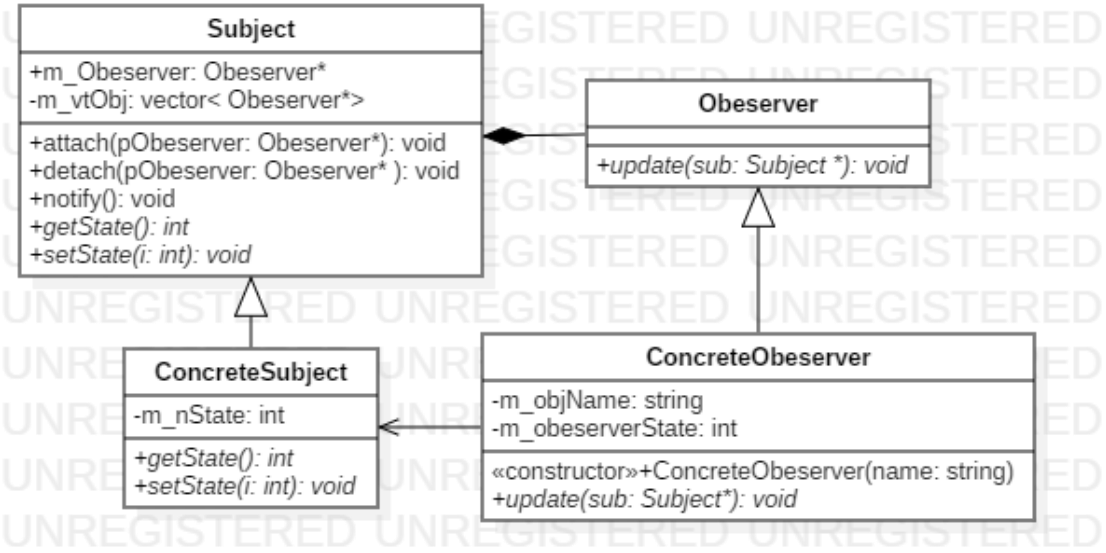
模式定义

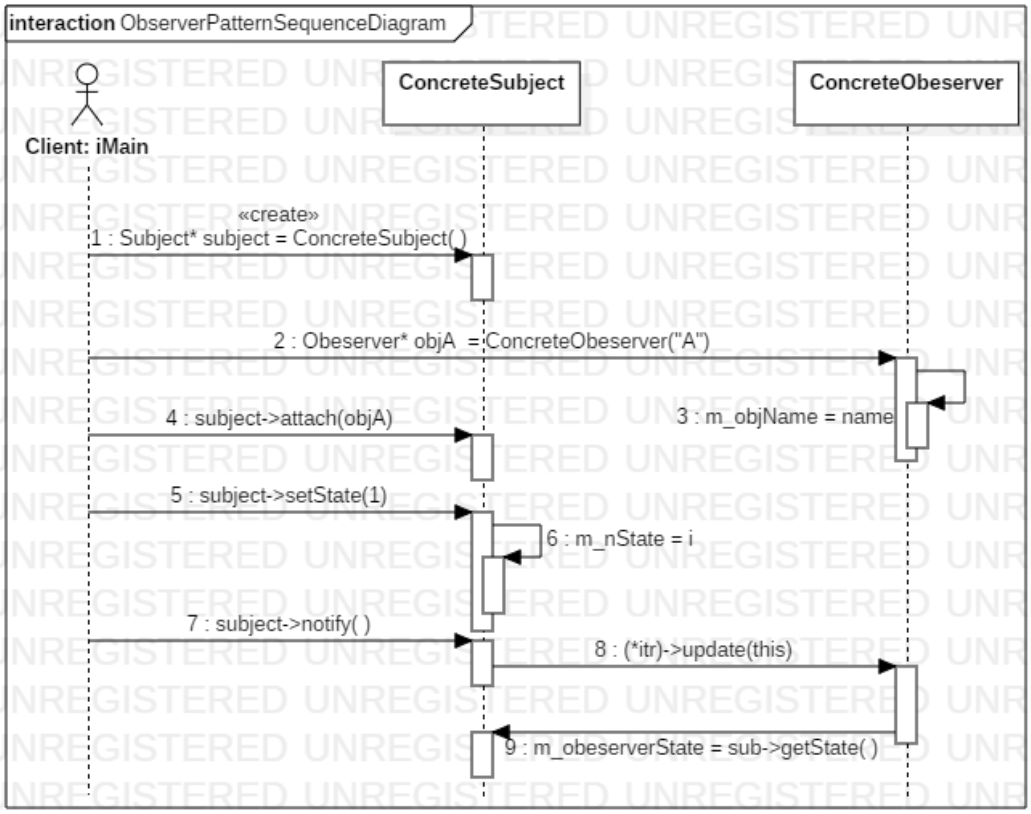
观察者模式：定义对象间的一种一对多依赖关系，使得每当一个对象状态发生改变时，其相关依赖对象皆得到通知并自动更新。观察者模式有叫做发布_订阅模式、模型_视图模式、源_监听模式或从属者模式。

观察者模式是一种对象行为型模式。

模式结构



时序图



关键代码

```
int main()
{
    Subject* subject = new ConcreteSubject();
    Obeserver* objA = new ConcreteObeserver("A");
    Obeserver* objB = new ConcreteObeserver("B");

    subject->attach(objA);
    subject->attach(objB);

    subject->setState(1);
    subject->notify();

    cout << "--------------------" << endl;

    subject->detach(objB);
    subject->setState(2);
    subject->notify();

    delete subject;
    delete objA;
    delete objB;
```

```cpp
ConcreteObeserver::ConcreteObeserver(string name)
{
    m_objName = name;
}

ConcreteObeserver::ConcreteObeserver(string name)
{
    m_objName = name;
}

void Subject::attach(Obeserver * pObeserver)
{
    m_vtObj.push_back(pObeserver);
}

void ConcreteSubject::setState(int i)
{
    m_nState = i;
}

void Subject::notify()
{
    for (vector<Obeserver*>::iterator itr = m_vtObj.begin(); itr != m_vtObj.end(); itr++)
    {
        (*itr)->update(this);
    }
}

void ConcreteObeserver::update(Subject * sub)
{
    m_obeserverState = sub->getState();
    cout << "update oberserver[" << m_objName << "] state:" << m_obeserverState << endl;
}

int ConcreteSubject::getState()
{
    return m_nState;
}
```

测试结果

```cpp
int main()
{
    Subject* subject = new ConcreteSubject();
    Obeserver* objA = new ConcreteObeserver("A");
    Obeserver* objB = new ConcreteObeserver("B");

    subject->attach(objA);
    subject->attach(objB);

    subject->setState(1);
    subject->notify();

    cout << "--------------------" << endl;

    subject->detach(objB);
    subject->setState(2);
    subject->notify();

    delete subject;
    delete objA;
    delete objB;

    system("pause");
```