**Discussion of Object Oriented Programming**

In this document, we’ll be discussing five different concept of Object Oriented Programming and how they apply to our website, Smart City 2.0

Inheritance

Inheritance is a concept describing when an object uses a similar implementation to another object, thereby inheriting that object’s properties. Essentially, creating a object built upon an already existing object.

We’ve applied this concept to our website since we’re using a number of objects that are very similar to each other. For example, an administrator account, while having its own purpose and set of features, is built on the regular account system we’ve put in place. Another example are the suggested locations that our users can submit to the website. Since they are kept separately in the database, they function as separate objects, however they both function in a very similar way.

Encapsulation and Abstraction

Abstraction is the concept of hiding the more complex details of an object in order to show the most essential features of it. Encapsulation is the concept of keeping an object’s inner workings hidden away while still providing an outwardly simple set of functions. As such, these concepts go hand-in-hand towards making code that is easily workable by other developers.

Because we are using the Django framework, a lot of this work is done for us, since we are able to use Python and Django’s many functions to accomplish what could take us much longer. For example, the Django database system takes database objects that we create and turns them into Python objects with attributes. These objects can then be easily used by us developers when we develop features for the website involving information from the database.

Polymorphism

Polymorphism is the concept of having different objects of one type fall under the same interface. Essentially, it allows different objects be usable in the same situations.

Our website employs polymorphism with regards to the many different types of locations that our users can browse. We have multiple functions that pull the information of these different objects and display it all in the same way on the website, but not to the same types of users. Also, the forms objects used in the website’s HTML files are polymorphic as well. The different forms represent different types of data, but since they all fall under the forms label, they can all be displayed in the same way on our website when pulled through the same functions.

Interfaces

Interfaces are the concept of requiring a specific type of object to be able to perform a certain function, or set of functions. In other words, the public methods and properties form a set of expectations that you can use to classify an object. An interface is often called a contract.

The forms class used in our HTML pages functions as an interface, since a forms-type of object must be called in order for its functions to be used to display an object’s information on the webpage. We use various forms in our website, such as on the registration page, which takes multiple different types of forms objects (CharField, ChoiceField, DateField etc.) and, being put through the interface, displays them on the website in the same fashion.