The class *InventoryItem* is responsible for updating the value of various items in an e-commerce system. Each item has a name, a value and a "sell in" property which denotes the number of days remaining before the product should be sold. Every day, the method *ageByOneDay*() is called on each item instance to update its value. The rules by which an item's value changes every day differ depending on the type of item:

* **Default items**: they decrease in value by 1 every day, but once the sell-in period has expired they start decreasing in value by 2 every day.
* **Vintage Wine**: this i*ncreases* in value by 2 each day.
* **World Cup Tickets**: this increases in value each day according to the following rules:
  + If there are more than 10 days left, increases by 1 each day
  + If there are 10-6 days left, increases by 2
  + If there are 5 days or fewer left, increases by 3
  + If there are 0 days left, the value of the item drops to 0
* **Perishable items**: these work like default items but they degrade twice as fast (i.e. 2 per day within sell period, 4 per day afterwards)
* **Gold bar**: this is a special item whose value is always 80 and never changes.

General rules for value are:

* Value can never go below 0 for any item
* Value can never go above 50 (other than the Gold Bar)

The exercise is:

1. Look at the provided implementation of *InventoryItem* and provide and brief code review i.e. describe briefly what's bad about it.
2. Refactor this implementation into objects implementing interfaces. Do not use implementation inheritance (i.e. a class extending another class). Interface inheritance is fine.

Don't worry about persisting the state of the objects (e.g. in a database).