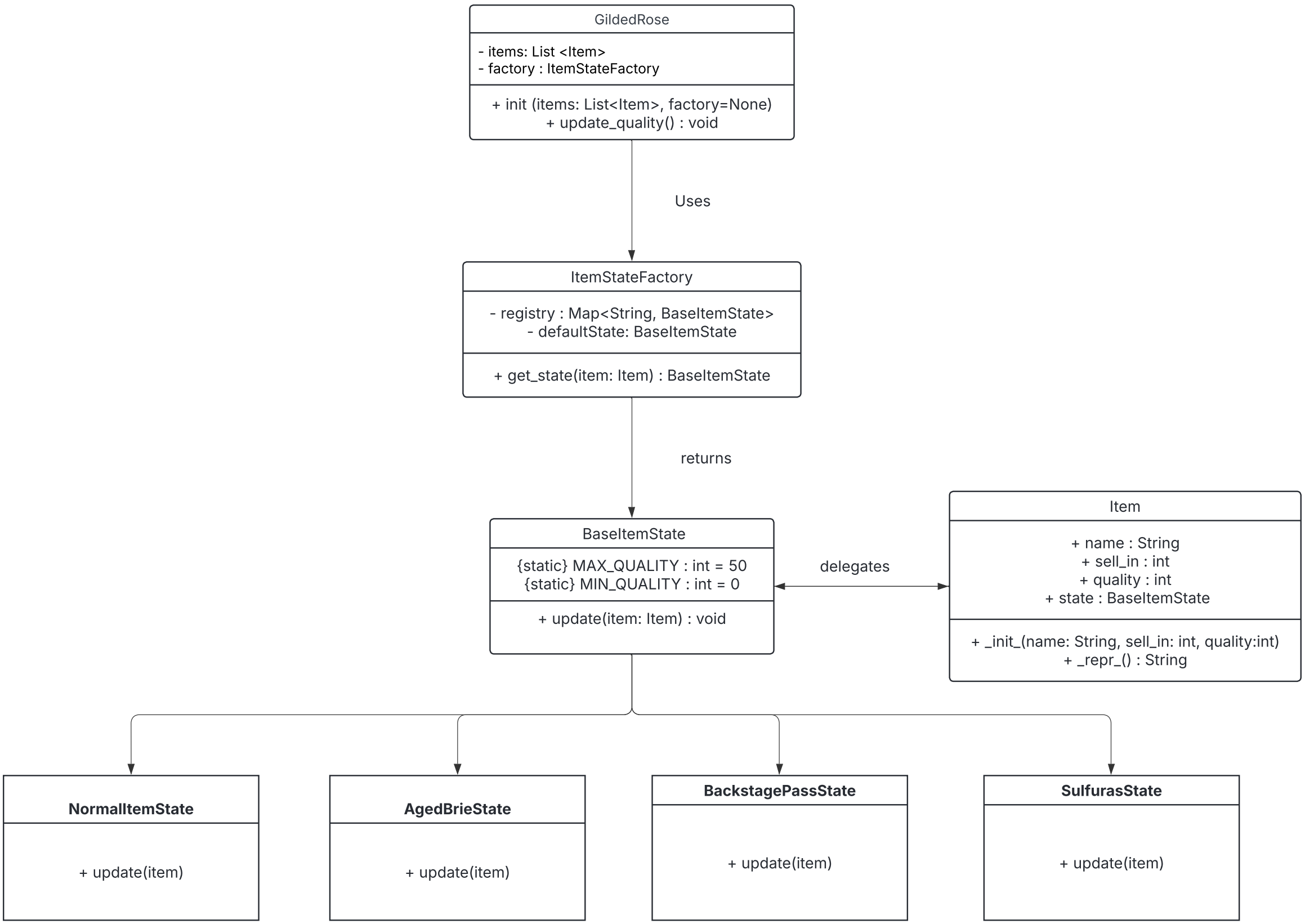
UML Class diagram for Gilded Rose Refactor using State Pattern



Description

- The main problem with the original code is that it has many nested if/else statements inside update_quality(), which makes the code hard to read,test and breaks when changes are made.
- To solve this, I used the State pattern, where each item type (Normal, Aged Brie, Backstage Pass, Sulfuras) has its own class that handles its update logic.
- The Item object simply passes the update work to its current state.
- This keeps update_quality() simple and makes each items behavior easier to understand and modify.
- A possible downside is that this approach creates more classes, and it still depends on matching item name strings unless those are later replaced with constants or enums.