abstract factory

دیزاین پترن ها در کنار تمرکز بر حل مشکلات برنامه نویسی شی گرا، اهداف متفاوتی را دنبال می‌کنند .یکی از اهداف اصلی آن ها، افزایش خوانایی کدها و کاهش میزان کدنویسی است .الگوی طراحی کارخانه انتزاعی یا Abstract Factory جزو الگوهای طراحی سازنده (Creational)است که برای مدیریت ساخت اشیا از کلاس‌ها توسعه داده شده است. این الگوی طراحی به شما اجازه می‌دهد که مجموعه ای از اشیا مرتبط را بدون نیاز به ساخت کلاس‌های جداگانه و متعدد ایجاد کنید.

کاربرد الگوی طراحی کارخانه انتزاعی تا حدودی مانند الگوی طراحی Factory است. این الگوی طراحی معمولا زمانی استفاده می‌شود که کاربر به صورت دقیق از نوع شی ای که می‌خواهد ایجاد کند اطلاع نداشته باشد. اگر قصد دارید از این الگوی طراحی در فرآیند کدنویسی خود استفاده کنید، از شما دعوت می‌کنم تا در ادامه این مطلب همراه ما باشید.

فهرست محتوای این مقاله

چرا باید از الگوی طراحی Factory استفاده کنیم

تصور کنید که یکی از تولیدکنندگان محصولات اداری به منظور طراحی نرم افزاری به منظور فروش و مدیریت خط تولید محصولاتش به شما مراجعه می‌کند. در این نرم افزار محصولاتی از جمله انواع مبلمان، میز و... طراحی می‌شوند و سپس به فروش می‌رسند. در حالت پیش فرض نرم افزار شما باید به ازای هر محصول یک کلاس (Class) داشته باشد تا رفتار و وظایف مربوط به هر کدام از اشیا را برای خط تولید تعریف کند. از طرفی دیگر این محصولات معمولا با یکدیگر به فروش می‌رسند و مکمل یکدیگر هستند.

فرض کنید این کارخانه فقط شامل سه محصول میز، کاناپه و مبل است. پس کلاس‌های مورد نیاز در نرم افزار عبارتند از: مبل + کاناپه + میز. همچنین در این کارخانه محصولات در سه سبک ساده، هنری و مدرن طراحی و ایجاد می‌شوند. از آنجایی که محصولات یک سبک معمولا با یکدیگر سفارش داده می‌شوند، باید آن‌ها به صورت خانواده ای از محصولات مرتبط در نظر گرفته شوند.

بنابراین باید قابلیت افزودن محصولات یا سبک‌های جدید را در نرم افزار در نظر گرفت. زیرا اگر این امکان پیش بینی نشده باشد در آینده باید کدهای موجود در نرم افزار تغییر یابند. زیرا فروشندگان محصولات معمولا خودشان لیست محصولات و کاتالوگ‌ها را به روز می‌کنند و منطقی نخواهد بود که هر بار که تغییری در محصولات ایجاد شود، کد اصلی نرم افزار نیاز به تغییر داشته باشد.

همچنین از طرفی دیگر باید در نرم افزار راهی برای فروش محصولات مرتبط با یکدیگر در نظر گرفته شود تا آن‌ها با سایر محصولات خریداری شده از یک نوع و سبک باشند. زیرا اگر محصولات ارسال شده به مشتری از لحاظ سبک متفاوت باشند، قطعا از آن‌ها ناراضی خواهد بود. فرض کنید که مشتری یک مبل، کاناپه و میز سفارش داده است و مبل از سبک ساده، کاناپه از سبک هنری و میز از سبک مدرن برای آن ارسال شوند. برای جلوگیری از چنین مشکلاتی در نرم افزارها، باید راه حلی در نظر گرفت.

راه حل الگوی طراحی کارخانه انتزاعی

راه حلی که الگوی کارخانه انتزاعی برای اینگونه مسائل پیشنهاد می‌دهد این است که یک Interface برای هر نوع محصول صرف نظر از سبک‌های آن ایجاد شود. مثلا در این مثال باید سه Interfaceبرای مبل، کاناپه و میز به صورت جداگانه تعریف شوند. سپس سبک‌های مختلف از این محصولات باید از اینInterface ها پیروی کنند. به عنوان مثال، تمام مبل‌ها با هر سبکی که دارند باید Interface مبل را پیاده سازی (implement)کنند؛ این روند به همین صورت باید برای محصولات دیگر انجام شود.

سپس باید در نرم افزار کارخانه ساخت محصولاتی را که از یک سبک هستند به صورت جداگانه تعریف شوند. برای این منظور نیاز به ایجاد یک Interface جدید خواهیم داشت که توابع و روش ساخت محصولات در آن تعریف می‌شوند. مثلا اگر نام کارخانه اصلی را کارخانه ساخت مبلمان در نظر گیریم، در این Interface توابع مربوط به ایجاد مبل، کاناپه و میز تعریف می‌شوند. همه این توابع باید یک محصول از نوع انتزاعی (Abstract) برگردانند. همچنین محصولاتی در این کارخانه‌ها تولید می‌شوند باید همگی محصولاتی باشند که به صورت Interface تعریف شده اند: مثلا مبل، کاناپه و میز.

برای هر سبک از محصولات، باید یک کلاس کارخانه ساخته شود که از Interface کارخانه ساخت مبلمان پیروی می‌کند. بر این اساس هر کارخانه یک کلاس است که از یک interface برای دریافت توابع پیروی می‌کند و محصولاتی از یک نوع خاص را بر می‌گرداند. به عنوان مثال، کارخانه ساخت مبلمان مدرن تنها می‌تواند مبل مدرن، کاناپه مدرن و میز مدرن ایجاد کند.

نرم افزار باید به گونه ای طراحی شده باشد که حتما کارخانه‌ها و محصولات ازInterface ها پیروی کنند. با این کار شما می‌توانید بدون اینکه نیازی به تغییر در کدهای اصلی نرم افزار داشته باشید، به راحتی در کارخانه و محصولات تغییر ایجاد کنید. از طرفی دیگر با این کار هر محصولی که توسط مشتری سفارش داده می‌شود، همیشه از لحاظ سبک با یکدیگر سازگار خواهند بود و دیگر از بابت نامتناسب بودن آن‌ها نگرانی نخواهد بود. زیرا کارخانه‌ها از لحاظ سبک تولیدی از یکدیگر متمایز شده اند.

Prototype Design Pattern

دیزاین پترن ها در سال 1994 توسط گروهی به نام Gang of Fourدر سه دسته عمومی طبقه بندی شدند. به آن دسته از الگوهای طراحی که با هدف مدیریت ایجاد اشیا توسعه یافته اند، الگوهای طراحی سازنده یا Creational می‌گویند .الگوی طراحی پروتوتایپ یا Prototypeیکی از الگوی‌های طراحی سازنده است که به منظور جلوگیری از ساخت یک شی جدید استفاده می‌شود.

این الگوی طراحی راهکاری برای ایجاد یک شی جدید به واسطه کپی کردن آن از اشیا موجود دیگر ارائه می‌دهد. با استفاده از این تکنیک بدون اینکه وابستگی میان شی جدید و کلاسی که از آن ساخته شده است ایجاد شود، می‌توان اشیا جدیدی در نرم افزار تولید کرد. همچنین در مواردی که مواقع نیاز به ایجاد تغییرات در اشیا جدید وجود دارد تا بتوانیم اشیایی از یک نوع و با پارامترهایی متفاوت ایجاد کنیم.

مثلا تصور کنید بخواهید از یک شی ماشین چندین کپی ایجاد کنید با این تفاوت که هر کدام رنگی متفاوت داشته باشد. با استفاده از الگوی طراحی پروتوتایپ دیگر نیاز نیست که هر بار یک شی جدید از کلاس ایجاد کنید و سپس پارامترهای آن را مشخص کنید. زیرا شما می‌توانید از کلاس ماشین یک شی ایجاد کنید و سپس از آن به تعدادی که نیاز دارید کپی (Clone) تولید کنید.

فهرست محتوای این مقاله

تصور کنید که تصمیم گرفته اید یک هواپیما بر اساس نمونه خارجی آن بسازید. برای بومی ساز این هواپیما تنها راهی که پیش روی شما خواهد بود این است که دقیقا از پارامترها و ویژگی‌های ساختاری نمونه خارجی آن آگاه باشید. اما از آنجایی که تکنولوژی هایی مربوط به ساخت هواپیماها انحصاری هستند و به راحتی در اختیار دیگر شرکت‌ها قرار نمی‌گیرند، این امر برای شما امکان پذیر نخواهد بود. اما تصور کنید که یک ماشین کپی سه بعدی غول پیکر برای شبیه سازی اشیا داشته باشید. با این کار می‌توانید هواپیما خارجی به آن دستگاه وارد کرده و هواپیما شبیه سازی خود را دریافت کنید. این ماشین شبیه ساز کاری شبیه الگوی طراحی پروتوتایپ انجام می‌دهد.

بنابراین اگر بخواهید از شی ای موجود در نرم افزار دقیقا یک کپی ایجاد کنید. ابتدا باید یک شی جدید از کلاس مورد نظر ایجاد کنید. سپس باید تمام پارامترها و ویژگی‌های شی اصلی را دریافت کرده و بر روی جسم جدید اعمال کنید. اما این روش همیشه به درستی عمل نخواهد کرد زیرا بعضی از پارامترها و ویژگی‌های یک شی ممکن است به صورت خصوصی یا Privateتعریف شده باشند. در این صورت دیگر امکان دسترسی به آن‌ها از خارج از کلاس امکان پذیر نخواهد بود.

بنابراین یکی از مشکلات کپی کردن اشیا در برنامه نویسی به صورت مستقیم، عدم دسترسی به برخی از پارامترهای خصوصی است .پس کپی کردن یک شی خارج از کلاسش همیشه امکان پذیر نخواهد بود. از طرفی دیگر به دلیل اینکه شما اشیای کپی شده را بر پایه کلاس‌های شی مبدا ایجاد می‌کنید، اشیای جدید به این کلاس‌ها وابسته می‌شوند و این کار باعث ایجاد وابستگی‌های زیاد در بین کلاس‌ها خواهد شد.

در الگوی طراحی نمونه اولیه یک Interface مشترک برای تمام اشیایی که باید کپی شوند در نظر گرفته می‌شود. این Interfaceبه شما اجازه می‌دهد که یک شی را بدون نیاز به دوباره نویسی کدها در کلاس شی، کپی کنید. معمولا، چنین Interfaceشامل تنها یک تابع به اسم Clone(کپی) هستند. پس با پیروی کلاس‌های مربوط به اشیا از این Interface دیگر نیازی به نوشتن توابع اضافه و یا استفاده از ایجاد مجدد یک شی نخواهیم داشت. برای این کار فقط کافی است تابع Clone را در کلاس‌ها پیاده سازی کنیم.

پیاده سازی تابع Clone در همه کلاس‌ها بسیار شبیه به هم است. این تابع وظیفه دارد تا یک شی از کلاس فعلی ایجاد کند و سپس تمام مقادیر پارامترهای آن را به شی جدید منتقل می‌کند. همچنین مزیت دیگری که این الگوی طراحی در اختیار شما می‌گذارد این است که حتی می‌توانید پارامترهای Private را کپی کنید. زیرا اکثر زبان‌های برنامه نویسی اجازه می‌دهند که اشیاء به فیلدهای خصوصی از اشیاء دیگر که متعلق به یک کلاس هستند دسترسی پیدا کنند.

به اشیایی که کلاس‌های آن‌ها از قابلیت کپی شدن پشتیبانی کنند یک نمونه اولیه یا Prototypeنامیده می‌شوند. همچنین هنگامی که اشیا دارای ده‌ها پارامتر و صدها تنظیمات باشند، می‌توانید از اشیا کپی شده به عنوان جایگزینی برای کلاس‌های فرزند استفاده شوند. با این کار در نرم افزار دیگر نیازی به ساخت کلاس‌های فرزند از کلاس اصلی نخواهید داشت. زیرا می‌توانید اشیا ساخته شده بر اساس کلاس اصلی را متناسب با نیازهایی که دارید تغییر دهید.

تصور کنید که یک کلاس به اسم سلول (Cell) در نرم افزار تعریف شود. برای اینکه این سلول به یک موجود تکامل یافته تبدیل شود، نیاز به تکثیر و نمونه سازی دارد. بنابراین به هزاران یا میلیون‌ها شی از این کلاس نیاز خواهیم داشت تا اندام‌های مختلف یک موجود زنده را تشکیل دهند و هر کدام به وظیفه خاص خود بپردازد. حال تصور کنید، اگر می‌خواستید همه این اشیا را به صورت مستقیم و یک به یک بسازید و پارامترهای آن‌ها را مقدار دهی کنید، چه میزان از زمان شما مصرف می‌شد.

کاربردهای الگوی طراحیPrototype

به کارگیری الگوی طراحی پروتوتایپ در طراحی نرم افزارها، باعث کاهش میزان کدنویسی خواهد شد. مزیت اصلی این دیزاین پترن افزایش سرعت کپی کردن یک شی از شی‌های است. زیرا این عمل خیلی سریع‌تر از ساخت اشیا می‌باشد. در عملیات کپی کردن اشیا تابع سازنده آن‌ها (Constructor)دیگر اجرا نخواهد شد. الگوی طراحی سازنده در موارد زیر کاربرد خواهد داشت:

- برای کاهش تعداد کلاس‌های فرزند (Subclass) استفاده می‌شود. همچنین با این روش می‌توان اشیا ساخته شده را متناسب با نیازهای مختلف پیکربندی کرد.
- به منظور جلوگیری از ایجاد وابستگی میان شی کپی شده به کلاس مورد نظر استفاده می‌شود.
- زمانی که بخواهیم از یک کلاس با پارامترهای Private زیاد شی ایجاد کنیم.