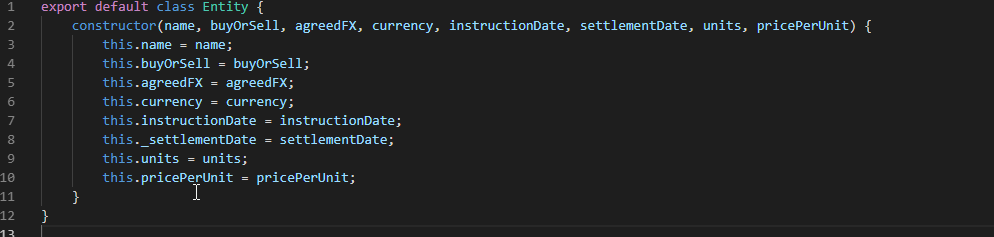
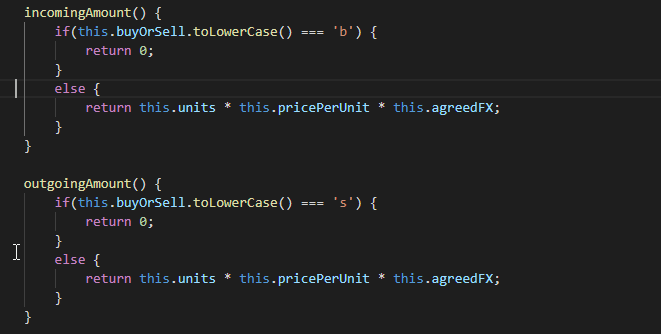
# Thought Log

Looking at the sample data presented in the problem, I think each of the entries can be expressed using a class. I created the following class with the same fields as the sample data:

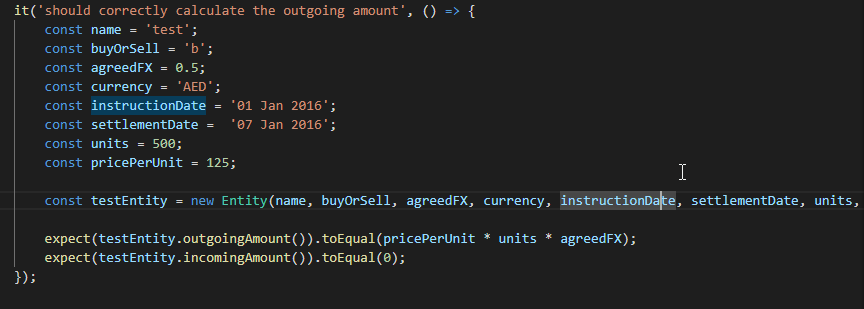


For this example, it is okay to name the class entity but if it were part of a larger codebase it would need to have a more specific name. Since we don’t know the context in which this entity resides I’ll leave it as entity.

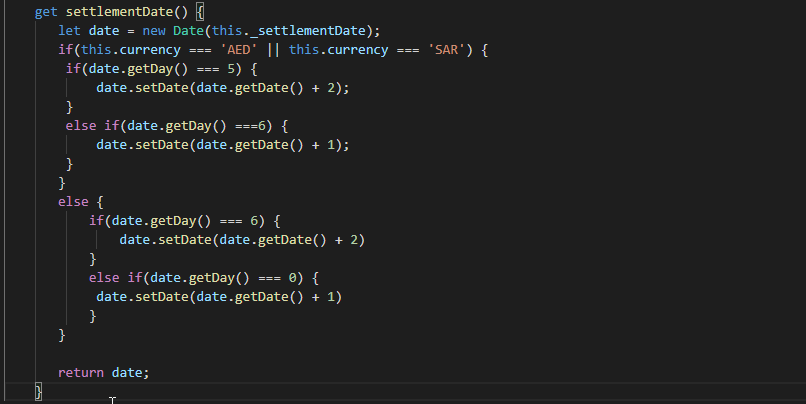
It seems there is a need to calculate the incoming amount and outgoing amount on a per entity basis. I created methods that check to see if the entity is a buy or sell and return the amounts as per the formula:



Next it would be good to write some tests to ensure our entity class does as expected. In each test case its important to check that the test fails before fixing it to pass.



Now for each entry the true settlement date may vary depending on whether the date falls on a weekend and the type of currency. One neat way to sort this would be to make use of es6 properties(getter). I’ll create get settlementDate() property which applies the rules of the exceptions.

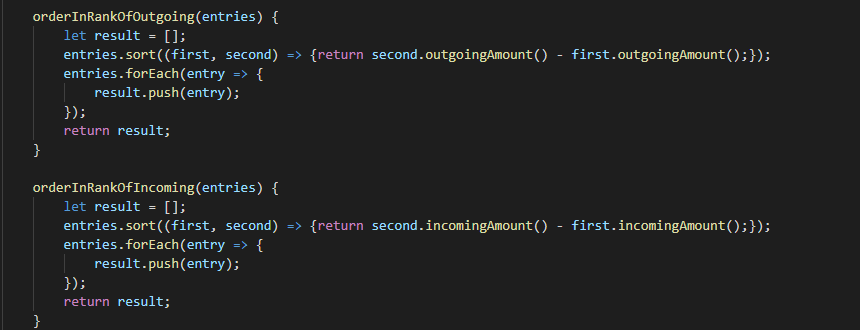


Note I am making use of the Javascript date class, like the exception currencies the date.getDay() method starts the week on Sunday and ends it on Saturday. I.e Sunday corresponds to 0 and Saturday corresponds to 6.

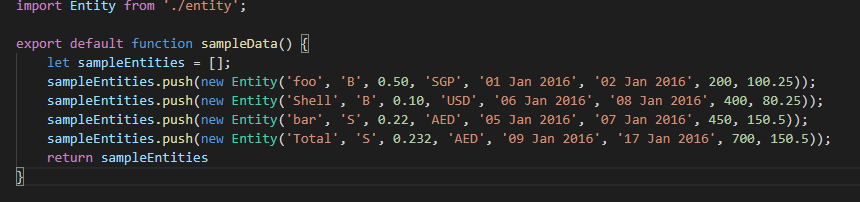
Next it’s important to write some tests that ensure the settlement date getter works correctly for all cases. These were added to entity.test.js.

I figured the next thing to do would be to write a class that can generate all the necessary information. This class will rank all the entries in outgoing and incoming amounts at get the total outgoing and total incoming over each day.

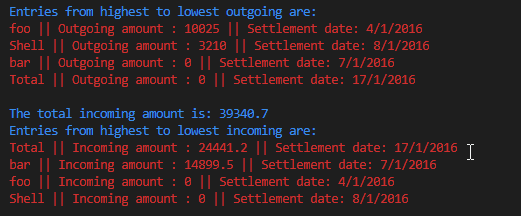
The methods used for ordering the list of entries could leverage the javascript array.sort method. Since arrays are passed by reference it would be best to return a new array. The sort method can take a predicate which can determine how its should sort each object in the array. Since we are ordering by outgoing and incoming rank I’ll use the methods present on the entity for the predicate.



I then made a simple file for the purpose of sample data. This sample data can be used to write the test for the ReportGenerator class.



Since the requirement was to print a report to the console I wrote a quick little node script that uses an npm package called chalk to print out a nice multicoloured report.



Just for fun I made a static page with source maps included to show the code working.

<http://amankatariajpmorgan.com.s3-website-eu-west-1.amazonaws.com/>