# Unit Testing Design Report

SOFTWARE ENGINEERING | SOFT20111 | ASSIGNMENT 3

Hassaan Naveed | N0898071
NOTTINGHAM TRENT UNIVERSITY

## Introduction

The Route.findPosition() function takes the string soughtName as a parameter, and finds a RoutePoint bearing the specified name, returning its position. It throws an invalid argument exception if the argument is an empty string, and a domain error exception if the name is not found.

## Typical Inputs

For the findPosition() function, the main testing considerations would be to test test a selection of different RoutePoints that contain various data within their Position and Name. The content of the Position is not relevant to the function as the function is not concerned with the actual content, it should only be returning the Position object.

Additionally, non-standard charcters used within the Name should also be tested, for example, Names that contain spaces and symbols, as we should test whether the function is able to interpret such characters correctly.

## Error Cases

The significant error cases that will be tested shall be an empty string given to the function. This is an invalid input as the function is not designed to handle empty names, and therefor should throw an invalid argument error.

Furthermore, a domain error should also be thrown if the given name cannot be found within any of the Points of the Route object, as the function is not designed to handle names of Routes that do not exist.

## Edge and Boundary Cases

The main edge case to test for would be the first and last Points in the Route object, as we must ensure the behaviour at these extremes is the same as the behavious in the middle. Another edge case would be to test with a Route object that contains only a single point.

A boundary case to test would be a Route object that contains multiple points that share the same Name. For example, we should identify whether the function returns the Position value of the second point in the object if the first and second points both share a Name.

## Track Class

Finally, we should also test this function on the Track subclass in addition to the Route class, to ensure its functionality is the same between both. We must also consider the Track ‘granularity’ when testing on the Track subclass, as if some of the TrackPoints have been merged together, we must identify if the function is still correctly able to find the Name and Positional data of a single Point.