

Dynamic Dispatch

So we've now covered *Inheritance* and *Static Dispatch*. . .now how does inheritance work with dynamic memory?

Remember our original classes? What is outputted when the following is run?

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

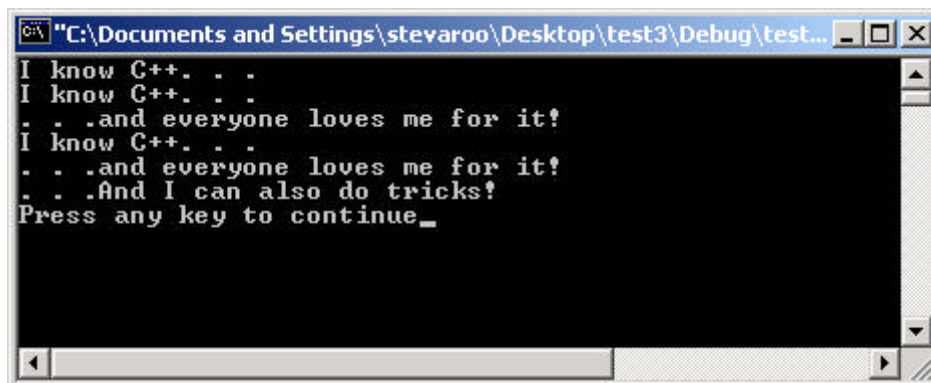
int main ( ) {
    brian * BRIAN;
    BRIAN = new brian(); // no problem, right?

    brian * BRIAN2;
    BRIAN2 = new jeff(); // uh-oh. . .what about this?

    brian * BRIAN3;
    BRIAN3 = new steve(); // weird. . ??

    return 0;
}
```

The following is outputted:



What if I tried to do this?

```
steve * STEVE;
STEVE = new brian(); // does this work?
```

No, it doesn't work. Why?

Dynamic Dispatch, part 2

Now consider the following program:

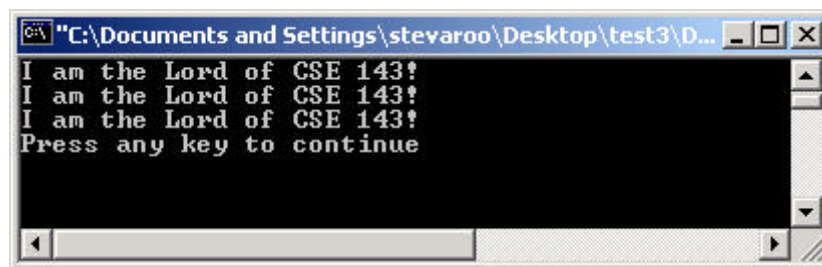
```
//main file
#include "brian.h"
#include "jeff.h"
#include "steve.h"

int main ( ) {
    brian * BRIAN;
    BRIAN = new brian();
    brian * BRIAN2;
    BRIAN2 = new jeff();
    brian * BRIAN3;
    BRIAN3 = new steve();

    BRIAN->speak();
    BRIAN2->speak();
    BRIAN3->speak();

    return 0;
}
```

This program compiles fine, but it might not do what you thought it would!
Here's what is outputs:

A screenshot of a Windows command prompt window. The title bar reads "C:\Documents and Settings\stevaroo\Desktop\test3\D...". The window contains the following text:

```
I am the Lord of CSE 143!
I am the Lord of CSE 143!
I am the Lord of CSE 143!
Press any key to continue
```

What if we wanted to call the speak methods for the derived classes? Can we do that using just a base class pointer?

...You bet we can!

Dynamic Dispatch, part 3

The key word is:

virtual

And we use it like so:

```
// class spec for brian class

class brian {
public:
    brian( ) { cout << "I know C++. . ." << endl;}
    virtual 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;}
    virtual void speak( ) { cout << "I sing most beautifully!" << endl; }
};
```

```
// class spec for steve class

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

Note that we don't really need the virtual in class steve, since it's a derived class. However, we put it there anyways since it lets us know what's going on.

Dynamic Dispatch, part 4

NOW what does this program output?

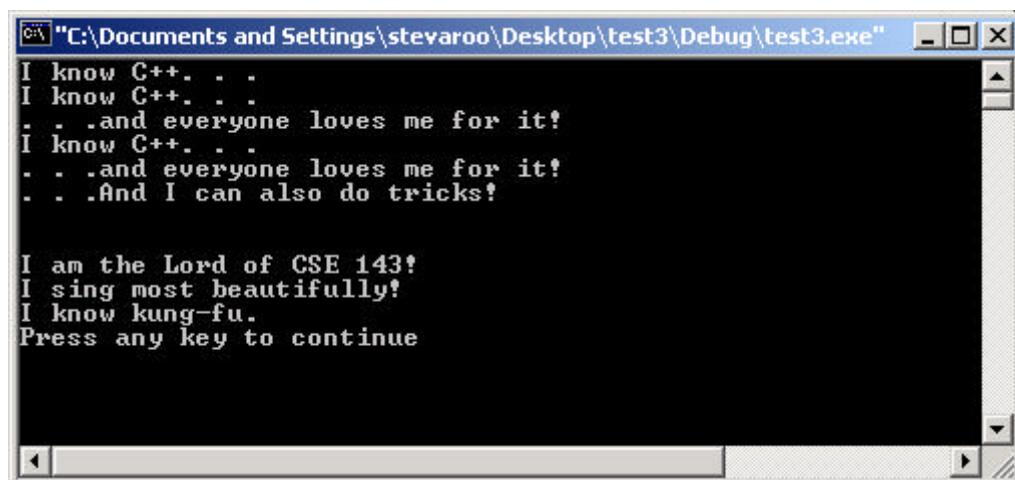
```
//main file
#include "brian.h"
#include "jeff.h"
#include "steve.h"

int main ( ) {
    brian * BRIAN;
    BRIAN = new brian();
    brian * BRIAN2;
    BRIAN2 = new jeff();
    brian * BRIAN3;
    BRIAN3 = new steve();

    BRIAN->speak();
    BRIAN2->speak();
    BRIAN3->speak();

    return 0;
}
```

You guessed it!



The screenshot shows a Windows command prompt window with the title bar "C:\Documents and Settings\stevaroo\Desktop\test3\Debug\test3.exe". The window contains the following text output from the program:

```
I know C++. . .
I know C++. . .
. . .and everyone loves me for it!
I know C++. . .
. . .and everyone loves me for it!
. . .And I can also do tricks!

I am the Lord of CSE 143!
I sing most beautifully!
I know kung-fu.
Press any key to continue
```