طراحی رابط کاربری برنامه‌ای را آغاز کند، اما نمی‌داند از کجا باید این کار را شروع کند. درک کدام یک از موارد زیر می‌تواند به او در شروع و مراحل ابتدایی طراحی رابط کاربری کمک کند؟

a. استفاده از کدام فونت و چه سایز فونتی می‌تواند رابط کاربری را جذاب‌تر کند.
b. چه مطالبی قرار است تا در برنامه نمایش داده شود.
c. استفاده از کدام پالت رنگی در برنامه می‌تواند به هدف آن نزدیک‌تر باشد.
d. چیدمان مطلوب برای صفحه اصلی برنامه به چه نحوی باشد.

پاسخ: گزینه ۲

دیگر گزینه‌ها از جزییات تلقی می‌شوند و باید در مراحل بعدی طراحی رابط کاربری به آن‌ها توجه داشت. در مرحله اول که تحلیل رابط کاربری[15] است، باید سوالاتی مثل گزینه ۲ پرسیده شود. (صفحه ۱۰ اسلاید ۱۵ که به تحلیل رابط و سوال‌های سطح بالای این مرحله اشاره دارد)

16. کدام گزینه با اصول طراحی رابط کاربری برنامه‌های موبایل و وب در تناقض است؟

a. رابط کاربری برنامه موبایل باید روی کاری که کاربر در حال انجام است، تمرکز داشته باشد.
b. رابط کاربری باید طوری طراحی شود که حرکت بعدی کاربر را بداند و متناسب با آن کنش‌ها و مطالب مناسب را به کاربر نمایش دهد.
c. ناوبری کاربر در برنامه باید کنترل شده و طبق اصولی که طراح آن‌ها را وضع می‌کند باشد و رابط کاربری این ناوبری را تا جای ممکن آسان کند.
d. طبق قانون Fitts، زمان مورد نیاز برای دسترسی به هدف در رابط کاربری، رابطه مستقیم با اندازه آن و رابطه عکس با فاصله با آن دارد.

پاسخ: گزینه ۴

رابطه در Fitts Law برعکس است. (صفحات ۱۹ تا ۲۱ اسلاید ۱۵)

17. در شرکت شما، مهندسان برای طراحی نرم‌افزارها و به منظور استفاده مجدد از راهکارهای گذشته و کاهش زمان ایجاد نرم‌افزار، تعدادی الگوی طراحی پیشنهاد کرده‌اند. شما به عنوان رهبر فنی این شرکت، وظیفه بررسی این الگوها را دارید. کدام یک از معیارها را برای بررسی موثر بودن این الگوها در نظر نمی‌گیرید؟

a. مسئله‌ای که الگو سعی در حل آن دارد، مسئله‌ی پیش پا افتاده‌ای است؟
b. الگوی پیشنهادی، فعالیت‌های انسانی برای طراحی نرم‌افزارهای سطح شرکت را به حداقل می‌رساند؟
c. الگو می‌تواند استراتژی‌های خوبی در قبال مسائل شرکت، مطرح کند؟

---

[15] Interface Analysis

d. روابط و مکانیزم‌های درونی سیستم‌هایی که طراحی می‌شوند، در الگو در نظر گرفته شده‌اند؟

پاسخ: گزینه ۳

الگوها باید پاسخ‌های جامع به مسائل پرتکرار در حوزه‌ای مشخص باشند؛ نه صرفا استراتژی‌های حل مسئله! (صفحه ۵ اسلاید ۱۶)

18. کدام یک از گزینه‌های زیر، سطح انتزاع الگوها را به درستی از بالا به پایین (زیاد به کم) بیان می‌کند؟

a. الگوهای معماری ← الگوهای طراحی ← الگوهای مولفه[16]
b. الگوهای طراحی ← الگوهای مولفه ← الگوهای معماری
c. الگوهای معماری ← الگوهای مولفه ← الگوهای طراحی
d. الگوهای طراحی ← الگوهای معماری ← الگوهای مولفه

پاسخ: گزینه ۱

رجوع به ص ۲۶ اسلاید سری ۱۶

19. از شما خواسته شده است تا برای نانوایی محله، یک برنامه تحت وب ایجاد کنید تا اهالی محل راحت‌تر بتوانند از این نانوایی خرید کنند. برای کاهش زمان ایجاد تصمیم گرفته‌اید تا از الگوهای طراحی موجود استفاده کنید. کدام یک از گزینه‌های زیر در رابطه با الگوهایی که می‌توانید در طراحی این نرم‌افزار استفاده کنید، صحیح نمی‌باشد؟

a. الگوهای ناوبری می‌توانند در طراحی سلسله مراتب و ساختار پیوندهای استفاده شده در برنامه کمک کنند.
b. الگوهای عملکردی[17] می‌توانند در تعریف جریان‌های کاری و المان‌های الگوریتمی برنامه کمک کنند.
c. الگوهای ارائه[18] در برنامه‌های تحت وب، ساختار کلی اطلاعاتی را که به کاربر نمایش داده می‌شود، تعیین می‌کند و همچنین مشخص می‌کند که کاربر چطور با این اطلاعات تعامل می‌کند.
d. الگوهای تعامل[19] به طراحی رابط کاربری برنامه کمک می‌کنند.

پاسخ: گزینه ۳

این تعریف مربوط به الگوهای معماری اطلاعات[20] است. (صفحه ۲۴ اسلاید ۱۶)

---

[16] Component
[17] Functional
[18] Presentation
[19] Interaction
[20] Information Architecture

20. در برنامه‌ای که طراحی کردیم، نیاز داریم تا قابلیت دیدن نقشه به آن اضافه شود. به این منظور قصد داریم تا از سرویس Google Maps استفاده کنیم. برای اضافه کردن این سرویس به نرم‌افزار و استفاده از آن، باید از کدام یک از الگوهای طراحی استفاده کنیم؟

a. Abstract Factory Pattern
b. Factory Pattern
c. Adapter Pattern
d. Aggregate Pattern

پاسخ: گزینه ۳

برای استفاده از سرویس‌های بیرون از سیستم، از الگوی adapter استفاده می‌شود. (صفحه ۸ اسلاید ۱۶)

سوالات تشریحی

در مهندسی نرم‌افزار یکی از مفاهیم اصلی طراحی استقلال عملکرد[21] است.

1. دو مفهوم اصلی در رابطه با استقلال عملکرد، Cohesion و Coupling هستند. درباره‌ی این دو مفهوم به صورت مختصر توضیح دهید. (۳ نمره)
2. آیا زیاد یا کم‌شدن هر یک از دو مفهوم Cohesion و Coupling بر روی دیگری اثر مستقیم دارد؟ توضیح دهید. (۳ نمره)

جواب بخش ۱)

Cohesion یک مفهوم توسعه‌یافته از مخفی‌کردن اطلاعات است. در این مفهوم ما می‌گوییم که یک ماژول اگر می‌خواهد منسجم[22] باشد، باید تنها یک کار را انجام دهد و با کامپوننت‌های دیگر در جاهای دیگر برنامه کمترین تعامل را داشته باشد. اگر بخواهیم به طور ساده بیان کنیم، یک ماژول Cohesive باید تنها یک کار را انجام دهد.

Coupling نشان‌دهنده‌ی اتصال ماژول‌ها در یک ساختار نرم‌افزاری است. وابستگی[23] به پیچیدگی رابطه‌ی بین ماژول‌ها بستگی دارد. در طراحی نرم‌افزار باید برای کم‌ترین میزان ممکن وابستگی تلاش کنیم. وابستگی کمتر به ما کمک می‌کند که راحت‌تر کد را متوجه شویم، بهتر بتوانیم تغییری در کد ایجاد کنیم و همچنین به نگهداری[24] از برنامه کمک شایانی می‌کند.

جواب بخش ۲)

در نگاه اول شاید به نظر بیاید که زیاد و کم شدن یکی لزوماً باعث زیاد و کم شدن دیگری می‌شود، اما این نگاه درست نیست. شما ممکن است برای این که cohesion را بالا ببرید، فانکشنالیتی‌های مختلف را از ماژولی به چند ماژول دیگر ببرید تا cohesion هر کدام از این ماژول‌ها بالا باشد، اما یک کلاسی وجود داشته باشد که از تمامی این فانکشنالیتی‌ها استفاده می‌کند. این باعث می‌شود که coupling بالا برود و آن کلاس با تمامی ماژول‌های جدید نیز در ارتباط قرار گیرد و تغییر در هر کدام از این ماژول‌ها باعث شود که مجبور شویم کلاس مذکور را نیز تغییر دهیم.

همچنین ممکن است در سناریویی دیگر، با بالا بردن cohesion از طریق جداسازی توابع، طوری این نیازهای کلاس مذکور را به این ماژول‌ها بچینیم که کلاس تنها با یک ماژول ساده شده در ارتباط باشد و آن ماژول از بقیه‌ی ماژول ها به همین صورت سلسله‌مراتبی فانکشن‌های خواسته شده را درخواست کند. در این حالت coupling به حداقل خودش می‌رسد و cohesion نیز همزمان بالا رفته است.

---

[21] Functional Independence
[22] Cohesive
[23] Coupling
[24] Maintenance

2. در کتاب درس دو کلمه‌ی Design Principles و Design Patterns آورده شده است. فرق این دو مفهوم چیست؟ از هر کدام یک مثال بزنید. (۴ نمره)

درواقع Design Principles مجموعه‌ای از اصول است که به ما نمی‌گوید دقیقا چگونه کدمان را پیاده کنیم، می‌گوید کدتان باید یک سری ویژگی‌هایی را داشته باشد. اما Design Pattern ها به ما به طور خاص دیکته می‌کنند که چگونه کدمان را طراحی کنیم تا به هدفی خاص برسیم.

برای مثال Single responsibility یک design principle از اصول SOLID است که به ما می‌گوید در توسعه‌ی شی‌گرا باید کلاس تنها و تنها یک عمل داشته باشد. انواع دیگر اصول می‌توان Separation of concerns، Encapsulation، Dependency inversion، Explicit dependencies، Persistence ignorance، انواع دیگر اصول SOLID و شی‌گرا و ... را نام برد که مورد تایید است.
مثال برای design pattern نیز هر کدام از الگوهای ۲۳ گانه‌ی GoF مورد قبول است.

“I’m not a great programmer; I’m just a good programmer with great habits.”
- Kent Beck.