**Major issues:** The original code squeezes everything in one big if-else method inside the GildedRose class. If there comes another special item, we need to add it somewhere inside the if-else statement. If that happens frequently, it's hard to maintain and easy to break existing logic.
**Design choices:** We have an abstract ItemWrapper class that wraps the original Item using composition (since we can't modify Item). Different kinds of items inherit from it and have their own way to update quality. Also, we have an Updateable interface to enforce the updateQuality() method. Now, if a new type of item comes in, just add another item class that inherits from ItemWrapper. The GildedRose class just loops through items and calls updateQuality() — it doesn't care what type they are.
**Potential drawbacks:**
-More classes to manage compared to one big method
-Need some way (like a factory) to create the right wrapper based on item name
-If item types share similar logic, might end up with some duplicated code
-initial setup is more complex even though future changes are easier