Design Method – Singleton

A coding story by Paul and Verity

The Singleton Design Pattern

NOTE – the ‘Many Universe’ theory is false is this example!

A Singleton Class is used when you only ever want one instance of itself to be created.

An example of this could be: a Class Universe.

Singleton Concepts

- Makes use of the Static property

Declaring a method or variable as static means it is of type ‘Class’.

These can be used without an instance of the class having been created.

I.e. like the ruby class method in pizza shop:

PizzaShop.show_all ()

This method is called on the class, not an object instance.

Singleton Concepts

- Has a Private Static Instance of itself

A Singleton Class has a Private Static instance of itself.

“BREATH…”

So a property a singleton class is a class level instance of itself that no-one but itself can access. (called ‘instance’ in this case)

```
public class Universe {

  private static Universe instance;

}
```

Singleton Concepts

- The constructor is Private

Unlike all other class constructors we have seen so far, which are public.

A Singleton Constructor is private, so can only be called by itself. Nothing else can create an instance of this class.

```
private Universe(String name) {

      this.name = name;

  }
```

Singleton Concepts

- It ALWAYS has a public method getInstance.
- This method will create the first and ONLY instance!

A Singleton Class ALWAYS has a public static method getInstance. As class level it is accessible before the object is created.

This method will call the private constructor, but will only do it once.

```
public static Universe getInstance(String
name) {

        if(instance == null) {

            instance = new Universe(name);

        }

        return instance;

    }
```

Singleton Concepts

- All further creations will just be a reference to the first.

Lets see the code and test to prove this and show the concepts we’ve just covered.

Our Example is Universe, but more generic uses could be:

- **Factory Design**
- **Pool Design**