

Inheritance

Jonathan Windle

University of East Anglia

J.Windle@uea.ac.uk

May 30, 2017

Overview I

1 Intro

2 Inheritance

- Base class
- Sub Class

3 Polymorphic pointers and references

- Advantages:
 - Encourages code use
 - Encourages modularity, data hiding and encapsulation
 - Makes programs maintainable and easily extendable
- Disadvantages:
 - Can lead to overly complicated implementations.
 - Introduces computational overheads.

Inheritance

- Abstract Base class:

```
class Speaker
{
private:
// Inherited, but not visible in sub-class
    string utterance;
public:
// Can be overridden, inherited and visible in sub-class
    virtual string getUtterance() {
        return utterance;
    }
// Pure virtual, must be overridden before instantiation
    virtual void speak() = 0;

    Speaker(string utterance) {
        this->utterance = utterance;
    }
};
```

Sub Class

- Concrete Sub Class:

```
class Philosopher : public Speaker
{
public:
    // Override pure virtual method, can now instantiate
    void speak() {
        cout << "My philosophy is..." << getUtterance() << endl;
    }

    // Add new methods to inherited behaviour
    inline void philosophise() {
        cout << "But what do I mean..." << getUtterance() << endl;
    }

    // Over-ride inherited virtual method, these cannot be inline
    string getUtterance() {
        return "Woof-woof";
    }

    // Call super-class constructor
    Philosopher(string philosophy) : Speaker(philosophy)
    {
    }
};
```

Polymorphic pointers and references

- Superclass pointers can point to sub-class objects.
- Superclass references can reference sub-class objects.
- Can only access methods present in the super-class.

```
int main(int argc, char* argv[]) {
    Philosopher descartes("I think, therefore I am");
    Dog rover;
    Speaker* speaker = &descartes;

    // Uses speak in Philosopher class
    speaker->speak();
    speaker = &rover;

    // Uses speak in Dog class
    speaker->speak();
    Speaker &orater = descartes;

    // Pointers and references can be polymorphic
    orater.speak();
    return 0;
}
```

The End