Support Code

What is it?

Programs consist of many class files. You make some; we make some. The project handout refers to the classes you edit as “skeleton” classes, and it refers to the classes that we edit as “support” classes. Simple as that. In the eyes of the computer, they’re all just class files.

Why do we need it?

Here’s a cool metaphor: Riding a bike is hard to get the hang of at first, as is coding. Training wheels enable you to ride your bike without having to worry about anything but pedaling. Pretty cool, right?! Support code is like training wheels, enabling you to code without having to deal with the nitty-gritty (ex: Graphics, Keyboard and Mouse interaction, Layout, etc.).
Thanks to support code, you can focus on learning foundation concepts such as parameters, inheritance, polymorphism etc. We’ll focus on the complicated stuff for now, so you can make cool things really fast. However, just like training wheels, as you start to gain balance, we’ll start to take away the support code. Trust us, you’ll be doing wheelies in no time!

What do I do with it?

Good question, ask the handout! The handout is your primary resource for the project. If you have questions about what’s in the support code, how to use the support code, and what to do with the support code, check the handout. It’s all in there! Since we don’t allow you to view the support code, we definitely make sure to provide a thorough explanation of everything you’ll need to get the project running. If you’ve read the handout and are still unsure about specific classes or methods, then come by and see a TA on hours!

Hold on… What’s up with the funky names?

All CS15 support code is part of a family, so each support class is named according to the same convention.

```
cs015.prj.<ProjectName>Support.<ClassName>
```

Try this!

What would the name be for a support class for a Thimble, if the project were called Monopoly?

```
cs015.prj.MonopolySupport.Thimble
```

You don’t have to worry about naming support classes, but it is important that you know how to instantiate and use support classes. On the following page, there’s an example for `cs015.prj.MonopolySupport.Thimble`.

Support Class

Name: `cs015.prj.MonopolySupport.Thimble`
Purpose: This class represents the thimble game piece.
Methods:

`cs015.prj.MonopolySupport.Thimble();`
Constructor. Creates a Thimble.

`void move();`
Makes the thimble move.

```
package Monopoly;

public class Gameboard {

	// Begin constructor for Gameboard
	// It gets passed a cs015.prj.MonopolySupport.Thimble
	public Gameboard(cs015.prj.MonopolySupport.Thimble thimble) {
		// Call the move method on the thimble passed in
		thimble.move();
	}

}
```

In this case, we know how to make the thimble move, because we read the handout’s support class section. We found the name `cs015.prj.MonopolySupport.Thimble`, read it’s purpose to understand what it is, and then checked out it’s methods so we know what it’s capable of. Any method on that list is fair game, as long as you call it on an instance of type `cs015.prj.MonopolySupport.Thimble`!