DESIGN PATTERN NEDIR

Design patternlər proqram mühəndisliyinin bir hissəsidir və bu patternləri bir birinə oxşar problemləri həll etmək üçün ümumi olaraq qəbul olunmuş strukturlar kimi də fikirləşmək olar. Programlaşdırmada Design patternlər ən optimal üsul olaraq qəbul edilir. Bunun səbəbi odur ki , bu patternlər illərlə istifadə edilib, test edilib və ən doğru yolun məhz bu yollar olduğu qəbul edilib. Dizayn patternlər kodu rahat başa düşmək, sonradan ortaya çıxan dəyişikləri rahat şəkildə tətbiq edə bilmək, yazılan kodların təkrarlanmasından qaçmaq, yazılan kodların bir çox yerdə istifadə oluna bilməsini təmin etmək və s üçün developerlər tərəfindən tez-tez istifadə olunur. Sadalanan səbəblərə görə Design patternlərdən istifadə edərək yazdiğımız kodların keyfiyyətini artıra bilərik.

Həmçinin, Design patternlər, proqram mühəndisləri arasında əməkdaşlığı asanlaşdırmaqla programlaşdırmanın daha yaxşı dizayn edilməsinə və idarə edilməsinə kömək edir. Lakin, hər halda Design patternlərin istifadəsinin ehtiyatla qiymətləndirilməsi və faydalarının dəqiq hesablanması lazımdır.Beləki bəzi kiçik proyektlərdə design patternlərdən istifadə etmək hər zaman optimal variant olmuya bilir.

DESİGN PATTERNLERİ TARİXİ

Dizayn patternlər proqram mühəndisliyinin inkişafı ilə ortaya çıxmisdir. İlk dizayn patternlər 1970 və 1980-ci illərdə proqram mühəndisliyində istifadə edilib. Bununla belə, dizayn pattern konsepsiyası Gang of Four kimi taninan Erich Gamma, Richard Helm, Ralph Johnson və John Vlissides tərəfindən yazilmis " Design Patterns: Elements of Reusable Object-Oriented Software" kitabı ilə məhşurlaşmışdır. Kitab 1994-cü ildə nəşr olunub və həmin ildən proqram mühəndisliyi üçün əsas mənbələrdən birinə çevrilmişdir.

Bu kitabda GoF müəyyən bir code problemi üçün təkrar istifadə edilə bilən və sınaqdan cıxmış 23 dizayn Pattern müəyyən etmişdir. Bu Patternlər movcud proqramlaşdırma problemlərinə diqqət yetirir və ümumi həllər təmin etməklə proqram təminatının çevikliyini, təkrar istifadəsini, oxunaqlılığını və davamlılığını artırmaq məqsədi daşıyır.

Bu gün dizayn patternler proqram mühəndisləri arasında geniş istifadə olunur. Proqram təminatı sənayesində dizayn patternler proqram təminatının keyfiyyətini yaxşılaşdırmaq və daha yaxşı dizayna nail olmaq üçün mühüm alət hesab olunur.

Design patternlar zamanla populyarlaşdı və bu gün proqramlaşdırma dillərinin hər birində istifadə edilir. Bu dillər arasında Java, C++, Python, C#, JavaScript və daha bir çox bu kimi dilleri misal cəkmək olar. Dizayn patternlərin hər bir proqramlaşdırma dilindəki syntaxı fərqlənə bilər, lakin hər bir patternin mənası və tətbiq sahəsi bənzərdir. Design patternlərin fərqli dillərdəki istifadəsi, dilin syntaksı və xüsusiyyətlərinə uyğun olaraq dəyişir .Proqramın bezi hissələrindən asanlıqla yenidən istifadə edə bilmək üçün müxtəlif design patternlərin istifadəsi mümkündür. Misal üçün, Factory Pattern bir çox obyektin yaradılması ehtiyacı olan məlumat sistemləri üçün idealdir. Adapter pattern, həmçinin, bir proqramlaşdırma dili ilə digər proqramlaşdırma dilləri arasında köklü dəyişikliklərə görə problem yarada biləcək məsələlər üzərində işləyir. Ve s

DESİGN PATTERNLERİN NOVLERİ

Design patternlar müxtəlif tiplərə bölünə bilər və C# dili daxil olmaqla hər hansı bir proqramlaşdırma dilində bu patternlərdən istifade etmək mümkündür.

1. Yaradıcı patternlar: Bu patternlar, obyekt yaratma prosesində tətbiq edilir və istənilən obyekti yaratmaq üçün fərqli yollar tətbiq edilir. Bu patternlərin bir neçə nümunəsi arasında Factory, Abstract Factory, Singleton və Builder Dizayn Patternlərini misal çəkə bilərik.

2. Struktural patternlar: Bu patternlər, obyekt və ya klassların təşkili və birlikdə işləməsi ilə bağlı məsələlərə yönəldilir. Bu patternlərin bir neçə nümunəsi arasında Adapter, Bridge, Facade və Composite kimi patternlər daxildir.

3. Davranış patternləri: Bu patternlər, obyekt və ya klassların bir-biri ilə münasibətləri və davranışları ilə bağlı problemlərə yönəldilir. Bu patternların bir neçə nümunəsi arasına Observer, Template Method, Strategy və Chain of Responsibility daxildir.

C# proqramlaşdırma dilində, design patternların istifadəsi çox yayğındır. C# dili, obyekt-orientəd bir dil olduğu üçün, design patternların çox hissəsi asanlıqla tətbiq edilə bilir.

Məsələn, Singleton pattern, yalnız bir dəfə yaradılmış bir obyektin istifadəsi ilə əlaqədar problemlərə yönəldilir. Bu patternin tətbiqi çox asandır və yalnız bir klas üzərində işləyir. C# dili, Singleton patternini tətbiq etmək üçün istifadəçilərə dəstək verir və bu patternin tətbiqi məsələlərinin həllinə kömək edir.

Adapter pattern isə hər hansı bir obyektin digər bir obyektdə istifadə edilməsini təmin edir. C# dili, Adapter patternını tətbiq etmək üçün interfeyslərdən istifadə edir və bu patternin tətbiqi məsələlərinin həllinə kömək edir.

Son olaraq Template Method patterni bir classın funksiyalarının bir hissəsinin dəyişdirilə biləcəyi və ya dəyişdirilməyəcəyi vəziyyətlərdə istifadə edilir ve s.

ESL HEYATDAN MİSALLAR

Design patternlər həyatımızın bir çox sahəsində və məsələlərində istifadə olunur.

1. Singleton pattern: Həyatımızda yalnız bir dəfə yaradılmış obyektə ehtiyacımız ola bilər. Məsələn, bir şirkətdə yalnız bir məlumat bazası yaradılması tələb oluna bilər. Bu halda, Singleton patternı tətbiq edərək yalnız bir dəfə yaradılmış obyektin istifadəsi təmin edilir.

2. Adapter pattern: Həyatımızda bir obyektin fərqli bir formatda istifadə edilməsi ehtiyacı ola bilər. Məsələn, bir evin elektrik prizinin əlavə bir USB portuna çevrilməsi üçün Adapter patterni tətbiq edilə bilər.

3. Strategy pattern: Həyatımızda bir məsələnin birdən çox həll yolu ola bilər. Məsələn, bir işçinin iş yerinə gəlməsini təmin etmək üçün avtomobil, velosiped və ya taksilərin istifadəsi mümkündür. Bu halda, Strategy patternı tətbiq edərək, fərqli həll yolundan istifadə edən obyektlərin yaradılması və uyğun həll yolu seçilməsi təmin edilir.

4. Observer pattern: Həyatımızda, bir məlumatın digər məlumatlara uyğun şəkildə ötürülməsi ehtiyacı ola bilər. Məsələn, bir mağaza maşınının stokda olmadığı halda, müştəriyə yaxınlaşanda, maşının stokda olmadığının bildirilməsi Observer patternı tətbiq edərək həyata keçirilə bilər.

Bu nümunələr, design patternların həyatımızın bir çox sahəsində istifadə olunduğunu göstərir və design patternların proqramlaşdırma problemlərindən çox daha geniş sahələrdə istifadə edilə biləcəyinin göstəricisidir.