Inheritance and Virtual Functions

11-26-2001

Opening Discussion

- Do you have any questions about the quiz?
- What did we talk about last class?

Two Roles of Inheritance

- Inheritance plays two roles in programming languages like C++. We have discussed these over the last two classes.
  - Subtyping - When one type inherits from another it becomes a subtype of it.
  - Reuse - Methods defined for a superclass can be called for instances of a subclass without being rewritten.

When Not to Reuse

- At the end of last class I asked you if you always want to reuse code from a superclass when you inherit from it. As you can now guess from the title of this slide, the answer is no.
- Sometimes subclasses need to do the same type of thing, but they need to do different specific actions. That is, they need methods with the same names but different implementations.

Virtual Functions

- When you want different subclasses to have different implementations of a method, that method needs to be declared virtual.
- This is done simply by putting the virtual keyword before the return type.

```
virtual int decideMove();
virtual void calcValue();
```

Virtual vs. Non-virtual

- When a function takes as an argument a reference or pointer to a supertype when a non-virtual function is called it always calls the method of the supertype, even if the subtype overloads it.
- With a virtual function it calls the "closest" declared method.
- We aren't going to discuss the issue of scoping non-virtual functions.

Once Virtual, Always Virtual

- Once a function has been declared virtual in a superclass it is virtual in all the subclasses in C++. As a result, the subclasses don't have to explicitly say it is virtual.
- However, I would recommend that if a function is virtual you call it such everywhere so that people don't have to look to superclasses to find out.

Pure Virtual

- A virtual function declaration in a class can be made pure virtual which implies that the method has no implementation of the superclass. Classes with outstanding pure virtual functions can't be instantiated.

```
virtual int decideMove()=0;
virtual void calcValue()=0;
```

Look at Code

- Let's now go add a virtual function to the Pac-Man code that we started working on in the last class.
- Last class we put in a function that had the entity move one step in the direction that it was set to move in. They also need to decide what direction to move in at a given time. This should not be the same for all of them and hence should be virtual.

Minute Essay

- Try to think of an example of an instance where inheritance helps describe a logical ordering of types. What behaviors I this case might be virtual?
- We will not be having class the rest of this week. Feel free to use the time to work on assignment #7 and maybe even to try writing a little piece of code that uses inheritance.