1. We can listen to the events from Map and Multiple Data Structure using Entry Listener.
2. The events in Hazelcast have recently been remodelled and I find they are a little bit confusing at the moment.  
   The documentation, for much of the API, calls them entry listeners.  
   But as we see in places, the API expects an instance of a map listener which I believe confusion matters. At least when you’re first starting out.  
   **Documentation calls EntryListener but actually it is Map Listeners**.
3. There are a number of entry listeners available that you can implement but we will be considering the following ones.  
   Their names pretty easily tell you when they’re invoked from a Map or Multiple Data Structure.  
   Chart

   Description automatically generated with medium confidence
4. Let’s see an example of how we can use these.
5. In the last module, we created a queue on which we could queue up email objects.  
   In that example, we simply printed out the entries as they were right off the queue.

But imagine if that same process was used to read the email from the queue and then actually send that email to a recipient.   
Now consider an example:  
When we add a customer to a customers IMap, we may say that customer is registered with us and therefore we want to send them a welcome email.  
How can connect the Customers IMap that we already have and the email queue that we already have.  
The answer is of course the EntryListener

Diagram

Description automatically generated

1. To code:
2. Following is a new class implementing Entry Listeners and marked with @Service so that we can use Spring to inject HazelcastInstance using @Autowire.
3. Text

   Description automatically generated  
   See method init() is annotated with @PostConstruct and so will be called when this class instance is created and all dependencies are inject.  
   Here we’re getting the reference to the IMap “customers” and we’re calling addEntryListener(this, true) on this IMap.
   1. **first Parameter**: This is map listener, and this is where the API gets confusing.
      1. The EntryAddedListener, EntryUpdatedListener, EntryRemovedListener actually stay in the MapListenerInterface.  
         A screenshot of a computer

         Description automatically generated A screenshot of a computer

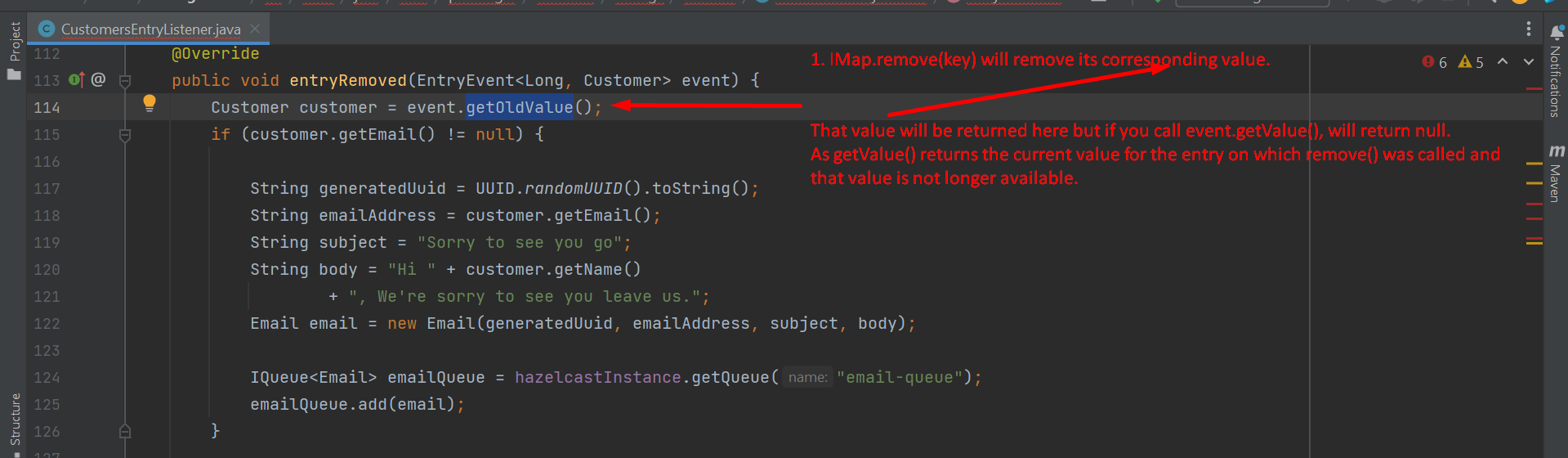
         Description automatically generated
   2. second Parameter:
      1. If true, EntryEvent will include the value from the entry on which operation (add, remove, update) is being called. In our case the value is Customer and that Customer will be available to the EntryListener.
      2. In our case, as we want Customer to get his email in order to send him welcome msg.
4. In the below snapshot, note: **EntryAddedListener.entryAdded(EntryEvent)**

Chart

Description automatically generated

1. Let’s write the Test Case to show this works.
2. A picture containing text

   Description automatically generated  
   **EntryUpdatedListener.entryUpdated(EntryEvent), EntryRemovedListener.entryRemoved(EntryEvent)** will be implemented in the similar way.
3. EntryEvent has two methods to retrieve the map value on it.
4. Table, timeline

   Description automatically generated
5. 
6. Make sure, EntryListeners are efficient. You should not use Entry Listener to run long-running processes based on entry events.