PHP Developer Test

The objective of this test is the assessment of your capability to **solve the tasks** according to given **requirements** and to make your **own decisions** towards acceptable **time/quality ratio** of the results.

1. Process

After you've read ALL the tasks, **please email (Reply ALL to the email with the test) estimated amount of time required for completing it, in hours.**

When you are done with the tasks, email the resulting file (or archive with multiple files in zip format) by replying to all on the email with the test.

We will contact you via Gotomeeting (meeting invitation/link will be sent by email) or Skype shortly after the test results are received (on the same or next working day). Be prepared to answer the question about your implementation and implement corrections, if required.

2. Tasks

1. In the provided php source file (Source.php), implement the observer pattern for the class EntityManager. Use the standard PHP SplSubject interface. Then, whenever EntityManager::update() gets updated with new data, call notify() which will update observers.

2. Create two new classes that are observers of EntityManager, (they each implement the standard PHP SplObserver interface):

   A. One observer class that is updated with the result of every EntityManager::update() and logs this result to simple text file.

   B. One observer class that e-mails you a message every time a qoh for an inventory item dips below 5, with the sku of the item, and the current qoh.

3. Test your two new observer classes in conjunction with calls to itemsReceived() and itemsHaveShipped() (the driver function is already there to make some example calls!).

4. Use the included driver function to test not just your observers, but the system. I.e. observe outputs and compare with expectations. You can add whatever *echo* or *vardump* type debugging statements you need to help you with this (or any other part of the whole task!). If you find any errors in the existing code, make note of them, and fix them if you can.

Notes

One: If you want to send the changed data to the notify() function, (and then to the observer's update() functions), **you can add extra parameters to these functions, but you must default the extra parameters to null**, because the interfaces definitions of notify() and update() don't know about the extra parameters. So, your notify() function would look like this, (just an example!)

```
public function notify($changedData = null)
{
     etc.
}
```

Two: One clarification: under (2) (B), if you cannot get it to actually send an e-mail, that's fine. Just include the code (using the php mail() function) that would send e-mail.

Three: In the InventoryItem class, the data field "qoh" refers to Quantity On Hand.