Private Class Data design pattern :

**Intent**

* Control write access to class attributes
* Separate data from methods that use it
* Encapsulate class data initialization
* Providing new type of final - *final after constructor*
* A class may expose its attributes (class variables) to manipulation when manipulation is no longer desirable, e.g. after construction. Using the private class data design pattern prevents that undesirable manipulation.
* A class may have one-time mutable attributes that cannot be declared final. Using this design pattern allows one-time setting of those class attributes.
* The motivation for this design pattern comes from the design goal of protecting class state by minimizing the visibility of its attributes (data).

The Private Class Data Pattern addresses the issue a class may have, concerning around protecting object state, where final cannot be declared. The Private Class Data Patterns approach is to remove exposure of data by securing it within a class maintaining the data’s state. As a result separating data from that methods that use it and thus creating another layer of separation from the clients who have relationships with our applications entities. In a nutshell, the Private Class Data Pattern encapsulates class data initialization.

References:

<https://sourcemaking.com/design_patterns/private_class_data>

<https://medium.com/@dholnessii/structural-design-patterns-private-class-data-3b8e7f9881fc>