Name: _____

Today's Date: _____

Today's Goals

- Complete course evaluations
- Recognize when to use virtual member functions
- Describe dynamic dispatch

Today's Question(s)

What's one thing you'd like us to cover in Friday's review session?

Lingering Questions

Reminder: CS70 Petting Zoo

```
void pet(Animal animal)
{
    animal.speak();
}


int main() {
    Cow bessie;
    Raptor peri;

    pet(bessie);
    pet(peri);

    return 0;
}
```

Class Exercise: (Bad) Alternatives

Why are these not good ideas?

Dynamic Dispatch

In C++, dynamic dispatch must be explicit, using the `virtual` keyword

```
// Animal is a "virtual class"
class Animal {
public:
    virtual void speak() const;
};
```

Dynamic Dispatch and Derived Classes

Once a member function is virtual, it's virtual all the way down the hierarchy.

Worksheet: What does the program print, and why?

Worksheet: Which of these work, and what do they do?

Derived Classes and Destructors

Consider these two classes:

```
class Animal {
    Leg* legs_;
public:
    Animal() {
        legs_ = new Leg[...];
    }
   ~Animal() {
        // TODO
    }
};
```

```
class Cow : public Animal {
    Spot * spots_;
public:
    Cow() {
        spots_ = new Spot[...];
    }
    ~Cow() {
        // TODO
    }
};
```

Which class should destroy what?

```
Animal* allocateAnAnimal() {
    return new Cow;
}

Animal* c = allocateAnAnimal();
delete c;
```

Pure virtual member functions

Arrays and Base Classes

This also means that we can't make arrays of `Animals`:

```
    Animal zoo[2];
```

```
error: array of abstract class type 'Animal'
    Animal zoo[2];

./animal.hpp:7:18: note: unimplemented pure virtual method
    virtual void speak() const = 0;
```

How do you make a "zoo" of animals, then?

```
vector< TODO > zoo;

for (size_t i = 0; i < 5; ++i) {
    // TODO Put a Cow in the zoo
}

for (size_t i = 0; i < 5; ++i) {
    // TODO Put a Raptor in the zoo
}

for (auto animal : zoo) {
    // TODO Make the animal speak
}
```

What does this program print, and why?

```
Cow noel;
Animal a = noel;
noel.speak();
a.speak();

Animal & ar = noel;
ar.speak();

Animal * ap = new Cow;
ap->speak(); // remember to delete ap!
```

Which of these work, and what do they do?

```
Animal a;
a.speak();

Cow c;
c.speak();

Animal a;
a.milk();

Cow c;
c.milk();

Cow c;
Animal a = c;
a.speak();
a.milk();

Cow c;
Animal & a = c;
a.speak();
a.milk();

Cow c;
Animal * a = &c;
a->speak();
a->milk();

Cow c;
Animal & a = c;
Cow & c2 = a;
c2.speak();
c2.milk();

Cow c;
Animal * a = &c;
Cow * c2 = a;
```

```
c2->speak();
c2->milk();
```