**Problem 1: Contradictions in the specification**

* To quote the specification:
  + 'charges customers for this service based on the weight, volume, origin, destination, and priority of the mail being sent.'
  + AND
  + 'The price customers are charged is based on priority, volume, weight, and destination'
* Here we can see that there's a contradiction in the specification. The first says origin is involved in the price calculation for customers, the second doesn’t say the origin matters for the price calculation for mail.
* How I handled this?
  + I’m going to presume that for the purposes of charging customers internationally, that origin is counted in the calculation but domestically it is not.
  + The reason for this is directly after 'The price customers are charged is based on priority, volume, weight, and destination', it also states that 'All New Zealand domestic destinations are considered the same for the purposes of charging the customer.'
  + This led me to believe that the second statement was trying to say that irrelevant of the origin in New Zealand, the price is the same for customers but internationally it isn’t.

**Problem 2: Ambiguity in the specification**

* To quote the specification:
  + 'Domestic air priority and domestic standard priority are the same.'
* What does this mean?
  + Same in terms of charging the customer?
* How I handled this?
  + I wrote a scenario to presume that they’re the same for charging the customer. This directly contradicts ‘The higher the priority, the more expensive it is for the customer’. Stupid…
  + It’s not my fault the specification isn’t concise enough. I’m just writing specification tests. Not my fault if they contradict each other.

**Problem 3: Non-deterministic nature of domain model**

* Trying to test mailing priorities was impossible as given X input, Y was never consistently determined. It was sometimes Y, sometimes Z or sometimes X. A TransportRoute type was never consistently AIR, SEA or LAND, it changed given the same inputs.
* The method inside KPSServer getTransportMap().calculateRoute(Mail) only ever returned a list of size 1, irrespective of how many possible routes there were. So the domain was flawed in the sense that one, it sometimes returned sea, sometimes air, but also it could only return one or the other.
* How I handled this?
  + Scenario outlines with the same values repeated multiple times to depict the non-deterministic nature of the domain model given some set of repeated inputs. I don’t know what else I could’ve done. My tests test the specification, it’s not my fault the domain model is rubbish and doesn’t consistently output the same answer. Our purpose in this assignment wasn’t to unit test and fix the bugs in the domain model, although maybe that would’ve been more fun as there’s a lot of them.
* Other instances of this:
* The specification never states how the KPS system should calculate the routes. Thus, we assume that the implementation is correct (located in the ‘calculateRoute’ method in TransportMap.java). However, the implementation doesn’t output consistent results and doesn’t work so the concept of ‘correct’ is somewhat of a ‘loose’ term to use. Hence, writing tests for inconsistent outputs is impossible.

I could’ve spent hours writing permutation tests to test how the domain model handles assigning routes etc, but the specification never mentioned how the system was meant to handle this, so I didn’t see the point. I stuck to trying to test tests (the best I could) in a BDD environment for the specification.

To test customer prices, I copied the <price> tags from logs.xml into data.xml so the customer price definitions were available when the KPSServer was started

I was never really sure if I should’ve been writing tests that passed with the data.xml or failed or a mixture.