Ödev-06

1. Ödev-06

1.1. Iterator Pattern

Bu örnekte basit bir iterator mantığı anlatılmaktadır. Item isminde bir class oluşturulmuştur. İçerisinde iki tane integer değişken bulunmaktadır.

Iterator, içerisinde Item değişkenini bulunduran ItemContainer sınıfı içindeki elemanlara ulaşmak için kullanılmaktadır.

Iterator normal ve tersine olmak üzere ikiye ayrılmıştır.

```
#include <iostream>

class Item{                                                    ❶
public:
    int id;
    int value;
};

template <typename T>
class Iterator {                                               ❷
public:
    virtual ~Iterator() {
    }

    virtual void first() = 0;
    virtual void next() = 0;
    virtual bool isDone() = 0;
    virtual T currentItem() = 0;
};

template <typename T>
class Container {                                              ❸
public:
    virtual Iterator<T> *createIterator() = 0;
    virtual Iterator<T> *createReverseIterator() = 0;
};
```

```
class ItemContainer : public Container<Item> {                    ❹
public:
    ItemContainer(unsigned int size) {
        list = new Item[size]();
        for (int i = 0; i < size; ++i) {
            list[i].id = i;
            list[i].value = size - i;
        }
        count = size;
    }

    Iterator<Item> *createIterator() override;                    ❺
    Iterator<Item> *createReverseIterator() override;             ❻

    ~ItemContainer() { delete[] list; }

    unsigned int size() const { return count; }

    Item at(unsigned int index) { return list[index]; }

private:
    Item *list;
    unsigned int count;

};

class ItemIterator : public Iterator<Item> {                      ❼
public:
    ItemIterator(ItemContainer *l) : cont(l), index(0) {}

    void first() override { index = 0; }
    void next() override { index++; }
    bool isDone() override { return (index >= cont->size()); }

    Item currentItem() override {
        if (isDone()) {
            return Item();
        }
        return cont->at(index);
    }

private:
    ItemContainer *cont;
    int index;
```

```
};

class ItemReverseIterator : public Iterator<Item> {                    ❽
public:
    ItemReverseIterator(ItemContainer *l) : cont(l), index(0) {}

    void first() override { index = cont->size() - 1; }
    void next() override { index--; }
    bool isDone() override { return (index < 0); }

    Item currentItem() override {
        if (isDone()) {
            return Item();
        }
        return cont->at(index);
    }

private:
    ItemContainer *cont;
    int index;
};

Iterator<Item> *ItemContainer::createIterator() {                      ❾
    return new ItemIterator(this);
}

Iterator<Item> *ItemContainer::createReverseIterator() {               ❿
    return new ItemReverseIterator(this);
}

int main() {
    unsigned int size = 12;
    ItemContainer cont = ItemContainer(size);

    Iterator<Item> *it = cont.createIterator();
    for (it->first(); !it->isDone(); it->next()) {
        std::cout << "Item id: " << it->currentItem().id << std::endl;
    }

    std::cout << std::endl;

    it = cont.createReverseIterator();
    for (it->first(); !it->isDone(); it->next()) {
        std::cout << "Item id: " << it->currentItem().id << std::endl;
```

```
    }

    return 0;
}
```

1. Item sınıfının tanımlanması.
2. Iterator sınıfının tanımlanması. Her türden sınıfa iterator oluşturabilmek için template olarak oluşturuldu.
3. Container sınıfının tanımlanması. Her türden sınıfa iterator oluşturabilmek için template olarak oluşturuldu.
4. ItemContainer sınıfının tanımlanması.
5. createIterator fonksiyonunun override edilmesi. Bu fonksiyon ile iterator baştan sona olacak şekilde containerı gezecektir.
6. createReverseIterator fonksiyonunun override edilmesi. Bu fonksiyon ile iterator sondan başa olacak şekilde containerı gezecektir.
7. ItemIterator sınıfının tanımlanması. ItemContainerı baştan sona olacak şekilde gezen iteratordır.
8. ItemReverseIterator sınıfının tanımlanması. ItemContainerı sondan başa olacak şekilde gezen iteratordır.
9. createIterator fonksiyonunun tanımlanması.
10. createReverseIterator fonksiyonunun tanımlanması.

UML Diagramı