به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

درس مهندسی نرم‌افزار ۲، نیم‌سال دوم سال تحصیلی ۱۴۰۲-۱۴۰۱

تمرین پنجم (امتیازی)

توضیحات تکمیلی:

- مبحث این تمرین الگوهای طراحی نرم‌افزار است.
- مهلت تحویل تمرین **۷ تیر** در نظر گرفته شده و قابل تمدید نمی‌باشد.
- پاسخ به تمرین‌ها به صورت **گروهی** می‌باشد و همه اعضای گروه موظفند در آن مشارکت داشته باشند.
- در صورت برخورد با پاسخ‌های مشابه بیش از حد بین گروه‌های مختلف، نمره کسر خواهد شد.
- ارسال فایل مربوطه در سامانه کورسز توسط سرگروه (نام ذکر شده در گوگل شیت گروه‌بندی‌های درس) کافی بوده و نیازی به بارگذاری توسط تمام اعضای گروه نیست.
- نظم و خوانایی تمرین از اهمیت بالایی برخوردار می‌باشد.
- تمرین خود را در قالب یک فایل PDF با نام"**SE2_HW[number]_[Group Number]**" مانند "SE2_HW5_6.pdf" در مهلت یاد شده در سایت درس بارگذاری کنید.
- برای بخش اول توضیح متنی، برای بخش دوم کد نوشته شده و برای بخش سوم نمودارها و توضیحشان را بیاورید.
- پرسش‌های خود درباره‌ی این تمرین را می‌توانید با آی‌دی nialda@ در تلگرام مطرح کنید.

مقدمه

موضوع این تمرین، الگوهای طراحی[1] در نرم‌افزار است که در درس مهندسی نرم‌افزار به طور مقدماتی با آن آشنا شده‌ایم. با توجه به امتیازی بودن این تمرین، پاسخ به پرسش‌های آن، علاوه بر استفاده از آموخته‌های قبلی خود، نیازمند جستجو در اینترنت است.

در ابتدا با پاسخ به پرسش‌های تئوری، برخی از این الگوها را مرور کرده، و سپس در بخش دوم برای سناریوهای مختلف این الگوها را به طور عملی در قالب کد و نمودار پیاده می‌کنیم.

معیار اصلی نمره‌دهی به پاسخ‌های شما، در درجه‌ی اول مسیر استدلال شما در تشخیص مسئله و توضیح راه حل است. در درجه‌ی بعدی پاسخ نهایی مورد بررسی قرار می‌گیرد.

بخش تئوری

۱. تشخیص الگوی طراحی مناسب

برای هر یک از موارد زیر، ابتدا مشخص کنید چه دسته‌ای از میان سه دسته‌ی اصلی الگوها مناسب بوده، و سپس الگوی مناسب را تشخیص و توضیح دهید. کافی‌ست استدلالی منطقی از این که الگوی پیشنهادی شما چطور مسئله را حل می‌کند، ارائه کنید.

الف) یک کلاس مدل داریم که در خود مجموعه‌ای از قطعه‌ها را ذخیره می‌کند. هر قطعه می‌تواند آهنی یا چوبی باشد. همچنین می‌تواند رنگ‌آمیزی، سمباده‌کشی یا روکشی شود. قطعه‌ها همچنین گاهی رادیواکتیوی یا مغناطیسی می‌شوند. چه الگوی طراحی‌ای امکان اضافه شدن نوع‌های جدید قطعه را (بدون تغییر کدهای قبلی) فراهم می‌کند؟

ب) کلاس مدل یک بازی، در خود شئ‌ای[2] از ربات بازی را ذخیره می‌کند. این ربات محیط یک هزارتو شامل برخی موانع را پیمایش می‌کند. در هنگام اجرای بازی، ربات می‌تواند با قطعه‌های جدید به‌روز شود تا مشخصاتی مانند سرعت و قدرتش تغییر کند. چه الگویی می‌تواند برای شئ ربات امکان تغییر رفتارش را در زمان اجرا به طور منعطف فراهم کند؟

ج) یک کتابخانه داریم که امکان جستجوی بازگشتی یک پوشه[3] برای فایل‌ها را فراهم می‌کند. این کتابخانه قابلیتی را فراهم می‌کند که طی آن کد کلاینت (کدی که از کتابخانه استفاده می‌کند)، با پاس دادن شئ‌ای نتایج جستجو را فیلتر کند، به طوری که به ازای هر یک فایل‌های نتایج، کتابخانه از شئ فیلتر پاس داده شده استفاده می‌کند تا تصمیم بگیرد این فایل قابل قبول است یا باید حذف شود (مشابه کتابخانه‌ی FileFilter جاوا). چه الگویی امکان این ارتباط منعطف را میان کد کلاینت و این کتابخانه فراهم می‌کند؟

بخش عملی

---

[1] Design Patterns
[2] Object
[3] Directory

۱. پیاده‌سازی الگوهای طراحی

در کنار دستور کار تمرین، پروژه‌ی اولیه‌ای به زبان جاوا برای این بخش قرار گرفته است. شما مجاز به ریفکتور[4] آزادانه‌ی این کدها هستید، تنها پیاده‌سازی الگوهای خواسته شده اجباری است.

الف) پکیج telephone داده شده در پروژه‌ی اولیه، شامل بخش‌هایی از کد برنامه‌ی یک تلفن خیالی‌ست. این کد را تغییر دهید تا از الگوی طراحی Observer استفاده کند. برای این کار،

- یک interface برای observerها تعریف کنید.
- مدل PhoneModel را به گونه‌ای تغییر دهید که Observerها را از وارد شدن یک رقم جدید شماره تلفن، با خبر کند.
- بخش UI (کلاس Screen) را به گونه‌ای تغییر دهید که دو observer زیر را داشته باشد.
  - اولین observer برای چاپ کردن شماره‌ی جدید وارد شده بر روی نمایش‌گر.
  - دومین observer با تکمیل شدن شماره تلفن، عبارت «Now dialing X» را چاپ می‌کند که در آن X شماره تلفن وارد شده است.

محدودیت‌ها:

- تنها UI می‌تواند روی نمایشگر چاپ کند.
- مدل (PhoneModel) باید از UI (Screen و Keypad) جدا (decoupled) باشد؛ (مدل نباید از وجود UI خبر داشته باشد و به آن وابسته باشد.)

ب) پکیج websearch داده شده در پروژه‌ی اولیه، شامل بخش‌هایی از کد یک موتور جستجوی خیالی‌ست. این برنامه داده‌ها را از یک فایل خوانده و وانمود می‌کند که هر خط، یک پرسمان[5] واقعی‌ست که کاربران ارسال کرده‌اند. هدف ما این است که برنامه با خواندن هر پرسمان، بی‌درنگ[6] تشخیص دهد که آیا این پرسمان شرایطی را برقرار می‌کند یا خیر.

کد اولیه از الگوی Observer برای خبر دادن هر پرسمان به Snooper استفاده می‌کند. کلاس Snooper صرفاً پرسمان‌ها را چاپ می‌کند.

این کد را تغییر دهید تا از الگوی طراحی Strategy استفاده کند. برای این کار

- یک interface جدید بنویسید که واسط یک شئ Policy را توصیف می‌کند. هر شئ Policy یک فیلتر را تعریف می‌کند.

---

[4] Refactor
[5] Query
[6] On the fly

- هر شئ فیلتر پرسمان Policy یک متد دارد. این متد یک رشته ورودی گرفته و در صورتی که مدل بایستی observer را از این پرسمان باخبر کند، مقدار true و در غیر این صورت مقدار false برمی‌گرداند.
- مدل را به گونه‌ای تغییر دهید که زمانی که یک Observer جدید ثبت (register) می‌شود، متد ثبت‌نام (registration) یک شئ فیلتر پرسمان Policy نیز ورودی بگیرد.
- مدل را به گونه‌ای تغییر دهید تا به ازای هر پرسمان (هر رشته از فایل)، چک کند آیا هر یک از Observerها به این پرسمان علاقه‌مند هستند یا خیر، و در صورت علاقه‌مند بودن، آن‌ها را باخبر کند.
- کد کلاینت (Snooper) را تغییر داده و دو Observer بسازید:
  - اولین Observer زمانی که یک پرسمان حاوی رشته‌ی «friend» باشد، عبارت «Oh yes! <query>» را چاپ می‌کند.
  - دومین Observer زمانی که یک پرسمان بیش از ۶۰ کاراکتر باشد، عبارت «So long! <query>» را چاپ می‌کند.

محدودیت‌ها:

- مدل نباید هیچ اطلاع یا وابستگی‌ای به پیاده‌سازی داخلی شئ‌های فیلتر پرسمان Policy داشته باشد، و تنها به interfaceـی که Policy تعریف می‌کند وابسته است.

ج) پکیج Bakery در پروژه‌ی اولیه، شامل بخش‌هایی از کد یک شیرینی‌پزی خیالی‌ست. این شیرینی‌پزی که در حال حاضر دو نوع کیک، وانیلی و شکلاتی می‌پزد، می‌خواهد به سراغ کیک‌هایی پیچیده‌تر برود؛ برای مثال «کیک وانیلی چند طبقه با شکر اضافه و نوشته‌ی «تولدت مبارک!».» این کد را به کمک الگوی Decorator به گونه‌ای تغییر دهید که یک سفارش (Order) بتواند شامل این کیک‌های پیچیده‌تر نیز بشود.

- کلاس‌های Decorator مورد نیاز را بنویسید.
  - برای کیک چندطبقه، به هزینه (cost) مبلغ 5$ اضافه شده و به نام آن عبارت «Multi-layered» اضافه شود.
  - برای شکر اضافه، به هزینه 2$ اضافه شده و به نام آن عبارت «with extra sugar» اضافه شود.
  - برای کیک با نوشته‌ی X، به هزینه مبلغی اضافه نمی‌شود، اما به نام آن عبارت «with saying X» اضافه می‌شود.
- نوع کیک جدید توت‌فرنگی (Strawberry) را اضافه کنید، که هزینه‌ی آن دو برابر کیک ساده است.
- کد کلاینت را تغییر داده تا سفارش زیر ساخته شده و چاپ شود.

- Chocolate cake
- Vanilla cake with saying "PLAIN!"
- Vanilla cake with sprinkles with saying "FANCY"

- Multi-layered Strawberry cake with double sprinkles and two sayings "One of" and "EVERYTHING"

محدودیت‌ها:

- اضافه کردن یک نوع جدید کیک، نباید باعث تغییر کدهای قبلی شود.
- اضافه کردن یک راه جدید برای پیچیده‌تر کردن کیک (مانند «چندطبقه بودن» یا «شکر اضافه») نباید باعث تغییر کدهای قبلی شود.

۲. رسم نمودار الگوهای طراحی

برای سناریوی زیر، به سؤالاتی که در ادامه آمده‌اند پاسخ دهید.

به شما وظیفه‌ی توسعه‌ی یک سیستم کنترل پمپ معدن داده شده است. هدف این سیستم نظارت و مدیریت سطح آب و میزان گازهای معدن است.

کنترل سطح آب: سیستمی که باید توسعه داده شود، برای نظارت بر سطح آب در چاه معدن از دو حسگر استفاده می کند. یک حسگر سطح آب بالا که سطح مجاز بیشینه آبدهی در یک چاه را قبل از شروع پمپاژ اندازه گیری می کند و یک حسگر سطح آب پایین که سطح کمینه‌ی مجاز آبدهی و توقف پمپاژ را اندازه گیری می کند. این حسگرها برای به راه افتادن پمپ معدن استفاده می‌شوند. هنگامی که سطح سیلاب به سطح تعیین شده توسط حسگر سطح آب بالا می رسد، پمپ روشن می‌شود. وقتی آب به میزان کافی خارج شده و حسگر پایین سطح آب را به میزان کمینه‌ی قابل قبول اندازه‌گیری کند، پمپ خاموش می‌شود.

کنترل میزان گاز: علاوه بر سیلاب، خطر ورود گاز متان نیز معادن را تهدید می‌کند. در صورت ورود این گاز، کارگران نمی‌توانند تنفس کنند و حتی جرقه‌های وسایل الکتریکی می‌تواند منجر به وقوع انفجار در معدن شود. سیستم از حسگر متان برای اندازه‌گیری این گاز در فضا (به واحد ppm) استفاده می‌کند. در صورتی که میزان این گاز از حد مجاز بیش‌تر باشد، فارغ از شاخص‌های آب، پمپ خاموش شده و هشدار تخلیه به صدا در می‌آید.

نقش‌ها: این سیستم توسط دو نقش کلیدی استفاده می‌شود. اولین نقش، نقش اپراتور است. این نقش برای ورود باید نام کاربری و رمز عبور خود را وارد کند. پس از ورود موفقیت‌آمیز، اپراتور می‌تواند تنها در شرایطی که سطح آب بین محدوده‌ی مجاز حسگرهای پایین و بالا باشد، پمپ را روشن یا خاموش کند. دومین نقش، نقش ناظر است. یک ناظر باید همانند اپراتور با نام کاربری و رمز عبور خود وارد حسابش شود. پس از ورود موفقیت‌آمیز، او می‌تواند در هر شرایطی پمپ را روشن یا خاموش کند. به عبارت دیگر، حتی در شرایطی که سطح سیلاب از حد تعیین شده توسط حسگر پایین(بالا) کم‌تر(بیش‌تر) باشد، ناظر می‌تواند پمپ را روشن(خاموش) کند. در چنین شرایطی، دستور ناظر بر عملکرد پیش‌فرض و خودکار سیستم اولویت دارد. پس از چنین دستوری، ناظر باید دستور ری‌سِت[7] شدن سیستم را برای بازگشت به عملکرد پیش‌فرض وارد کند.

لاگ‌نویسی[8]: به دلایل قانونی، سیستم باید موارد زیر را در قالب لاگ ذخیره کند.

- روشن شدن پمپ، ناشی از اندازه‌گیری حسگر سطح آب بالا

[7] Reset
[8] Logging

- خاموش شدن پمپ، ناشی از اندازه‌گیری حسگر سطح پایین آب
- روشن یا خاموش شدن آب، ناشی از دستور یک اپراتور یا ناظر
- فعال شدن هشدار تخلیه، ناشی از حسگر متان
- اندازه‌گیری حسگر متان، هر ۳۰ دقیقه یک بار.

اندازه‌گیری‌های ۲۴ ساعت اخیر حسگر متان، توسط اپراتور و ناظر قابل دسترسی است. همچنین تمام اندازه‌گیری‌های ۳۰ روز اخیر توسط ناظر قابل دسترسی و خواندن است. ناظر همچنین دسترسی اضافه کردن یادداشت به لاگ‌های ۲۴ ساعت اخیر را دارد.

الف) برای این سناریو، معماری نرم‌افزار را در قالب نمودارهای کلاسی (Class Diagrams) نمایش دهید. برای هر کلاس، صفات (Attributes) و متدها را به همراه نوعشان مشخص کنید.

ب) الگوی طراحی Abstract Factory

فرض کنید شرکت شما تصمیم گرفته درایور[9]هایی را برای مجموعه‌ای از پلتفرم‌های گوناگون فراهم کند تا برنامه شما بتواند با هر نوع حسگری که توسط برنامه پشتیبانی می‌شود، ارتباط برقرار کند (حسگر آب بالا، حسگر آب پایین، حسگر متان). این درایورها موارد زیر هستند.

- درایور لینوکس USB
- درایور لینوکس Serial Port
- درایور لینوکس Wireless

یک نمودار کلاسی بکشید که در آن کلاس‌های Abstract Factory لازم برای پشتیبانی این گونه‌های مختلف حسگرها نشان داده شود. هر کلاس Factory را به صورت Singleton تعریف کنید. به نظر شما اهمیت Singleton بودن کلاس‌های Factory در چیست؟

ج) الگوی طراحی Composite

شرکت شما بنا دارد که یک شبیه‌ساز محیط توسعه دهد تا با شبیه‌سازی رخدادهای محیط واقعی، سیستم خود را سنجیده و ایرادیابی کند. به عنوان بخشی از این شبیه‌ساز، شما باید یک Object Structure بسازید که در آن عبارت‌های زبان PSL – Pump Simulator Language مدل می‌شوند. این زبان از موارد زیر تشکیل می‌شود.

- High Water Event: این عبارت به معنای آن است که سطح سنجیده شده‌ی آب، از بیشینه‌ی مجاز بیش‌تر است.

---

[9] Device driver

- Low Water Event: این عبارت به معنای آن است که سطح سنجیده شده‌ی آب، از کمینه‌ی مجاز کم‌تر است.
- Methane Alarm: این عبارت حاکی از آن است که میزان سنجیده شده‌ی متان، از میزان مجاز بیش‌تر است.
- Supervisor Switches Pump On: این عبارت به معنای روشن شدن پمپ توسط ناظر است.
- Supervisor Switches Pump Off: این عبارت به معنای خاموش شدن پمپ توسط ناظر است.
- Run for N minutes: این عبارت به معنای روشن بودن پمپ به مدت N دقیقه است. این عبارت تنها پس از یکی از عبارات High Water Event یا Supervisor Switches Pump On می‌آید.
- Reset: این عبارت به معنای ری‌سِت شدن تمام هشدارها و خاموش شدن پمپ است.

این عبارات می‌توانند به صورت یک توالی در کنار هم قرار بگیرند و یک Workflow بسازند. هر Workflow یک Composite شامل یک یا چند عبارت و یا workflowهای کوچک‌تر است.

به کمک الگوی Composite، معماری‌ای در قالب نمودار کلاسی پیشنهاد دهید که مؤلفه‌های این زبان خیالی و ارتباطشان را بازنمایی می‌کند.

د) الگوی طراحی Adapter

پمپ‌ها ممکن است بیش از حد داغ شده و نیاز به سیستم سرمایشی داشته باشند. این سیستم باید هم‌زمان با عملیات پمپ اجرا شود.

یک نمودار کلاسی بکشید که یک interface یا کلاس برای سیستم سرمایشی توصیف می‌کند. این interface دو متد فراهم می‌کند.

- متد Switch (on/off)، این متد بر اساس پارامتر ورودی، سیستم سرمایشی را روشن یا خاموش می‌کند. هنگام گرفتن دستور خاموشی، سیستم سرمایشی به منظور سرد کردن پمپ‌ها به مدت ۲۰ دقیقه‌ی دیگر اجرا شده و سپس خاموش می‌شود.
- متد Emergency Off که سیستم سرمایشی را بی‌درنگ خاموش می‌کند.

یک Adapter بنویسید که سیستم پمپ‌ها و سیستم سرمایشی را هم‌زمان اجرا می‌کند. الگوی پیشنهادی خود را در قالب نمودار کلاسی توصیف کنید.

ه) الگوی طراحی Observer

هر پمپ هر ۳۰ ثانیه یک پیام شامل ۱. «دمای فعلی پمپ» ۲. «میزان آب پمپاژ شده از زمان آخرین» پیام و ۳. «زمان فعلی» آماده می‌کند. این اطلاعات باید به واسط کاربری (UI) فرستاده شده، و همچنین باید در قالب یک ایمیل به یک آدرس ایمیل مشخص ارسال شود. در نهایت این اطلاعات باید در یک پایگاه داده نیز ذخیره شود.

تغییرات مناسب در کلاس پمپ خود ایجاد کرده، و الگوی Observer را برای مدل‌سازی سناریوی گفته شده در قالب یک نمودار کلاسی نمایش دهید.

و) الگوی طراحی Visitor

هر حسگر و پمپ در سیستم باید دو data attribute زیر را داشته باشند.

- Last Serviced: نشان‌دهنده‌ی آخرین روزی که آن قطعه (سنسور یا پمپ) سرویس‌دهی شده است.
- Service Due: نشان‌دهنده‌ی روزی که سرویس‌دهی بعدی قطعه باید انجام شود. این داده در واقع بر اساس جمع مقدار Last Serviced با یک شیفت زمانی N ماهه به دست می‌آید.

علاوه بر این موارد، هر پمپ این اطلاعات را نیز نگه می‌دارد.

- محل پمپ (مثلاً "Shaft 6")
- نام پمپ

به کمک الگوی Visitor قابلیت گزارش‌دهی را به برنامه اضافه کنید، به طوری که به درخواست ناظر، گزارشی از اطلاعات تمام حسگرها و پمپ‌ها ساخته شود. معماری پیشنهادی خود را در قالب نمودار کلاسی توصیف کنید.