**چک لیست**

صفات موجود در کلاس های طراحی همانطور که ملاحظه می شود حاوی نام، نوع، دید و در صورت نیاز مقدار پیش فرض هستند و عملیات ها حاوی نام، نوع پارامترهای ورودی، نوع پارامتر خروجی و دید هستند.

از مهمترین ویژگی ها برای خوش فرم بودن کلاسهای طراحی وجود سه ویژگی در بین آنهاست:

1. کمال: کلاس ها نباید سرویس هایی کمتر از آنچه سرویس گیرنده انتظار دارد ارایه دهند.
2. کفایت: علاوه بر وجود "کمال" در کلاس ها، سرویس های ارایه شده نباید بیش از نیاز سرویس گیرنده باشند.
3. سادگی: سرویس ها باید ساده، اتمیک و یکتا باشند.

هر سه ویژگی فوق به دقت در نحوه کلاس بندی ها رعایت شده است به طوریکه هر کلاس تنها سرویس هایی را ارایه می دهد که از طرف کلاس دیگر و یا از بیرون تقاضا داشته باشند (نه بیشتر و نه کمتر) همچنین این سرویس ها به ساده ترین شکل موجود بوده و در صورت وجود پیچیدگی سرویس مورد نظر تا حد امکان به سرویس‌های ساده تر شکسته شده است.

به عنوان مثال کلاس project تمام اعمالی را که بر روی یک پروژه موجود از طرف کاربران گوناگون قابل انجام است پشتیبانی می کند، این اعمال به صورت ساده شده و در پایین ترین سطح ممکن به صورت "عملیات" های جدا در نظر گرفته شده اند، همچنین کاربران می توانند تمام نیازهای ارتباطی مربوط به یک پروژه را از طریق این کلاس برطرف کنند و همچنین "عملیات" یا "صفات"ی بیش از محدوده تعریف project در این کلاس موجود نمی باشد.

دو ویژگی مهم دیگر در راستای خوش فرم بودن کلاس های طراحی عبارتند از:

1. انسجام زیاد: هر کلاس باید به تنهایی یک مفهوم انتزاعی خوش تعریفی را داشته باشد و تمام عملیاتهای موجود هدف اصلی کلاس را دنبال کنند.
2. ارتباط کم: باید ارتباط یک کلاس با سایر کلاس ها به حداقل ممکن برسد و تنها کلاسهایی که از نظر منطقی ارتباط دارند در نهایت دارای رابطه باشند. (نباید صرف استفاده مجدد از یک کد، ارتباطات را افزایش داد.)

برای مثال می توان به وجود کلاس‌های catalogue در بین کلاس ها اشاره کرد، به عنوان مثال می توان با ادغام کلاس project و projectCatalogue یک کلاس حاوی اطلاعات پروژه و فهرستی از پروژه‌ها داشته باشیم که در این صورت از نظر معنایی مجموعه اطلاعات تمام پروژه‌ها را با اطلاعات جزیی مربوط به یک پروژه تلفیق کرده‌ایم که باعث کم شدن "انسجام" می شود و وجود کلاس‌های catalogue، این مشکل را برطرف می‌کنند.

همچنین به عنوان مثال می توان به ارتباط User به عنوان یک کارمند با کلاس projectCatalogue اشاره کرد. در منطق سامانه هر کارمند بر روی یک یا چند پروژه مشغول به کار است و منطقی است که ارتباطی یک به چند بین کارمند و پروژه ایجاد شود، اما با وجود کلاس projectCatalogue، این ارتباط که باعث افزایش "coupling" می‌شود را می‌توان کاهش داد، به این صورت که کارمند با کلاس کاتالوگ پروژه ها در ارتباط است و این کلاس نیز با کلاس پروژه، به این ترتیب کارمند با دریافت فهرست پروژه ها از کاتالوگ جزییات آنها را نیز از این کلاس درخواست می کند و کلاس projectCatalogue این اطلاعات را از طریق ارتباطی که با کلاس project دارد به کارمند می دهد، به این ترتیب با حذف دسترسی مستقیم کارمند به کلاس پروژه "ارتباط" را کم نگه داشته‌ایم و اطلاعات مورد نظر کارمند را نیز در اختیارش قرار داده‌ایم.