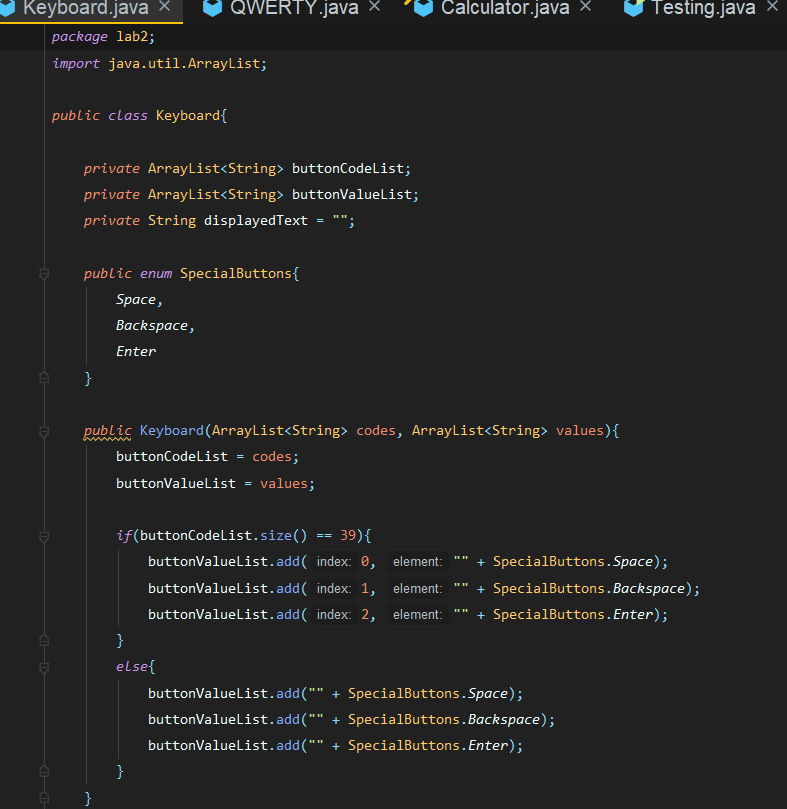
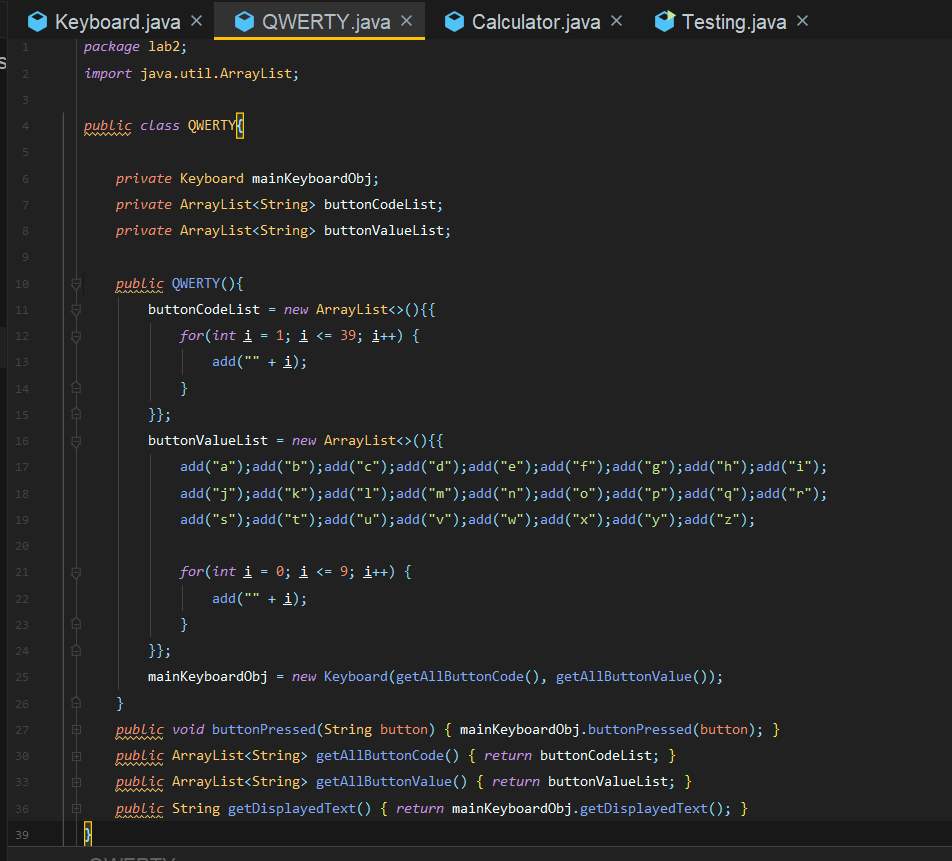
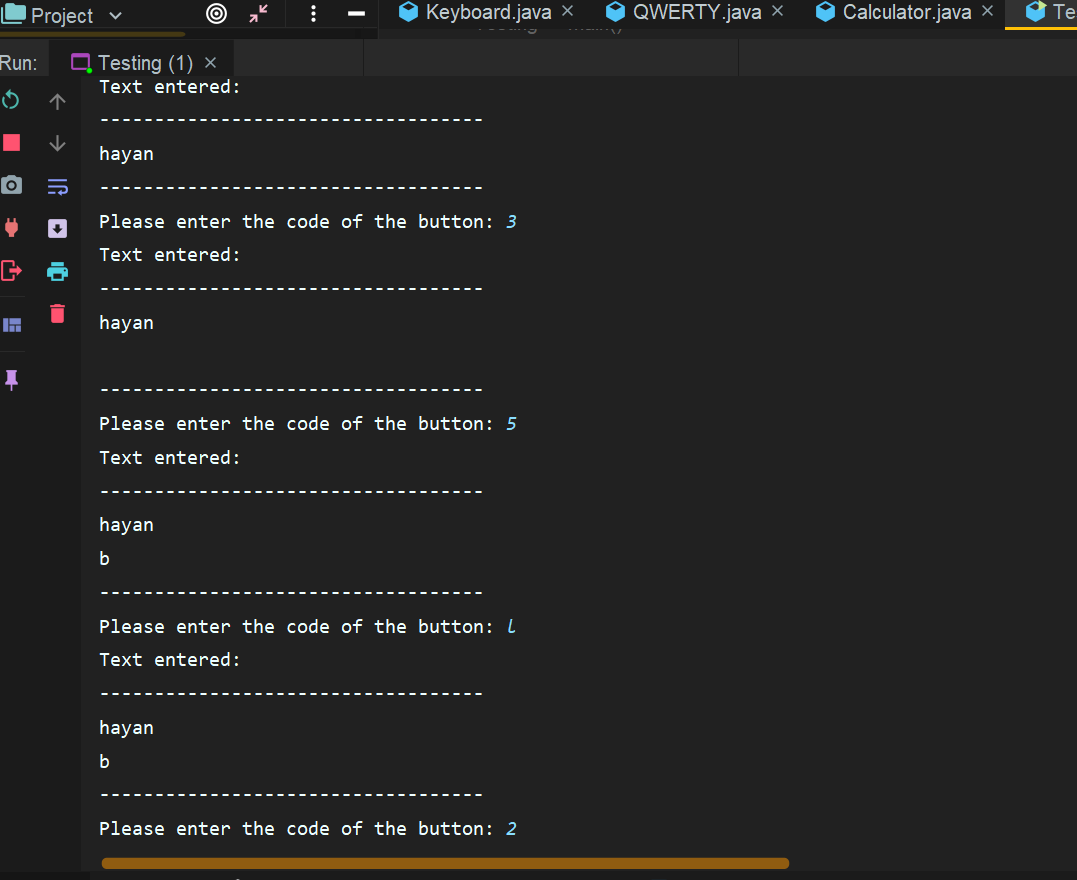
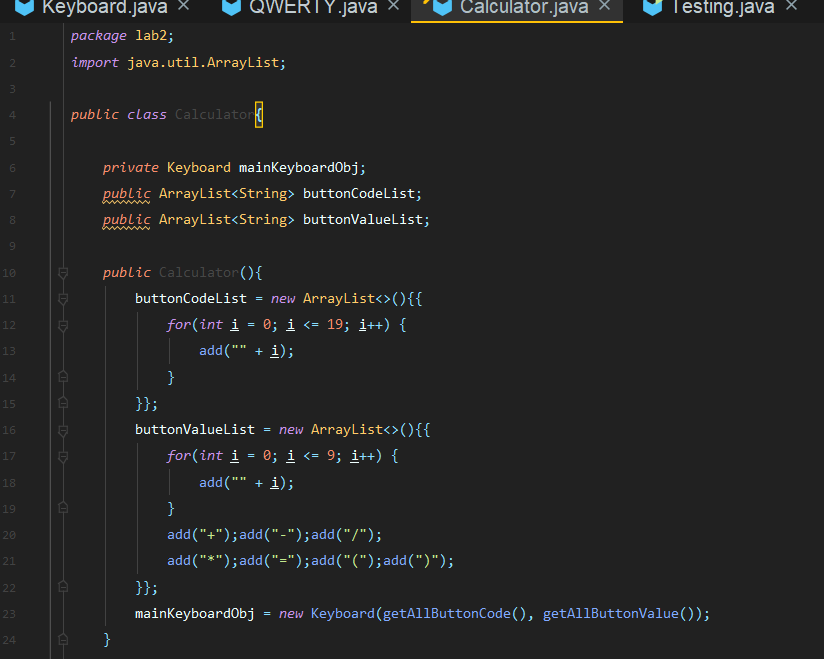
Part#1 keyboard simulator:Keyboard.java

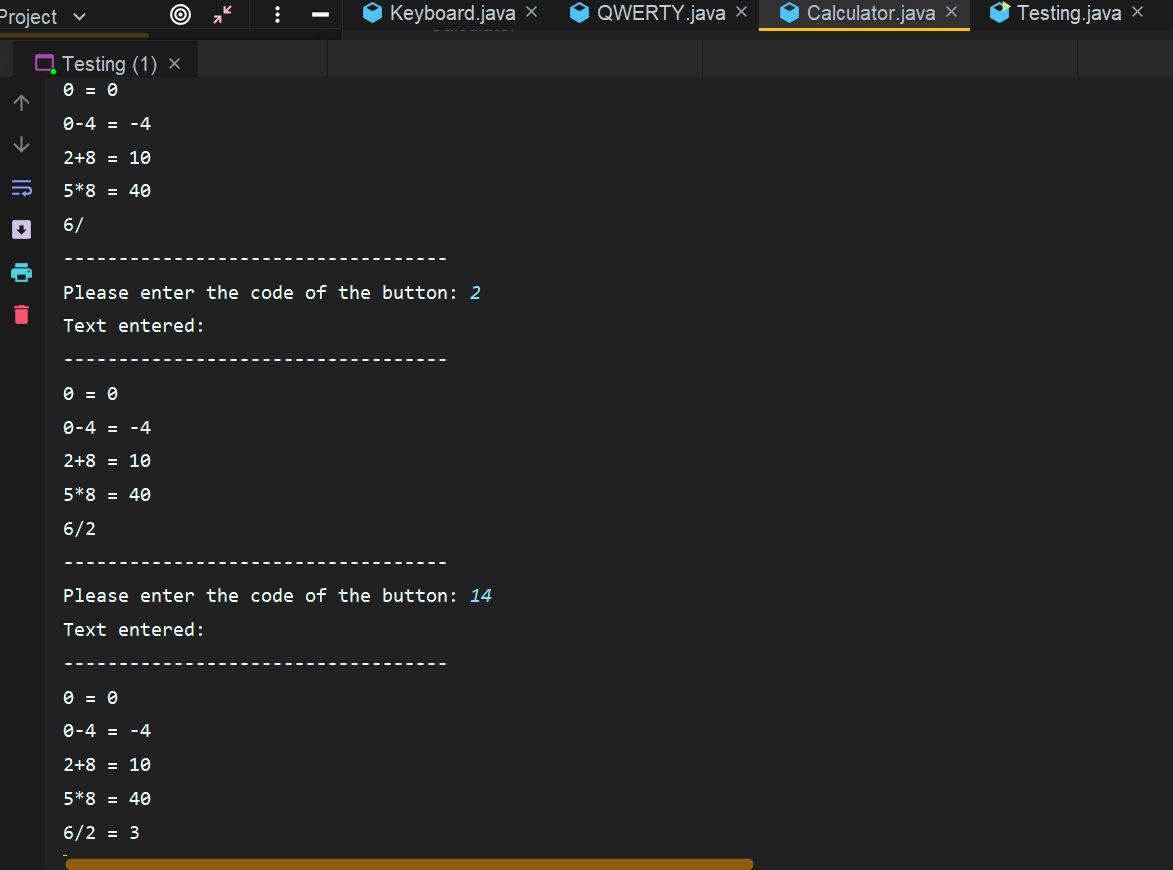


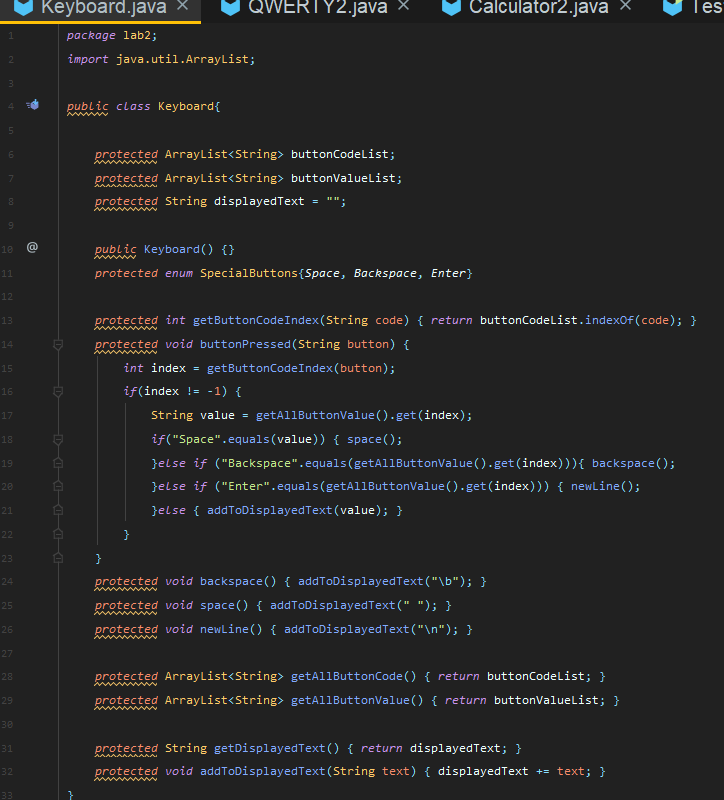


QWERTY.java

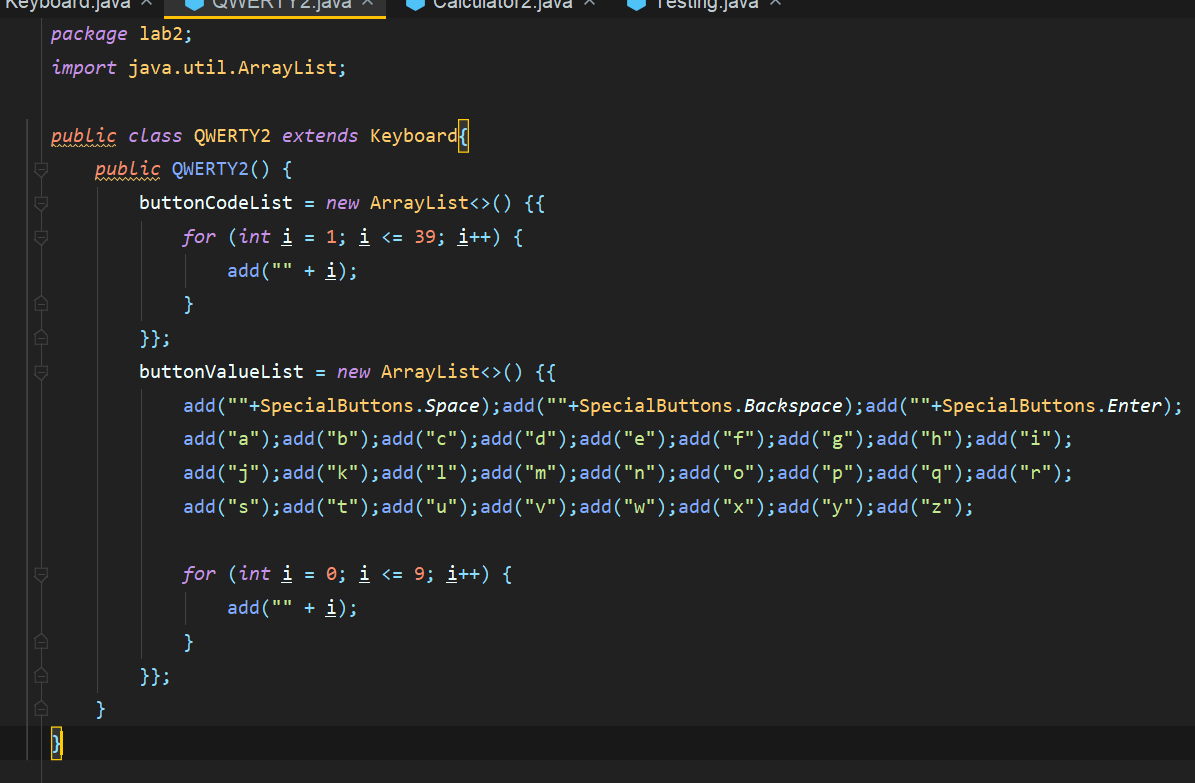


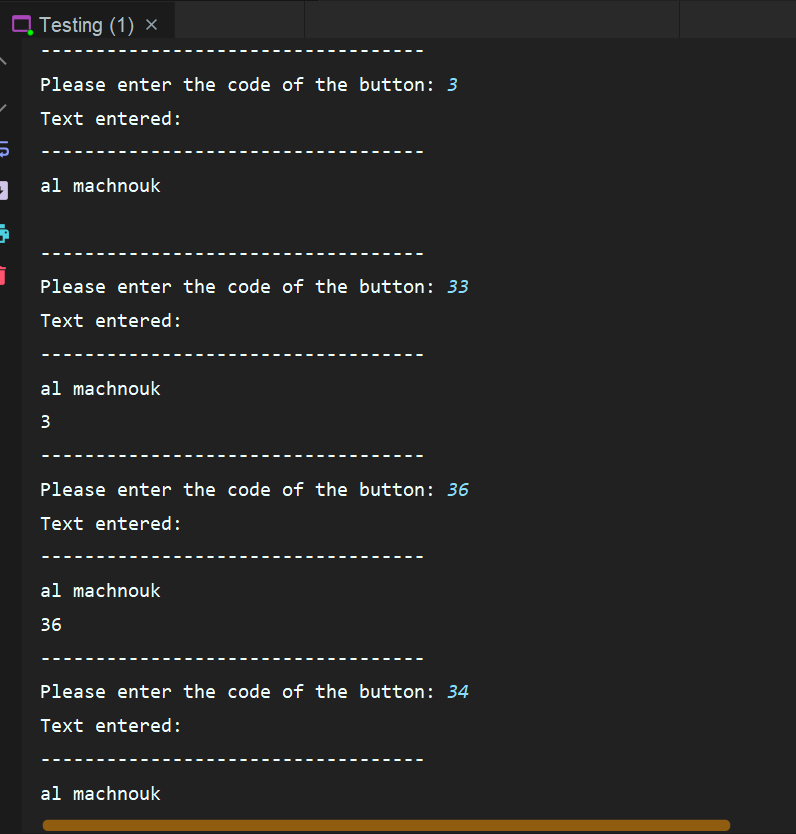
Calculator.java



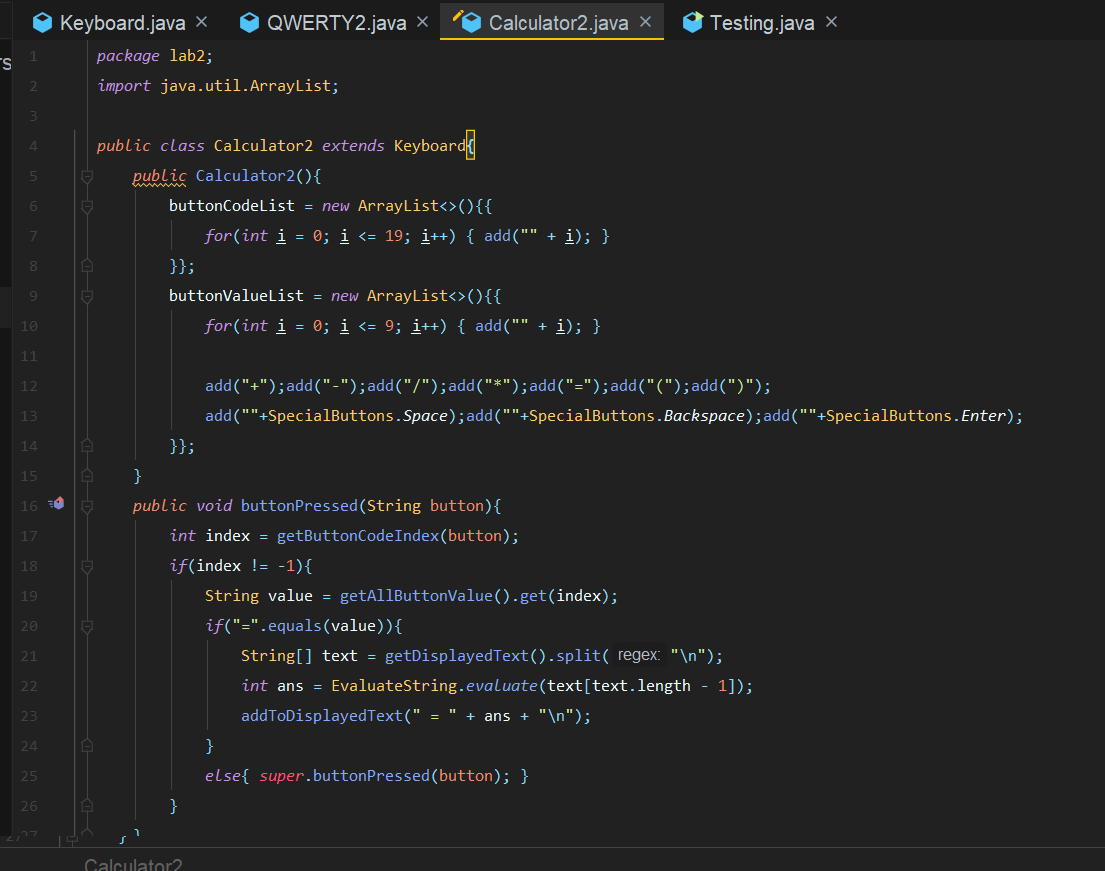
**Part#2 Using inheritance concept:  
Keyboard.java**

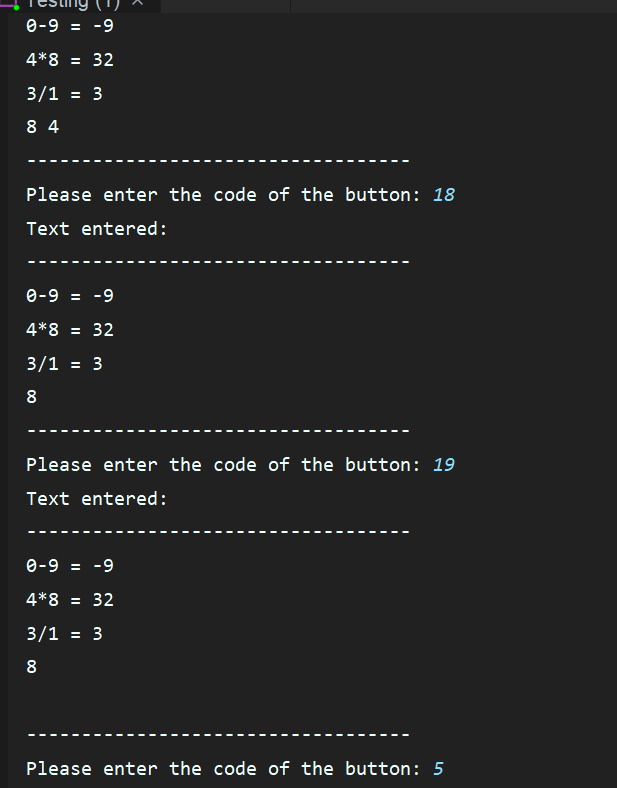
**QWERTY.java**





**Calculator.java**





**Part#3 What is the difference?**

The key distinction, is that inheritance allows the code to be more flexible. We interacted with the Keyboard class using the an instance object of itself in the first part, which had us write additional lines of code in both the QWERTY and Calculator classes, when we could just extend the methos directly from the super class via inheritance. The key rationale for using inheritance is to keep our code better organized and free from redundancy all of which, adhere to the OOP coding style's requirements.

When a programmer uses an external framework or API created by other developers for a specific purpose, that's an example of inheritance in action. However, the programmer is not permitted to change these classes and is only permitted to extend from them as needed. The code will be more stable as a result of this. Another example is that if you have a database model, you can design a main class that handles database-related tasks like adding tables, deleting tables, inserting new rows, and so forth. As a result, these methods are available in the super model class, and specific models can extend them.