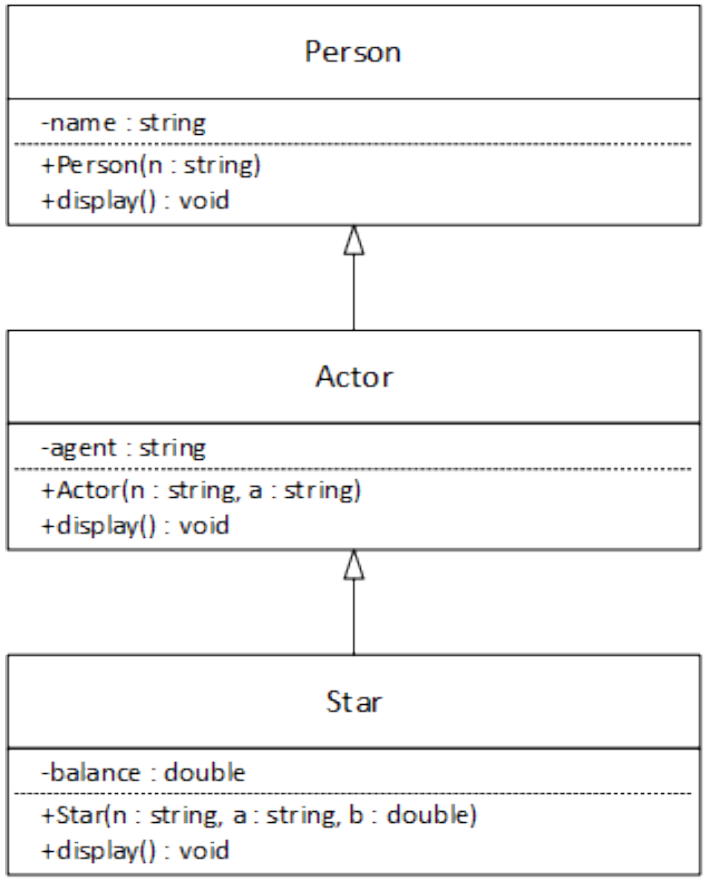
# ACTOR 1

Implementing and using basic inheritance

Delroy A. Brinkerhoff




# ACTOR 1 CLASS DIAGRAM

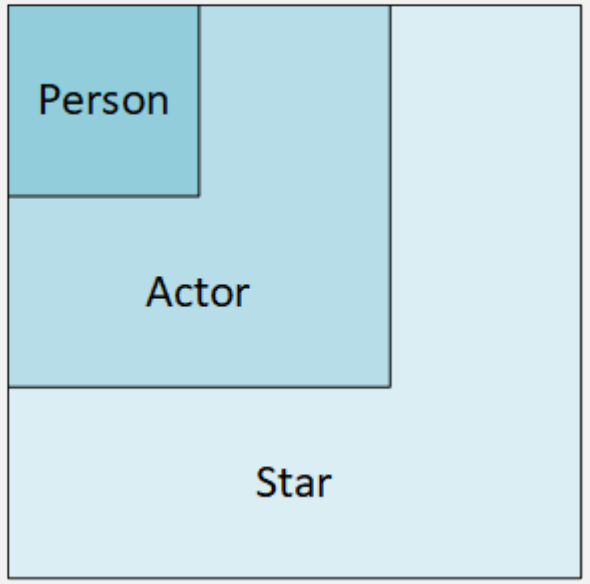
- Actor examples don't solve "real" problems
- Actor 1: all classes in a single file
- An inheritance hierarchy can be arbitrarily tall and wide
- Information is "pushed" upward through chained constructor calls
- Information is "pulled" downward through chained display function calls
- You can generalize the display function syntax to call other overridden member functions

# main

```
int main()
{
    // Automatic variable/object
    Star s("John Wayne", "Cranston Snort", 50000000);
    s.display();

    // Dynamic variable/object
    Star* s2 = new Star("John Wayne", "Cranston Snort", 50000000);
    s2->display();

    return 0;
}
```

# BUILDING INHERITANCE

```
class Person
{
    ...
};

class Actor : public Person
{
    ...
};

class Star : public Actor
{
    ...
};
```



# THE Person CLASS

```
class Person
{
    private:
        string    name;

    public:
        Person(string n) : name(n) {}

        void display() { cout << name << endl; }
};
```

# THE Actor CLASS

```
class Actor : public Person
{
    private:
        string    agent;

    public:
        Actor(string n, string a) : Person(n), agent(a) {}

        void display()
        {
            Person::display();
            cout << agent << endl;
        }
};
```

# THE Star CLASS

```
class Star : public Actor
{
    private:
        double    balance;

    public:
        Star(string n, string a, double b) : Actor(n, a), balance(b) {}

        void display()
        {
            Actor::display();
            cout << balance << endl;
        }
};
```