Making the Zombie Factory

This chapter is basically an introduction to Solidity. As most programming languages, it has structs, functions and various built-in data types (uint, uint8/16/32.., string). However all of these live inside a contract, as opposed to a class as you’re used to (such as Java). All of the variables inside a contract live inside the Ethereum blockchain, which can be thought of as a DB, more information has not been provided by the course on the term.

An important concept is that functions define very well before even reading their implementation whether they access or modify the data inside a contract. As data inside the blockchain can become expensive to maintain, there is strong incentive to be explicit with your data usage. Pure functions do not use the state of contracts at all, while view functions can access, but not modify the data inside a contract. However by not specifying either you get full read/write access to the state of a contract. It may seem similar to static/non-static methods in java, but this implementation goes a step further and defines a separate keyword for read-only access.

The default uint data type is surprisingly a 256 bytes data structure, as opposed to the nearly universal 32 bytes in most languages. So that is an important bit to keep in mind, especially If you plan to use a lot of numbers that likely won’t exceed the range of 32 bits numbers.

Arrays are not much different to other implementations, I like that we did away with the distinction between arrays and lists that we often find in C++ and Java. It makes me thing about how golang handles arrays, except there’s no complicated slicing on top of arrays, but an array can be declared to have a certain capacity or not at all, at which point it’s going to grow dynamically.

The naming conventions of the programming language are similar to those of python, except we still use camel case, but we start the name of private fields and functions with an underscore.

An interesting concept introduced is that of events. Events are being issued whenever you use the emit keyword, in turn other another application, such as a front-end app can listen to these events and execute anything based on the data being emitted. These seem to be a powerful tool to communicate with the blockchain, and I assume even the main way to do so. I’m looking forward to seeing more applications of the concept in future chapters.

Finally, I enjoyed studying the basics of solidity and it looks like it’s a language focused on simplicity, ease of reading, straight-forward concepts that are simple to pick up and use. I’d love to read more about contracts and events and see some solid applications of programs written in Solidity.