# Coding Assignment - Messaging Provider

This assignment is intended to be big enough to allow you to demonstrate your coding abilities, but constrained enough to be achievable in a sensible time frame.

# Background

We have used a few different pub/sub messaging providers over the years. These employ different architectures and offer differing levels of functionality. However, all we really need is to be able to send messages in one place and receive them somewhere else, either in or out of process, using a simple topic-based approach.

We want a fairly simple wrapper around our underlying messaging provider which provides the above functionality but also lets us change underlying provider fairly easily. To allow unit testing of our code-base we also want a simple “in memory” implementation that fulfils the same basic requirements but doesn’t use said underlying provider, in case that needs external brokers or has licensing implications.

We have written a simple set of interfaces which define the required functionality:

* MessageSender and MessageReceiver allow messages to be sent and received.
* MessageFactory creates instances of the above.
* Message wraps the information being transmitted and allows housekeeping tasks if required by the underlying provider. Message payload is just a byte array.

# Assignment

Your assignment is to write an in-memory implementation of the above messaging interfaces. As mentioned, this is for use in unit tests of other pieces of our code-base which rely on messaging. The implementation should be “functionally correct”, in other words it should provide the kind of encapsulation and isolation of a “real” messaging system. However, it is *not* intended to be an enterprise solution and so you need not concern yourself with issues like message overflow/loss or performance optimisation.

The attached zip file contains:

* src/main - com.ph.coding.messaging
  + interfaces defining the messaging system
* test/java - com.ph.coding.messaging.memory
  + empty MemoryMessagingFactory for you to complete
  + basic MemoryMessagingTest to see whether your code works

Import the above into your IDE and get it compiling, you should only need to import JUnit-4.12. Then, open MemoryMessagingFactory and get coding. If you get senders/receivers working with time to spare then try and implement a better MemoryMessagingFactory.waitForMessages(…).

# Remember

* This is intended to take a few of hours, maybe two to four. Please don’t go beyond that - a good but half-finished solution is still a great topic for discussion during interview.
* We are trying to see whether you:
  + can get to grips with a problem in a sensible time frame and come up with a simple but effective design;
  + are aware of or can figure out the key high-level points which are important for messaging systems;
  + can write good quality code which works and is easy to understand;
  + know how to make this thread-safe, in the face of multiple senders/receivers.
* We are *not* trying to see whether you can write an enterprise messaging system, or whether you know about existing libraries which do this, such as JMS implementations.
* If you think further unit tests are necessary or relevant then please add them.
* Comment the code with any:
  + interesting or specific design choices;
  + assumptions you’ve made;
  + limitations inherent in your solution;
  + improvements which you think would be sensible.
* This is a key part of our interview and recruitment process, it is the main way in which we determine whether you know your stuff. Please do invest the time and effort in returning a good response. And please trust that:
  + we do not just run your solution against more stringent tests to give it a pass or fail;
  + we always invest the time to go through and study all your code in order to form an opinion of your development skills.
* Lastly, if you are unsure about anything then please get in touch so we can help or clarify. We are not trying to trip you up and don’t want to waste your time, we just want to see whether you’re a good developer.

Good luck!