# Lab 5.2

*Out: Thursday, March 3rd, 2005*

## What you'll learn.

In this lab, you will learn how to subclass in C++ and how to override methods.

## How you'll do it.

You will subclass from your Image class from lab 5.1, allowing the image to represent a rotated image. You will override the getPixel(), getWidth(), and getHeight() methods in the subclass. Now, when the image is written out, it will come out as a rotated image.

## 1 Declaring a Subclass

Subclassing in C++ is pretty straight forward.

Let's say we have an Animal class:

```
class Animal{
     public:
        Animal(Color c);
        virtual ~Animal();

        void eat();
        void sleep();
        virtual void breathe();
};
```

Now, let's say we want to subclass this Animal class and create a Goldfish class. Goldfish are pretty particular about breathing with gills, so we are going to need to override the `Animal::breathe()` method. Because of that, we need to tell the compiler that we are going to want to override the method by putting the keyword `virtual` before the function name; every method that is going to be overridden **must** be defined `virtual`. Here is the definition for Goldfish:

```
class Goldfish : public Animal{
     public:
        Goldfish();
        virtual ~Goldfish();
```

```
        void breathe();
}
```

Notice that the Goldfish constructor does not take a Color. That's because, as everyone knows, goldfish are gold. Here is an example of a default constructor for a Goldfish:

```
Goldfish::Goldfish() : Animal(Color.gold){
}
```

You can see above an example of a superclass constructor being invoked within the subclass constructor.

In this lab, you are going to write a subclass of the Image class from last time that can represent a rotatable image. This image can be in only four states: as read, or rotated 90, 180, or 270 degrees. Think about what you would need to add to your Image class to accomplish this, then think about how you would take advantage of subclassing to put these changes into another class.

> **Task:** Create a new header file for a class named RotatableImage. Make it subclass from your Image class from last time. Adjust your class definition for Image to allow for the subclassing. The RotatableImage should have the same constructors as an Image. In the RotatableImage, override the following methods:
> `int getWidth();`
> `int getHeight();`
> `Color getPixel(int i, int j);`
> Also, add new methods to RotatableImage named `void rotate90(bool dir);` and `int getRotation();`. You may need to add some instance variables to keep track of state.

## 2 Implementing the Subclass

Now that you've filled in the header file, you can write the actual implementation of the `RotatableImage` class.

> **Task:** Fill in the class.
> First write out the skeleton. Make the constructors call the appropriate superclass constructors. Write `rotate90()` and `getRotation()`. Fill in `getHeight()`, `getWidth()`, and `getPixel()`.

## 3 Checking your Implementation

We want to make sure this works.

**Task:** Write a new main function. It should :
1. Instantiate a RotatableImage from an image in a file.
2. Rotate the image.
3. Write the image back to a file.

Open the file with `xv` to see if the image was rotated. If you want to make your program really cool, have it take as an argument by how much to rotate the image. This will make it easier to show off.

## You're done!