Announcements

• No quiz next week 😊
  – In fact, no more quizzes until November…
• Recitations this week: mandatory
  – Google Test
• Recitations next week: mandatory
  – More Google Test
• PA4 due today
  – Any questions?

Object Oriented Programming (Review)

1. Encapsulation
   – Collect data & code that operates on that data in one object
   – Provide a single, public interface
   – Changes to the implementation are local

2. Polymorphism (inheritance)
   – Abstraction via “is-a” relation
   – Write code at different levels of abstract
   – Avoid redundant code

3. Inheritance as Union
   – A child class is the union of parent and new field/methods

Dispatch

• How methods are selected and invoked
  • Methods/functions selected by signature
    – Name must match (e.g. WarmUp)
    – Arguments must match
      • Data types must match
      • Including hidden arguments
    – Return types not used for selection/dispatch
  • Polymorphism creates ambiguities
    – Animal::WarmUp();
    – Mammal::WarmUp();

Dispatch (II)

• Static Dispatch
  – Functions selected at compile time
  – Based on declared data types
  – This is how C dispatches functions
  – This is the C++ default
• Dynamic Dispatch
  – Functions selected at run-time
  – Based on run-time data type of object
    • Run-time type is “sticky”
    • But does not apply to copies
  – This is the Java default
  – Signaled by the ‘virtual’ keyword in C++
Review: WarmUp()

class Animal {
    public:
    bool WarmUp() {return false;}
    void Behave() {WarmUp();}
};
class Mammal : public Animal {
    public:
    bool WarmUp() {shiver(); return true;}
};
Mammal m;
m.Behave();

• Note that virtual keyword never appears
• Therefore dispatch will be static

WarmUp() version 2

class Animal {
    public:
    virtual bool WarmUp() {return false;}
    void Behave() {WarmUp();}
};
class Mammal : public Animal {
    public:
    virtual bool WarmUp() {shiver(); return true;}
};
Mammal m;
m.Behave();

Implementation: Mammal

• How are Mammal objects implemented?
  — What is sizeof(Mammal)?
    • Same caveat as before...
    • Every Mammal contains an Animal object
    • Mammals also have their own int field (1: body_temp)
    • sizeof(Mammal) == (sizeof(Animal) + sizeof(int))
  — How is Mammal organized?
    • The first thing in a Mammal is its Animal
    • As a result, any Mammal* is also an Animal*

Organization of Inheritance

• Assume Quagga inherits Equine
• Equine inherits Mammal
• Mammal inherits Animal

Virtual Function Pointer Table (VFPT)

• Every class with virtual functions has a VFPT
• Every entry in this table is the address of a virtual method
• If LifeSpan() is the 1st entry for Animal, it is also the 1st entry for every derived class from Animal
  — Even if the implementation changes
• Note: one table per class, not per object

Implementing Dynamic Dispatch

• There is 1 exception to inheritance == union
• For objects with virtual functions
  — The first field is a pointer to the VFPT
  — Before any instances of parent classes
  — It points to the VFPT of the object as created
Implementing Dynamic Dispatch II

• Dynamic dispatch is implemented through the VFPT

```
void Foo(Animal& a)
{
    a.Lifespan() = 75;
}
```

What happens?
The compiler jumps to the address in the VFPT* (1st field)
This will be the VFPT of the object as it was created
Lifespan will be the 1st entry in this table (selected by name)
Jump to the code at that address

```
// a is an Animal (or derived from it)
// Lifespan is a virtual function
// compiler generates (pseudo-code)
// a is the start of the object, and therefore also the VFPT ptr
// Follow that ptr to get to VFPT
// addr = *a
// offset of Lifespan calculated at compile time
// same for Animal, Mammal, ...
// addr += offset(Lifespan)
// this is the actual function call

jump to *addr
```

Implementation Illustrated