**Test Driven Development Report**

We chose this test because the actual method based on it was one of the difficult ones to implement in the project and it has limitless possibilities for testing it. Our project method had to calculate the distance between the ZIP code entered by the user and the ZIP codes of the hospitals stored in the database using the longitudes and latitudes. So testing it, we decided first to measure a country (Netherlands) from top to bottom. In our example, the returned distance is in meters, and it’s a double. We know that a double is prone to rounding errors. Moreover, we know that the method used in this calculation – the haversine formula – is somewhat imprecise for our *planet*. It is enough for our application now, but perhaps, we’ll want to replace it with higher-precision calculation. We calculated the distance across the Netherlands. But of course, it is a nice idea to double-check and to triple-check; so checked the distance for a few more countries.  
 We have covered some base cases – the main functionality of the method seems to work. So we decided to go further and stretch our code to the limit. Our planet is rather a ball than a plane. This means that there is a place where latitude goes from 180 to -180, leaving a *space* where we should be careful. So we tested further: from one pole to another, max distance on the planet, really small distance, if both points are identical, etc..  
 Next we were thinking of negative cases: when the method is supposed to refuse to do its job; like a controlled failure. Latitude is supposed to be defined in exactly [-90, 90] degrees range. Longitude – in [-180, 180] range. This means that in case when we passed an invalid value, our method throws an exception. Same tests we did for latitude2, and for both longitude parameters.

Evidence of all tests that were created during the development of our project can be found here: <https://github.com/RossMitchell1999/Agile_Development_Project1/tree/master/Sprint%202/Test_Driven_Development>. Most of it was based on sorting and searching as we wanted to implement that as soon as possible, so many sorting algorithms had to be tested(BubbleSort, InsertionSort, MergeSort, QuickSort) in order to figure out the most essential, easy and practical sorting for the project.  
  
*P.S.* I apologise for the stuttering during the TDD video; I get nervous on camera.