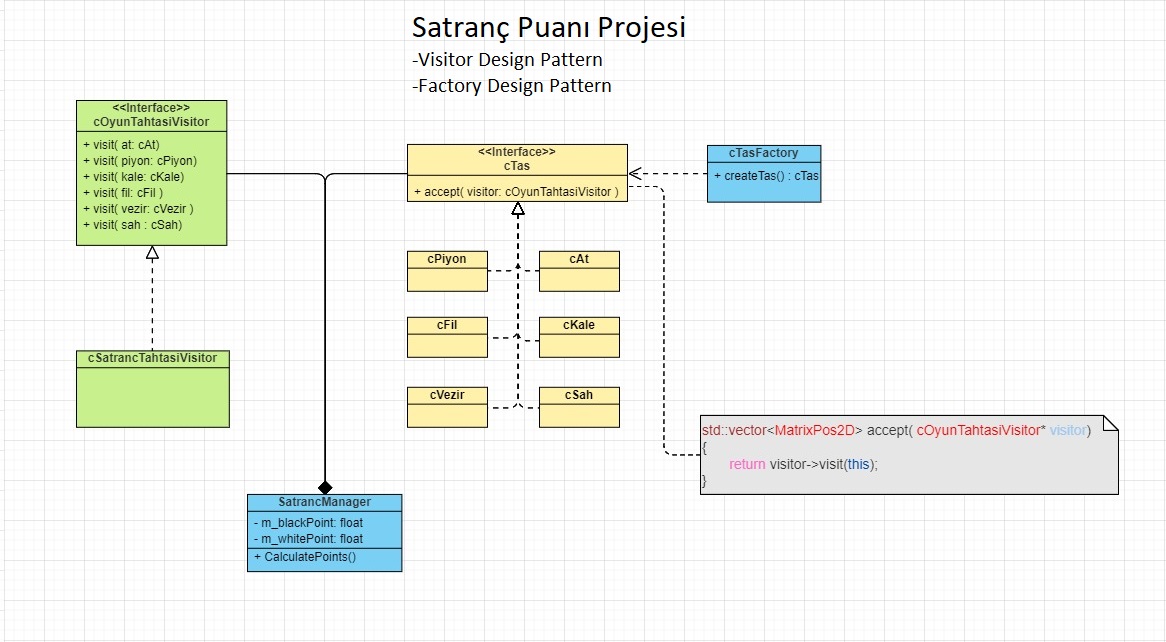
1. **Giriş**

Satranç tahtası üzerindeki taşların tehdit durumlarına göre beyaz ve siyah tarafın puanlarını hesaplayan proje, Visitor Design Pattern ve Factory Design Pattern kullanılarak gerçekleştirildi.

1. **UML Diagramı**



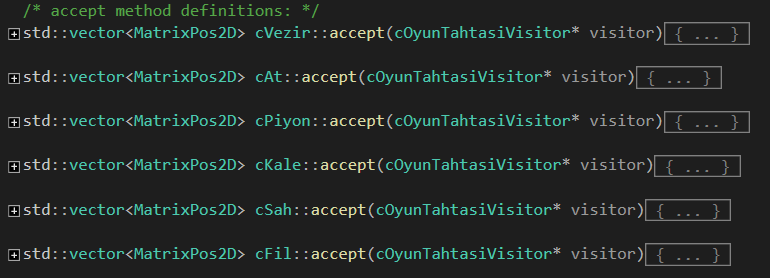
1. **Sınıflar (OOP Class)**
   1. **Taş Sınıfları**

Polymorphism özelliğinden yararlanabilmek için tüm taşların türediği bir abstract class (**cTas**) yazıldı. **cTas** abstract sınıfından:

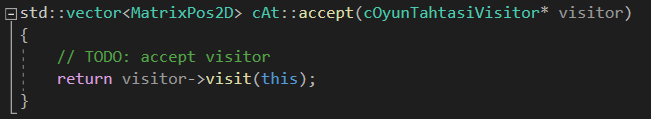
* **cPiyon**
* **cAt**
* **cKale**
* **cFil**
* **cVezir**
* **cSah**

concrete sınıfları türetildi.



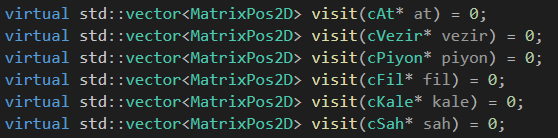


Visitor Design Pattern kapsamında her taş, kendi **accept** metodu içerisindeki oyun tahtası argümanının **visit** metodunu, kendisini (this) parametre olarak vererek çağırmaktadır. Bu sayede analizler yapılırken, taş türleri **if** blokları ile tek tek kontrol edilmeyip daha jenerik ve esnek bir yöntemle oyun tahtası nesnesi taşları ayrı ayrı ziyaret (**visit**) etmektedir. Yani taşlarda ziyaretçi kabul (**accept**) etmektedir.

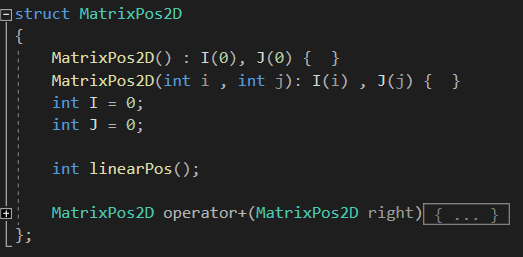


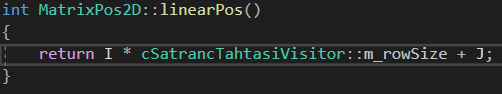
* 1. **Tahta Sınıfları**

Visitor Design Pattern kapsamında **visitor** rolünü gerçekleştirmek üzere abstract olarak **cOyunTahtasiVisitor** ve bunu gerçekleyen concrete **cSatrancTahtasiVisitor** sınıfları yazıldı. Visit metodlarında, argüman olarak aldığı taşların tehdit analizlerini yapmaktadır. Bu metodlar argüman olarak gelen taşın tehdit ettiği pozisyonları bir vektör içerisinde döndürmektedir.



* 1. **MatrixPos2D Veri Yapısı**

****

****

MatrixPos2D yapısı satranç tahtası üzerindeki kareleri matris koordinatları olarak ifade etmektedir. **linearPos** metodu ise 2-boyutlu matris koordinatlarından 1-boyutlu **int** index değeri döndürmektedir. Bu index, taşların tutulduğu bir map yapısında **key** olarak kullanılmaktadır.



* 1. **SatrancManager Sınıfı**

