2023 08 14

Classes

```
//class'larda member functions storege kaplamaz ancak data member yer kaplar
class Nec {
    public:
        void f1(int);
        void f2(int);
        void f3(int);
    private:
        int mx, my;
};

int main()
{
    // sizeof Nec = 8
    std::cout << "sizeof(Nec) = " << sizeof(Nec) << "\n";
}
```

```
/*
    isim arama önceliği;
        önce bloklarla(kendi scope ve dışındakı scopler)
        class'larda
        en son olarak namespace (global)

*/
```

this keyword

```
/*
this keyword
 1) this  bir keyword
 2) this  bir pointerdir  non static üye fonksiyon içinde
 hangi nesne için çağrıldıysa onun adresini döndürür.
 3) *this  o nesneni kendisini
*/
```

```
class Myclass
{
    public:
        void foo()
        {
            std::cout << "this = " << this <<"\n";
        }
    private:
        int mx{}, my{};
};
int main()
{
    Myclass m;
    // aynı adresleri döner
    std::cout << "&m = " << &m << "\n";
    m.foo();
}
```

```cpp
// this bir PR Value
class Myclass {
    public:
        void foo();
}

void Myclass::foo()
{
    Myclass m;
    this = &m; // syntax hatası
}
```

```cpp
class Tamer {
    public:
        Tamer* foo() {return this;}
        Tamer* bar() {return this;}
        Tamer* baz() {return this;}
}

int main()
{
    Tamer* p = new Tamer;
    p->bar()->foo()->baz();
}
```

```cpp
// const member functions
class Nec {
    public:
        // Nec*
        void foo(); // non-const member function
        // const Nec*
        void bar() const; // const member function
};
```

```cpp
class Myclass {
    public:
        void foo() const
        {
            mx = 56; // sytantax hatası

            Myclass m;
            m.mx = 89; // legal

            // const T* --> T*
            bar(); // illegal
        }

        void bar()
        {
            // T* --> const T*
            foo(); // legal
        }
    private:
        int mx;
}
```

```
class Myclass
{
    public:
        // overloading
        void foo();
        void foo() const;

        void bar();
    private:
        int mx;
}

int main()
{
    const Myclass m;
    m.foo(); // legal
    m.bar(); // illegal;
}
```

Mutable

```
// mutable keyword
class Fighter
{
    public:
        void foo() const
        {
            ++debug_call_count; // legal
            ++m_age; // illegal
        }
    private:
        std::string m_name;
        int m_age;
        int m_power;
        mutable int debug_call_count;
}
```

1. const üye fonks içinde sınıfın non-static veri elemanlarına atama yapamayız
2. const üye fon içinde sınıfın non-const üye fonksiyonlarını çağıramayız
3. const sınıf nesnelerini içinde sadece sınıf const üye fonks çağırabiliriz
4. const sınıf nesneleri için sınıfın non-const üye fonksiyonları çağıramayız