

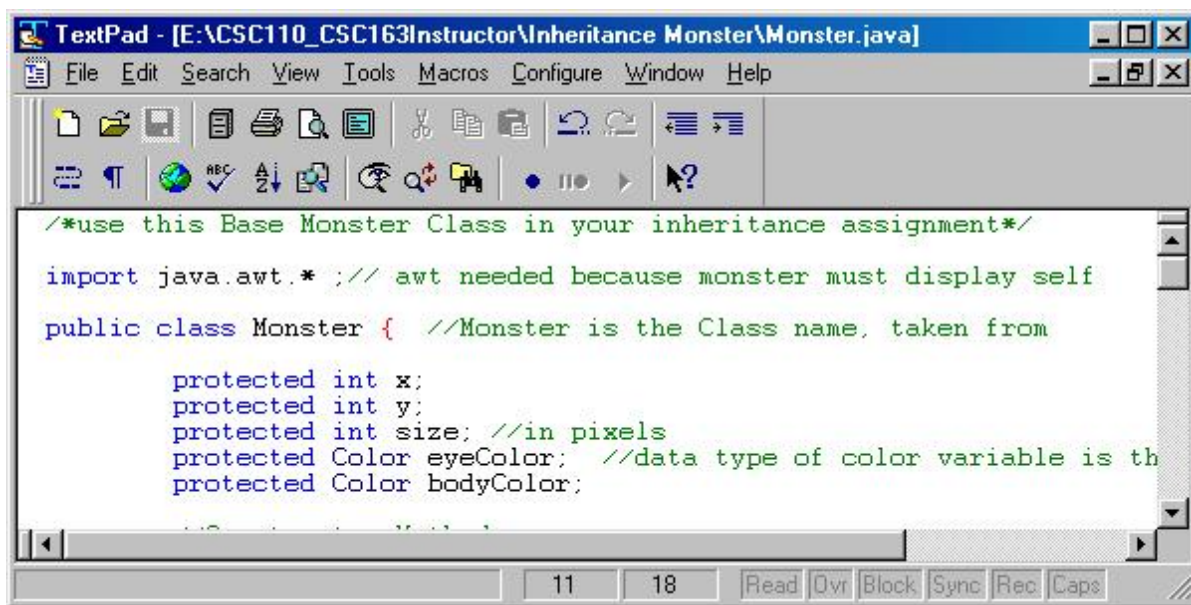
MONSTER INHERITANCE

Borrow a Monster class and MonsterControl class from your instructor. They are listed on the schedule as **Base Monster Class** and **Base Monster Controller**. **Tip: Please store these files in a separate directory called *Inheritance Monster* so that they will not be confused with earlier assignments.**

Use inheritance (extends) to create a new derived Monster2 class that adds 2 new things to what the borrowed base Monster class does (2 new methods). You may do more, if you like. Here's how:

A. CHANGE THE MONSTER PARENT CLASS

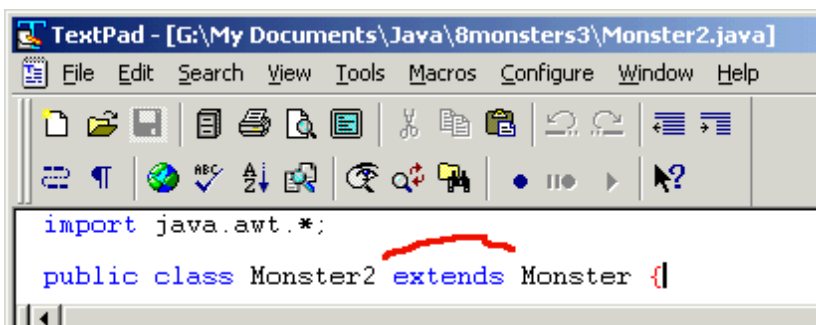
Change your instance variables from **private** to **protected**. That way, your derived class will be able to use them. (From the perspective of object-oriented programming, it's not "proper" to use the protected keyword. Instead, you should write accessor functions for *every* private variable you want your derived class to use. The protected keyword is the "practical" approach. In this case, practical means quicker and easier, so we're going to do it the practical way.) This is the only change you need to make to the parent class.



```
/*use this Base Monster Class in your inheritance assignment*/
import java.awt.*; // awt needed because monster must display self
public class Monster { //Monster is the Class name, taken from
    protected int x;
    protected int y;
    protected int size; //in pixels
    protected Color eyeColor; //data type of color variable is th
    protected Color bodyColor;
```

B. CREATE THE MONSTER2 CHILD CLASS

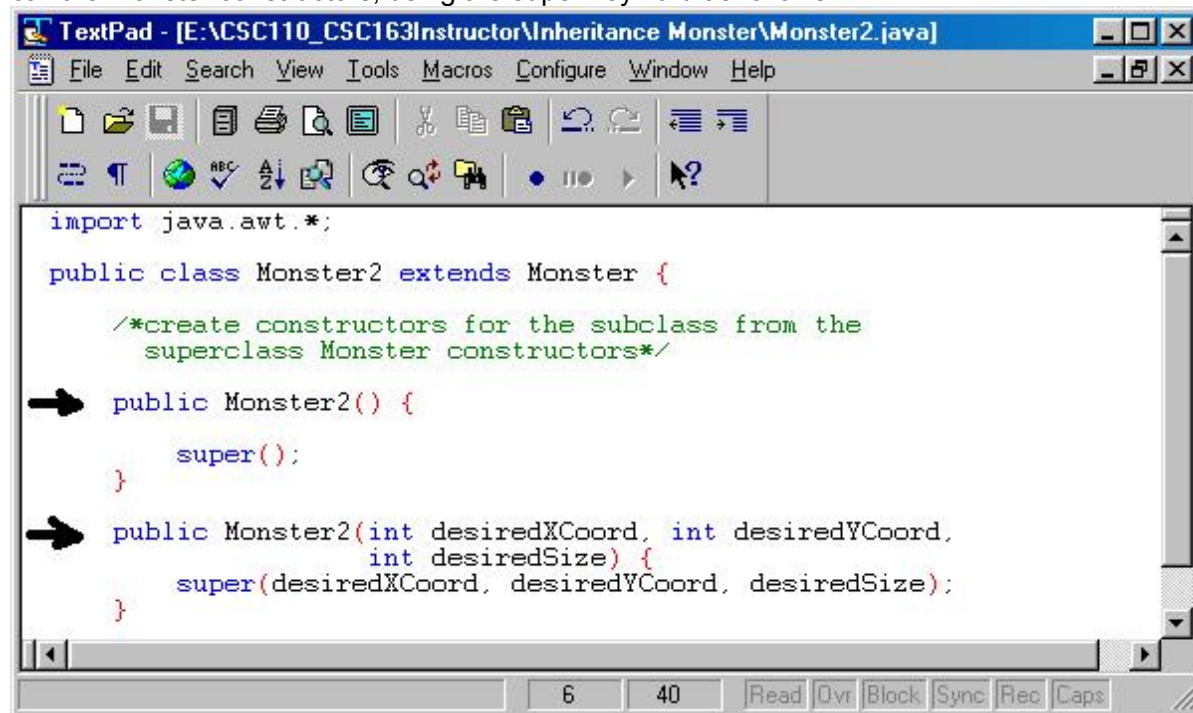
1. Create a new file for the Monster2 class.
2. Create the new class and use the extends keyword:



```
import java.awt.*;
public class Monster2 extends Monster {
```

3. Unfortunately, derived classes still must have their own constructors. So, what you've got to do is create

Monster2 constructors to match each constructor in the Monster class. In the Monster2 constructors, you call the Monster constructors, using the super keyword as follows:



```

import java.awt.*;

public class Monster2 extends Monster {

    /*create constructors for the subclass from the
    superclass Monster constructors*/

    ➔ public Monster2() {
        super();
    }

    ➔ public Monster2(int desiredXCoord, int desiredYCoord,
        int desiredSize) {
        super(desiredXCoord, desiredYCoord, desiredSize);
    }
}

```

As an aside, here are my Monster constructors that are called by super:

```

//Constructor Methods

Monster(int desiredXCoord, int desiredYCoord, int desiredSize)
{
    x = desiredXCoord;
    y = desiredYCoord;
    size = desiredSize;
    eyeColor = Color.red; //I decided not to allow the user to
    bodyColor = Color.black;
}

//Overloaded constructor, default constructor
Monster() {
    x = 30;
    y = 200;
    size = 50;
    eyeColor = Color.red;
    bodyColor = Color.black;
}

```

- Now you can add new methods or override existing methods to train Monster2 to do additional things. Here's one I added:

```

public void changeColor(Color userEye, Color userBody){
    //set the instance variables of the super class Monster
    eyeColor = userEye;
    bodyColor = userBody;
}

```

C. CHANGE THE MONSTERCONTROLLER CLASS TO WORK WITH THE NEW MONSTER2 CHILD CLASS

Revise the monster controller so it works with your new derived Monster2 class instead of with the borrowed base Monster class. First, change your declaration so it refers to Monster2:

```
private Monster2 fred;
```

Then, change the line where you instantiate your monster to use Monster2:

```
fred = new Monster2(100,150,100);
```

In the end, you will have 3 files:

- The borrowed, base Monster class
- Your derived Monster2 class that adds new methods to the base Monster class
- A revised MonsterController class.

Special rules:

1. You are NOT allowed to change the other person's Monster class code in any way, with one exception: You can change any or all of the instance variables from private to protected.
2. To save time, you may use the methods you created for your own Monster class, as long as they differ in some way from what the other person did.

Challenge: add a new static method that counts the number of monsters created. To do this, add a counter variable to Monster2. Revise Monster2's constructors so that, after calling super(), they add 1 to the counter. Finally, create a static method which returns the number of monsters created. Add a button to your monster controller to test this method.

Hint: There is a beginning solution on Blackboard under Answer Keys