Ödev-07

1. Ödev-07

1.1. Command Pattern

Bu örnekte bir satranç oyunundaki at ve piyon taşlarının nasıl hareket ettiği gösterilmektedir. Bu örnek yapılırken Command Pattern kullanılmıştır.

TasCommand sınıfı ile sadece ileri, geri, sağa ve sola gitme hareketleri tanımlanmıştır.

Atın veya piyonun yapacağı hareketler CompositTasCommand sınıfı kullanılarak belirlenmiştir.

```
#include <iostream>
#include <vector>

class Tas{
        ❶
public:
    int pozisyonX{0};
    int pozisyonY{0};

    Tas(int pozisyonX, int pozisyonY) : pozisyonX(pozisyonX),
 pozisyonY(pozisyonY) {}

    friend std::ostream &operator<<(std::ostream &os, const Tas &tas) {
        os << "pozisyonX: " << tas.pozisyonX << " pozisyonY: " <<
 tas.pozisyonY;
        return os;
    }
};

class Command{
        ❷
    virtual void hareketEt() = 0;
};

class TasCommand: public Command{
        ❸
```

```
public:
    Tas& tas;
    enum Hareket{ileri,geri,saga,sola} hareket;

    TasCommand(Tas &tas, Hareket hareket) : tas(tas), hareket(hareket)
 {}

    void hareketEt() override {
        ❹
        switch (hareket) {

            case ileri:
                tas.pozisyonY++;
                break;
            case geri:
                tas.pozisyonY--;
                break;
            case saga:
                tas.pozisyonX++;
                break;
            case sola:
                tas.pozisyonX--;
                break;
        }
    }
};

class CompositeTasCommand : public std::vector<TasCommand>, public
 Command{      ❺
public:
    CompositeTasCommand(const std::initializer_list<TasCommand>
 &commands) : vector(commands) {}

    void hareketEt() override {
        ❻
        for(auto cmd : *this)
            cmd.hareketEt();
    }
};

class At : public Tas, public CompositeTasCommand{
        ❼
public:
```

```
    At(int pozisyonX, int pozisyonY)
        ❽
            : Tas(pozisyonX, pozisyonY),
              CompositeTasCommand({
                                            TasCommand(*this,
 TasCommand::ileri),
                                            TasCommand(*this,
 TasCommand::ileri),
                                            TasCommand(*this,
 TasCommand::saga),
                                    }
              ) {}
};

class Piyon : public Tas, public CompositeTasCommand{
        ❾
public:
    Piyon(int pozisyonX, int pozisyonY)
        ❿
            : Tas(pozisyonX, pozisyonY),
              CompositeTasCommand({
                                            TasCommand(*this,
 TasCommand::ileri)
                                    }
              ) {}
};

int main()
{
    At a{1,0};
    Piyon p{4,1};
    std::cout << a << std::endl;
    std::cout << p << std::endl;
    a.hareketEt();
    p.hareketEt();
    std::cout << a << std::endl;
    std::cout << p << std::endl;
    return 0;
}
```

❶ Tas sınıfının tanımlanması. Buradaki **friend std::ostream &operator<<(std::ostream &os, const Tas &tas)** fonksiyonu ile taşın pozisyonu kolayca ekrana basılabiliyor.

❷ Command sınfının tanımlanması.

❸ TasCommand sınıfının tanımlanması. Bu sınıf ile taşın ileri, geri, sağa ve sola girme hareketleri gerçekleştirilir. CompositeTasCommand sınıfında kullanılmak üzere oluşturulmuştur.

❹ hareketEt override edilmesi. Bu fonksiyon ile hareketlerin pozisyonları nasıl etkileyeceği anlaşılmıştır. Her yöne sadece bir birim gidilmektedir.

❺ CompositeTasCommand sıfının tanımlanması. Bu sınıf bir dizi komutun tek fonksiyon ile çalışmasını sağlar.

❻ hareketEt fonksiyonunun override edilmesi. Dizide bulunan bütün komutların çalıştırılmasını sağlar.

❼ At sınıfının tanımlanması.

❽ At sınıfının constructor'ının tanımlanması. Bu fonksiyonda atın nasıl hareket etmesi gerektiği anlatılmaktadır.

❾ Piyon sınıfının tanımlanması.

❿ Piyon sınıfının constructor'ının tanımlanması. Bu fonksiyonda piyonun nasıl hareket etmesi gerektiği anlatılmaktadır.

UML Diagramı