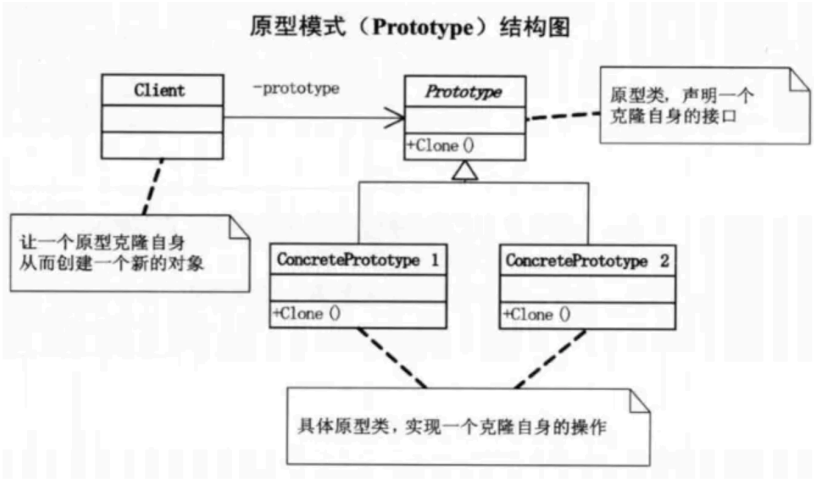
在GOF的《设计模式：可复用面向对象软件的基础》中是这样说的：用原型实例指定创建对象的种类，并且通过拷贝这些原型创建新的对象。这这个定义中，最重要的一个词是“拷贝”，也就是口头上的复制，而这个拷贝，也就是原型模式的精髓所在。

原型模式（Prototype Pattern）是用于创建重复的对象，同时又能保证性能。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。这种模式是实现了一个原型接口，该接口用于创建当前对象的克隆。当直接创建对象的代价比较大时，则采用这种模式。

原型模式提供了一个通过已存在对象进行新对象创建的接口（Clone），Clone（）实现和具体的实现语言相关，在 C++中我们将通过拷贝构造函数实现之。



由于克隆需要一个原型，而上面的类图中Prototype就这个原型，Prototype定义了克隆自身的Clone接口，由派生类进行实现，而实现原型模式的重点就在于这个Clone接口的实现。ConcretePrototype1类和ConcretePrototype2类继承自Prototype类，并实现Clone接口，实现克隆自身的操作；同时，在ConcretePrototype1类和ConcretePrototype2类中需要重写默认的复制构造函数，供Clone函数调用，Clone就是通过在内部调用重写的复制构造函数实现的。

```
#include <iostream>
#include <string>
using namespace std;
class ProtoType{
public:
    virtual ~ProtoType(){
        delete [] m_pDescribe;
    }
    ProtoType(string str,const char *pSrc){
        m_strName = str;
        int nLen = strlen(pSrc);
        m_pDescribe = new char[nLen + 1];
        cout << "Describe内存地址:" << &m_pDescribe << endl;
        strcpy(m_pDescribe,pSrc);//深度克隆
        m_pDescribe[nLen] = '\0';
    }
    virtual ProtoType* Clone() = 0;
    char *GetDescribe(){
        return m_pDescribe;
    }
    string GetName(){
```

```
        return m_strName;
    }
private:
    string m_strName;
    char *m_pDescribe;
};
class ConcretProtoTypeA : public ProtoType{
public:
    ConcretProtoTypeA(string str,const char *pSrc):ProtoType(str,pSrc){}
    ConcretProtoTypeA(ConcretProtoTypeA &other):ProtoType(other.GetName(),other.GetDescribe()){}
    ProtoType* Clone(){
        return new ConcretProtoTypeA(*this);
    }
};
class ConcretProtoTypeB : public ProtoType{
public:
    ConcretProtoTypeB(string str,const char *pSrc):ProtoType(str,pSrc){}
    ConcretProtoTypeB(ConcretProtoTypeA &other):ProtoType(other.GetName(),other.GetDescribe()){}
    ProtoType* Clone(){
        return new ConcretProtoTypeB(*this);
    }
};
int main(){
    const char *pDescribe = "I Love You";
    ProtoType *pProto = new ConcretProtoTypeA("Honey", pDescribe);
    ProtoType *pClone = pProto->Clone();
    cout << pClone->GetName() << ", Hi, " << pClone->GetDescribe() << endl;
    return 0;
}
```

```
192:DesignPattnsStudy weishichun$ ls ProtoType
ProtoType_1.cpp        ProtoType原型模式.pdf
192:DesignPattnsStudy weishichun$ g++ -o ProtoType.out ProtoType_1.cpp
192:DesignPattnsStudy weishichun$ ./ProtoType.out
Describe内存地址:0x7f8eaac017c0
Describe内存地址:0x7f8eaac01800
Honey, Hi, I Love You
192:DesignPattnsStudy weishichun$
```