The Little Singleton

A small Socratic exercise in the style of The Little Lisper

How would you create a single object?

```
new MyObject();
```

And, what if another object wanted to create a MyObject? Could it call new on MyObject again?

Yes, of course.

So as long as we have a class, can we always instantiate it one or more times?

Yes. Well, only if it's a public class.

And if not?

Well, if it's not a public class, only classes in the same package can instantiate it. But they can still instantiate it more than once.

Hmm, interesting.

Did you know you could do this?

```
public MyClass {

   private MyClass() {}

}
```

No, I'd never thought of it, but I guess it makes sense because it is a legal definition.

What does it mean?

I suppose it is a class that can't be instantiated because it has a private constructor.

Well, is there ANY object that could use the private constructor?

Hmm, I think the code in MyClass is the only code that could call it. But that doesn't make much sense.

Why not ?

Because I'd have to have an instance of the class to call it, but I can't have an instance because no other class can instantiate it. It's a chicken-and-egg problem: I can use the constructor from an object of type MyClass, but I can never instantiate that object because no other object can use "new MyClass()".

Okay. It was just a thought.

What does this mean?

```
public MyClass {

   public static MyClass getInstance() {
   }
}
```

MyClass is a class with a static method. We can call the static method like this:

```
MyClass.getInstance();
```

Why did you use MyClass instead of some object name?

Well, getInstance() is a static method; in other words, it is a CLASS method. You need to use the class name to reference a static method.

Very interesting. What if we put things together?

Now can I instantiate a MyClass?

```
public MyClass {

    private MyClass() {}

    public static MyClass getInstance() {
        return new MyClass();
    }
}
```

Wow, you sure can.

So, now can you think of a second way to instantiate an object?

```
MyClass.getInstance();
```

Can you finish the code so that only ONE instance of MyClass is ever created?

Yes, I think so...

(You'll find the code on the next page.)