3.7 الگوهای طراحی (Design Pattern)

3.7.1 Bridge

برای ذخیره‌سازی سوالات و جواب‌های آن‌ها از Bridge استفاده شده است. در این Design Pattern سعی می‌شود که پیاده‌سازی‌ها از Abstraction ها مجزا باشند تا بتوانند قابل جایگزینی و تعویض باشند. به همین منظور برای ذخیره‌سازی از این Pattern استفاده شده است تا بتوان عمل ذخیره‌سازی را به صورت‌های مختلف انجام داد. نتیجه این Design Pattern این است که می‌توان تمام داده‌ها اعم از تنظیمات مسابقه، امتیاز تیم‌ها، سوالات و جواب‌ها را در دیتابیس‌های مختلف و حتی در فایل ذخیره کرد.

3.7.2 Observer

این Design Pattern ما را قادر می‌سازد تا قسمت‌هایی از برنامه دائما در حال گوش‌دادن به تغییرات وضعیت یا State بخش دیگری از برنامه باشند و متناسب با این تغییرات واکنش متناسب نشان دهند. برای طراحی بسیاری از بخش‌های UI و ظاهر برنامه از این Design Pattern استفاده شده است. به طول مثال بخشی از برنامه یک بورد نشان می‌دهد که امتیاز تمام تیم‌های مسابقه را به صورت نمودار و همین طور به صورت عدد نشان می‌دهد. به محض این که امتیاز یک تیم تغییر کند، بورد خود به خود به روز شده و آخرین تغییرات را نشان می‌دهد. برای پیاده‌سازی چنین ویژگی از Observer استفاده شده است و بورد در حال Observe کردن امتیازات تیم‌هاست و اصطلاحا به این تغییرات Subscribe شده است و به محض تغییر امتیازات نمودارها دوباره به روز شده و رسم می‌شوند.

Builder 3.7.3

این Design Pattern ما را قادر می‌سازد تا جدا و خارج از یک نمایش خاص از Object بتوانیم در موارد نیاز آن را بسازیم و از آن استفاده کنیم. برای ساخت سوال در بازی از این Design Pattern استفاده شده است تا کاربر بتواند به راحتی خارج از صفحه نمایش دهنده سوال، یک سوال بسازد، تغییرات لازم را بدهد و آن را ذخیره کند و بعدا در صفحه نمایش دهنده سوال آن را ببیند و از آن استفاده کند.

Command 3.7.4

این Design Pattern ما را قادر می‌سازد تا بتوانیم اجرای یک دستور را در قالب یک Object توصیف کرده و به متد مورد نظر ارسال کنیم. در این برنامه برای ارسال اجرای بسیاری از دستورات از این Design Pattern استفاده شده است. به طور مثال برای ارسال دستور گرفتن نسخه پشتیبان از کل مسابقه یک Object ساخته می‌شود. این Object حاوی تمام جزییات لازم برای اجرای این دستور است، مانند محل ذخیره فایل پشتیبان و نام آن. سپس متدی که مسئول تهیه نسخه پشتیبان است با استفاده از این Object و این Design Pattern این دستور را اجرا می‌کند. برای بقیه عملیات‌های ذخیره‌سازی، پاک کردن و ریست کردن کل مسابقه و دیگر عملیات نیز از این Design Pattern استفاده شده است.

Servant 3.7.5

به کلاس‌هایی که دارای متدهای کمکی یا Helper باشند و دارای شئ خاصی نباشند Servant گفته می‌شود. برای بسیاری از عملیات از این Design Pattern استفاده شده است. مثلا برای نوشتن فایل‌های پشتیبان بر روی دیسک، خواندن فایل‌ها و فشرده‌سازی فایل‌های مسابقه از این کلاس‌ها و متدها استفاده شده است.

Singleton 3.7.6

این Design Pattern ما را قادر می‌سازد تا از یک Object تنها یک instance داشته باشیم و بتوانیم در سرتاسر برنامه از همان یکی استفاده کنیم. در این برنامه از این Design Pattern برای دسترسی به آبجکت Game استفاده شده است. این آبجکت حاوی تمام اطلاعات و دیتای مسابقه، سوالات، جواب‌ها، تنظیمات و امتیازات است که باید تنها یک instance داشته و همه جا به همین یکی ارجاع داده شود.