اهداف آزمایش

در این آزمایش هدف بر آن است که دانشجویان با به کارگیری اصول SOLID در یک پروژه‌ی عملی ساده آشنا شوند. برای تحویل گزارش در گیت‌هاب می‌توانید از قالب زیر استفاده کنید :

https://github.com/ssc-public/Software-Engineering-Lab/blob/main/courseworks/experiments/SOLID.md

بخش اول : توضیحاتی پیرامون برنامه‌ی داده شده

مفروضات مسئله

در ابتدا، یک برنامه‌ی ساده به زبان جاوا در اختیار شما قرار داده می‌شود. این برنامه‌ی خیلی ساده، شبیه‌سازی یک سیستم ساده‌ی ثبت سفارش را برای مشتریان یک رستوران انجام می‌دهد. مفروضات این برنامه به شرح زیر است:

۱. رستوران فقط ۲ نوع غذا را به قیمت‌های ۲۰۰۰ و ۱۰۰۰ واحد می‌فروشد (که از قبل در سیستم فرض شده‌است).
۲. در حال حاضر ۲ نوع روش **ثبت سفارش** داریم : روش حضوری و روش آنلاین.
۳. در حال حاضر ۲ نوع روش برای **پرداخت** داریم : روش حضوری و روش آنلاین.

با توجه به این که این سیستم صرفا یک شبیه‌سازی را انجام می‌دهد، در بدنه‌ی هرکدام از رفتارهای مربوط به ثبت سفارش و پرداخت آن، یک عمل چاپ بر روی صفحه‌ی نمایش اجرا می‌شود.

یک سناریوی ساده از اجرای برنامه

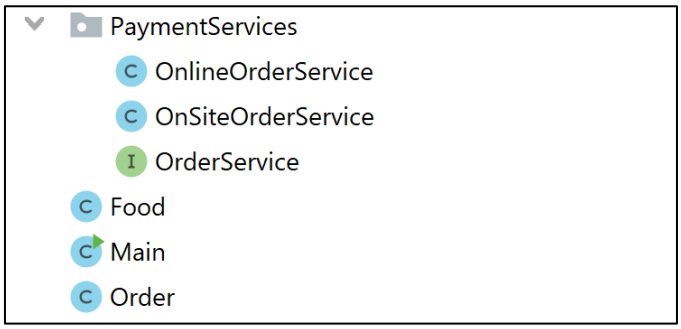
سناریوی یک سفارش خیلی ساده به شرح زیر است :

۱. نام مشتری به سیستم داده می‌شود.
۲. سفارش مشتری ثبت می‌شود (که ثبت چندین باره‌ی هرکدام از دو غذا بلامانع است).
۳. پس از پایان دریافت سفارش (با فشردن کلید ۳) عملیات دریافت سفارش پایان می‌یابد و در مرحله‌ی بعدی، **روش ثبت سفارش و روش پرداخت** (حضوری و یا آنلاین) تعیین می‌شود.

۴. در نهایت فاکتور مشتری در صفحه چاپ شده و اجرای برنامه پایان می‌یابد.

توضیحاتی اجمالی پیرامون ساختار برنامه

این برنامه، حاوی کلاس‌های زیر است :



۱. کلاس Main حاوی تابع main بوده و بخش عملیاتی آن را پوشش می‌دهد.

۲. کلاس Order حاوی اطلاعاتی در خصوص سفارش مشتری است و حاوی هیچ و یا تعدادی شی از نوع Food است.

۳. کلاس Food حاوی اطلاعات یک قلم غذا است.

۴. **واسط Order Service** حاوی توابعی برای ثبت سفارش (آنلاین و حضوری) و نیز توابعی برای پرداخت سفارش (آنلاین و حضوری) است (در مجموع ۴ تابع دارد).

۵. **برای هرکدام از روش‌های ثبت و پرداخت** سفارش یک کلاس در نظر گرفته شده‌است که واسط Order Service را پیاده‌سازی می‌کند. همه‌ی این کلاس‌ها همراه با واسط Order Service در بسته‌ی Payment Service قرار دارد.

بخش دوم : دستور آزمایش

آزمایش ۱ : افزودن یک روش پرداخت دیگر

۱. بدون آن‌که تابعی را از واسط Payment Service حذف کنید، یک کلاس دیگر تحت نام Phone Order Service ایجاد کنید و در آن، واسط Order Service را پیاده‌سازی کنید.

۲. سپس یک تابع برای **ثبت سفارش تلفنی (ورودی آن نام مشتری است)** و یک تابع برای **پرداخت سفارش تلفنی (ورودی آن مقدار کل مبلغ پرداختی سفارش است)** به واسط Order Service اضافه کنید و آن را در کلاس Phone Order Service پیاده‌سازی کنید.

۳. در بدنه‌ی هر تابع، از یک دستور چاپ ساده بر روی صفحه نمایش استفاده کنید.

۴. در ادامه، سعی کنید که قابلیت سفارش تلفنی را نیز به برنامه اضافه کنید؛ یعنی کاری کنید که کاربر بتواند در قالب **روش سوم** سفارش خود را ثبت کرده و پرداخت را انجام دهد.

۵. تغییراتی را که در کد فعلی برنامه می‌دهید، در جدول زیر ثبت کنید و در نهایت تعداد کل تغییرات را اعلان کنید.

- **توجه :** مواردی که به عنوان تغییرات باید اعلان شود شامل این موارد هستند :

۱. ساخت کلاس جدید

۲. افزودن تابع جدید به کلاس و یا واسط (برای توابع جدید صرفا اعلام تغییر کنید)

۳. هر خطوط پیاپی‌ای که در تابع main و برای افزودن یک **قابلیت جدید** اضافه می‌کنید. به عنوان مثال اگر سه خط را به منظور تشخیص نوع سفارش جدید اضافه می‌کنید، آن سه خط را در قالب یک تغییر اعلام کنید (البته جزئیات آن را در ستون شرحی کوتاه از تغییر، توضیح دهید).

ردیف	محل اعمال تغییرات (کلاس/واسط)	عنوان تغییر	شرحی کوتاه از تغییر
۱	Order Service	افزودن تابع پرداخت تلفنی	افزودن یک تابع void با عنوان phone Order Payment
۲	Order Service	افزودن تابع ثبت سفارش تلفتی	افزودن یک تابع void با عنوان phone Order Payment

مجموع تعداد تغییرات : .....

آزمایش ۲ : تحلیل و وارسی برنامه از منظر تحقق و یا عدم تحقق اصول SOLID

در خصوص این برنامه‌ای که نوشته شده بود و شما یک قابلیت به آن اضافه کردید، بر اساس اصول SOLID موارد نقض و یا محقق شدن هر کدام از آن اصول را بیان کنید. در بیان موارد تحقق و نقض، علت تحقق و یا نقض را نیز به صورت کامل توضیح دهید.

اصل ۱ Single Responsibility	**موارد تحقق**	
	موارد نقض	
اصل ۲ Open-Close Principle (OCP)	**موارد تحقق**	
	موارد نقض	

اصل ۳ Liskov Substitution Principle	**موارد تحقق**	
	موارد نقض	
اصل ۴ Interface Segregation Principle	**موارد تحقق**	
	موارد نقض	
اصل ۵ Dependency Inversion Principle	**موارد تحقق**	
	موارد نقض	

در خصوص هرکدام از موارد نقض هرکدام از اصول، یک راهکار را به منظور رفع آن مشکل ارائه داده و در جدول زیر ثبت نمایید.

اصل مربوطه (از اصول SOLID)	علت نقض	راه حل پیشنهادی

آزمایش ۳ : اصلاح موارد نقض

در نهایت، بر اســـاس تحلیلی که انجام داده‌اید و راه حل‌هایی که در بخش قبل ارائه کردید، کد را اصـــلاح کرده و بر روی ریپوزیتوری گیت‌هاب و در پوشه‌ای مجزا از آزمایش قبل commit و push کنید. انتظار می‌رود که تمامی راه حل‌های پیشنهادی خود را بر روی این نسخه اعمال کنید و تمامی بهبودهایی که انجام می‌دهید، در جداول بخش قبل موجود باشد.

آزمایش ۴ : بررسی مجدد تغییرات مورد نیاز

فرض کنید که آزمایش ۱ را برای کد اصلاح شده (پس از انجام آزمایشات ۲و ۳) اجرا کرده‌اید.

الف) در این صورت از انجام کدام یک از تغییرات ثبت شده در جدول آزمایش ۱ معاف خواهید شد؟

ب) تعداد تغییرات مورد نیاز، چند تغییر خواهد شد؟

آزمایش ۵ : جمع بندی

در این بخش، بیان کنید که از این آزمایش چه نتیجه‌ای گرفته‌اید؟ و به نظر شما به کارگیری صحیح اصول SOLID در آزمایشات ۳ و ۴ چه مزایایی را نسبت به حالتی دارد که این اصول رعایت نشده‌بود؟

منابع مناسب

منابع مفید برای انجام آزمایش، به صورت گام به گام در صفحه درس قرار گرفته و توضیحات مربوط به هر یک نیز ارائه گشته.

روش تحویل

۱. آزمایش ۱ را انجام داده و سپس کد نوشته شده‌ی خود را (با رعایت محدودیت‌های گفته شده در آزمایش) در یک پوشه به نام Exp_01_Non_SOLID ذخیره کنید و در ریپوزیتوری آزمایش ۲ قرار دهید. موارد توضیحی بایستی در بخش Read Me ریپوزیتوری قرار گیرد.

۲. آزمایش ۲ (که بخش تحلیلی است) در فایل Read Me مربوط به ریپوزیتوری آزمایش ۲ آورده شود (تمام جداول با فرمت داده شده و عناوین هرکدام از سوالات پرسیده شده بایستی در Read Me آورده شود).

۳. آزمایش ۳ که شامل بهبود است، بایستی به صورت جداگانه در پوشه دیگری به نام Exp_03_With_SOLID قرار داده شود و در ریپوزیتوری موجود باشد.

۴. آزمایش ۴ نیز در قالب توضیحات در Read Me ریپوزیتوری آورده شود.

۵. آزمایش ۵ (که نتیجه گیری است) در Read Me ریپورزیتوری آورده شود.

۶. **تمرین شما باید به زبان فارسی باشد – در غیر این صورت تصحیح نخواهد شد.**