Ödev-03

1. Ödev-03

1.1. Thread-Safe Singleton

Aşağıdaki örnek oyundaki bir düşmanı ve iki savaşçıyı göstermektedir.

Burada elimizdeki düşmandan farklı bir düşman yaratılmayacağı için düşman singleton sınıf olarak hazırlanmıştır.

Savaşçı sınıfı ise normal örneklenebilir sınıf olarak hazırlandı.

Savaşçılar aynı anda düşmana hasar vermektedir.

```
std::mutex
```

komutu ile herhangi bir sıkıntı çıkmadan bu işlem gerçekleştirilebilmektedir.

```
#include <iostream>
#include <future>
#include <mutex>
#ifdef _WIN32
#include <Windows.h>
#else
#include <unistd.h>
#endif

static std::mutex _mtx;

class Dusman {                                                    ❶

public:
    static Dusman* olustur() {
        _mtx.lock();                                              ❷
        if(m_instanceDusman == nullptr){                          ❸
            m_instanceDusman = new Dusman();
        }
        _mtx.unlock();                                            ❹
        return m_instanceDusman;
    }
```

```
    void hasar_al(int val){                                          ❺
        can -= val;
    }

    int get_can(){                                                   ❻
        return can;
    }

    int set_can(int val){                                            ❼
        can = val;
    }

private:
    Dusman() {                                                       ❽
        std::cout << "Dusman olusturuldu!\n";
    }
    ~Dusman() {}                                                     ❾

    int can = 100;
    static Dusman *m_instanceDusman;                                 ❿
};

Dusman* Dusman::m_instanceDusman = nullptr;                          11

class Savasci{                                                       12
public:
    Savasci(int id, int hasar) : id(id), hasar(hasar) {}

    int id;
    int hasar;
    int can = 100;
};

void hasarVer(Savasci *savasci){
    Dusman *obj1 = Dusman::olustur();                                13
    sleep(1);
    _mtx.lock();                                                     14
    obj1->hasar_al(savasci->hasar);
    std::cout << "Savasci_" << savasci->id <<" verilen hasar: "<<
 savasci->hasar << std::endl;
    std::cout << "Dusman kalan can: " << obj1->get_can() << std::endl;
    _mtx.unlock();
}
```

```
int main()
{
    std::cout<<"Singleton Example\n"<<std::endl;

    Savasci *sv1 = new Savasci(1,30);
    Savasci *sv2 = new Savasci(2,40);

    auto fut1= std::async(std::launch::async,hasarVer,sv1);          15
    auto fut2= std::async(std::launch::async,hasarVer,sv2);

    return 0;
}
```

❶ Dusman sinifi tanimlanir.

❷ **_mtx.lock()** komutu ile Dusman sınıfının örneği oluşturulurken threadlerin aynı anda belirtilen mutex aralığına erişmesi önlenir.

❸ Dusman sınıfının örneğinin daha önce oluşturulduğunu kontrol eden kısımdır.

❹ **_mtx.unlock()** komutu mutex aralığının bittiğini belirten fonksiyondur.

❺ Dusman sınıfındaki **can** değişkeninin değerini azaltan public fonksiyondur.

❻ Dusman sınıfındaki **can** değişkeninin değerini alan public fonksiyondur.

❼ Dusman sınıfındaki **can** değişkeninin değerini değiştiren public fonksiyondur.

❽ Dusman sınıfının private constructer fonksiyonudur. Private olmazsa singleton olmaz. Eğer public olursa bütün sınıflar Dusman sınıfından örnek oluşturabilir. Bu da Singleton prensibine aykırıdır.

❾ Dusman sınıfının private destructer fonksiyonudur.

❿ Dusman sınıfının kullanılabilecek tek örneğidir (instance). Statik yapılmazsa Singleton prensibine uygun davranmaz.

11 C++'da statik değişkene main fonksiyonu dışında bir değer atanması gerekmektedir. Bu satırda static sınıfa **nullptr** değeri atanır.

12 Savasci sinifi tanimlanir.

13 Bu satırda Singleton prensibinin kurallarına göre oluşturulan **olustur** fonksiyonu kullanılır. Bu fonksiyon Dusman sınıfından sadece bir tane örnek oluşturulduğunun kontrolünün yapılması için gereklidir.

14 std::cout komutlarının threadler arasında karışık olarak kullanılmaması için bu kısım mutex aralığına alınmıştır.

15 Bu satırda thread ile iki tane **hasarVer** fonksiyonu çalıştırılır.

UML Diagramı