# 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, these employ different architectures and offer differing levels of functionality. All we really want is to be able to send messages in one place and receive them somewhere else, using a simple topic-based approach.

To facilitate changes of messaging provider and allow deterministic behaviour in tests a simple set of interfaces exist which define the basic functionality required:

* MessageSender and MessageReceiver allow messages to be sent and received.
* MessageFactory provides instances of the above to the application.
* Message wraps up 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, intended for use in unit tests of other pieces of code which rely on said messaging. 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
  + implemented 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 couple of hours, please don’t go much beyond that - a good but half-finished solution is still a very good topic for discussion during interview.
* The code is for use in unit tests so doesn’t have to be particularly performant or efficient.
* Think about any differences, changes, additions etc. which you think might be useful on such a messaging system and its reference implementation - for discussion at interview.