**THE SOLID PRINCIPLES**

SOLID prinsipləri ilk dəfə 2000-ci ildə Komputer alimi Robert Cecil Martin(Uncle Bob olaraq da bilinən) tərəfindən öz məqaləsində təqdim edilmişdir. Lakin SOLID qısaltması isə daha sonra Michael Feathers tərəfindən təqdim edilmişdir. Həmçinin Robert C.Martin ən çox satılan Clean Code və Clean Architecture kitablarının müəllifi, Agile Alliance iştirakçılarından biridir.

SOLID dizayn prinsipləri, developerlərin gündəlik proqramlaşdırma zamanı ümumiyyətlə qarşılaşdığı Software dizayn problemlərin əksəriyyətini idarə etmək üçün istifadə olunan prinsiplərdir. Bu dizayn prinsipləri proqram dizaynlarını daha başa düşülən və sürətli hala gətirmək üçün müəyyən sınaqlardan keçmiş və sübut edilmiş mexanizmlərdir. Əgər prinsiplərə riayət edərək aplikasiyalar dizayn etsək daha yaxşı aplikasiyalar hazıraya bilərik.

**S - The Single Responsibility**

“Every software module or class should have only one reason to change.”

Single Responsibility prinsipinə əsasən class’lar və ona oxşar strukturların yalnız bir məsuliyyəti olmalıdır yəni bir modul yalnız bir məqsədi icra etməli və bu səbəbdən də bu modul’da dəyişiklik edilməsi üçün yalnız bir səbəb olmalıdır. Bu prinsipi tətbiq etmək vacibdir.

Əvvəla bir sıra komandaların üzvləri eyni proyekti inkişaf etdirə biləcəyindən və eyni class’ı fərqli səbəblərdən dolayı dəyişə biləcəyindən bu situasiya uyğunsuz modullara yol aça bilər. Əgər bir class’ın bir çox məsuliyyəti olarsa (if class has more than one responsibility), bu class üzərində dəyişiklik etmək üçün birdən çox səbəb olacaq. Class üzərində edilən hər hansısa dəyişiklik class’ın digər davranışlara da təsir edə bilər. SRP(Single Responsibility Principle) tətbiq olunduğu halda edilən dəyişikliyin əlaqəsi olmayan digər davranışlara heç bir təsiri olmayacaq.

İkinci olaraq version control asanlaşır. SRP tətbiq edildikdən sonra hər hansısa bir class üzərində edilən dəyişiklik commit olunubsa bir başqa komanda üzvü bu commit’i araşdırdıqda rahatlıqla dəyişikliyin nə ilə bağlı olduğunu anlayacaq. Həmçinin yaranan konfliktlər azalacaq və həlli asanlaşacaq

**O - The Open-Closed Principle**

“Software entities(classes, modules, functions .etc) are Open for extension, Close for modification.”

Module’lar(class’lar, function’lar) genişlənməyə açıq, modifikasiyaya qapalı olmalıdır. Modifikasiya deyiləndə mövcud class’ın kodlarında dəyişiklik nəzərdə tutulur. Genişlənmə deyiləndə isə yeni funksionallıqların artırılması, əlavə olunması nəzərdə tutulur. Mövcud class’ın davranışını dəyişmək ondan istifadə edən bütün sistemlərə təsir edəcək. Bu da potensiyal bug’ların yaranması riskinin gözə alınması deməkdir. Yeni bir funksionallıq əlavə etmək istəyiriksə mövcud olan class kodlarına toxunmamalı və ya üzərində dəyişiklik etməməliyik.

**L - Liskov Substitution Principle**

**I - Interface Segregation Principle**

“Clients should not be forced to implement any methods they don’t use.”

Hər bir interface’in məqsədi olmalı, daxili isə şişirilməməlidir. Client’lar istifadə etməyəcəkləri interface davranışlarını implement etməyə məcbur edilməməlidirlər. Bu problemin həlli üçün bir şişirdilmiş interface əvəzinə hər biri fərqli modullara xidmət edən kiçik interface’lər yaranmalı və lazım olduqları təqdirdə implement edilməlidir.