Exercise: `VectorCalculator` - a vector calculator (implemented using TDD)

In this exercise you will create a vector calculator (`VectorCalculator`) which operates on 2-dimensional Cartesian vectors. `VectorCalc` must be implemented using Test-Driven Development (TDD).

As you go along, try to be disciplined in your use of TDD: Create and maintain a feature list. When you get an idea for a new feature, add it to the feature list, but do not implement it right away. Instead, adhere to the discipline and continue with the feature you are currently developing, then address the new feature in time.

You will solve this exercise in pairs (2 persons – no more, no less) and use pair programming. One will be programming while the other maintains the feature list, plays the Devil's Advocate etc. Swap roles every 5-10 minutes.

Exercise 1:
Develop a class `Cartesian2DVector` which has X and Y coordinates.

Exercise 2:
Develop `VectorCalculator` with a *minium* of four of the below features using pair programming and TDD:

1. Add two vectors
2. Subtract two vectors
3. Scale a vector
4. Calculate the dot product of two vectors
5. The angle between two vectors
6. The magnitude of a vector
7. Transformation from Cartesian to polar coordinates

(If there is one or two of you who have forgotten how to take the dot product of two vectors, Uncle Google knows…)

Note on asserting equality of doubles:
Asserting that two doubles are equal is *not* done using the equality operator ('=='). Instead you assert that the difference between them is less than `Double.Epsilon`.