Inheritance

Consider the following classes:

```
// class spec for brian class

class brian {
public:
        brian( ) { cout << "I know C++. . ." << endl;}
        void speak( ) { cout << "I am the Lord of CSE 143!" << endl; }
};
```

```
// class spec for jeff class

class jeff : public brian {
public:
        jeff( ) { cout << ". . .and everyone loves me for it!" << endl;}
        void speak( ) { cout << "I sing most beautifully!" << endl; }
        void moo( ) { cout << "I like cows." << endl; }
};
```

```
// class spec for steve class

class steve : public jeff {
public:
        steve( ) { cout << ". . .And I can also do tricks!" << endl;}
        void speak( ) { cout << "I know kung-fu." << endl; }
        void star( ) { cout << "I am a superstar!" << endl; }
};
```

What does the picture of the class relationships look like? (you tell me. . .)

Static Dispatch

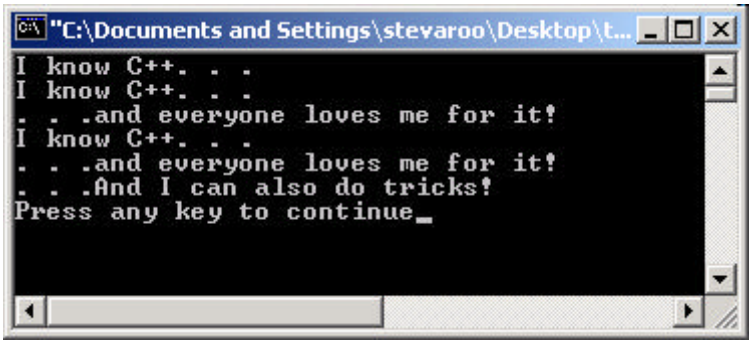
Given the code on the previous slide, what do you think is printed out when the following is executed?

```
//main file
#include "brian.h"
#include "jeff.h"
#include "steve.h"

int main ( ) {
        brian test1;
        jeff test2;
        steve test3;

        return 0;
}
```

The answer. . .



Note the order of the output!!! The topmost base level class constructors are run FIRST ALWAYS.

Static Dispatch, part 2

What's the output when the following is run (assume the same classes as before)?

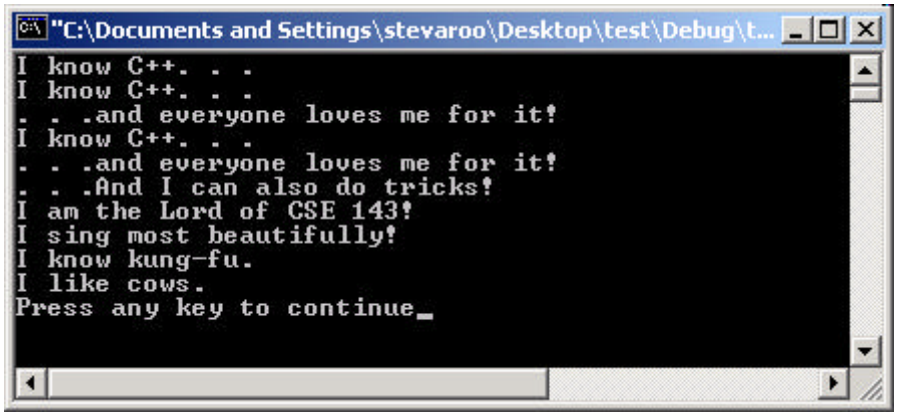
Are there any errors?

```
//main file
#include "brian.h"
#include "jeff.h"
#include "steve.h"
int main ( ) {
        brian BRIAN; jeff JEFF; steve STEVE;

        BRIAN.speak(); // is this ok?
        JEFF.speak();  // how about this?
        STEVE.speak(); // and this one?
        STEVE.moo();   // what about THIS one?

        return 0;
}
```

Nope, no errors! Here's what it does:



This is called **Static Dispatch**. The computer sees what type of variable you're trying to call the function on and chooses the function that's either in that class or a class above it.

Now, what about *THIS*:

BRIAN.moo();

Would this work? Why or why not?