In the Account class, the attribute owner is declared public, so it is accessible from all classes regardless of package. In contrast, balance is marked private and can therefore only be accessed within the Account class itself. The pin attribute is protected, making it accessible both within the same package and by any subclass (even if that subclass is in a different package). Lastly, internalNote uses default (package-private) access, meaning it is only visible to other classes in the same package.

When designing such a class, it's often better to rely on getters and setters to control access rather than exposing fields directly. For example, using private along with a getBalance() method allows read-only access while preserving encapsulation. If a field needs to be visible to subclasses, using protected can work, but it’s less controlled than providing a private field with a protected getter. While protected allows direct access and easier subclassing, it weakens encapsulation. In contrast, using private with a getter ensures stronger encapsulation and lets you customize access logic, minimizing the risk of unintended modifications in subclasses.