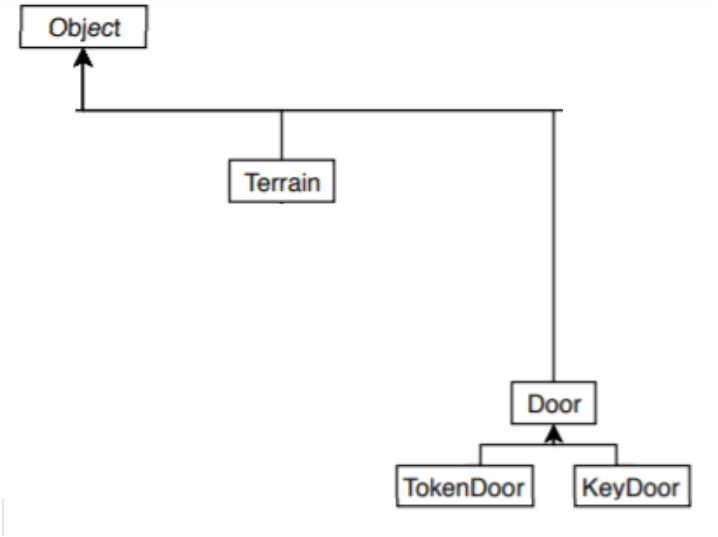
Hierarchy descriptions:



We have chosen to put these classes into an inheritance relationship as it allows us to use abstract methods and superclasses. We made this design choice as it removes repetition of methods and attributes, as they can be put in a more general superclass, rather than being repeated several times in smaller methods. Another advantage is that the code will be more readable, making it easier for group collaboration during the development cycle, thereby increasing the overall quality of the code. A good example of this in our classes is the *door* superclass, and the *TokenDoor* and *KeyDoor* subclasses. Without the door superclass, both of the subclasses would require a method *openDoor()*, which would do the same thing, in opening the door when the requirements of the door are met. Using the superclass *door* saves having to repeat this method in both classes, as it can instead just be written once inside the superclass.

In the section of our class hierarchy shown above, both *object* and *door* are subclasses of the class *object*. This class contains even more general information than *door*, such as information such as the objects coordinates, its texture, and a general draw method. *Terrain* and *Door* are not in subclasses of each other however, as they only need to share information stored in *object*, for example, both objects require *coordinates*, however *terrain* does not require a method *openDoor()*, as it is not a door, and therefore doesn't need to be opened.