Documentation for optimizer

- Optimizer module for the first scenario

Properties

- _dataPoints -> will hold the hourly information loaded from the csv from the sdm
- _productionUnits -> hold the production units loaded
- Results -> public get, will hold the computed results

Public methods

- LoadData() -> Loads data from the SDM to the OPT which are needed to compute the optimization
- OptimizeScenario1() -> Once the data is loaded, it will optimize the activation of heat boilers and make the results accessible
- CalculateNetProductionCost() -> Once the optimizer was run, this method will calculate the Net production costs for heat only boilers
- LogResults() -> Will display the computed results
- LogDataPoints() -> Will display the dataPoints if they are loaded from the SDM
- LogProductionUnits() -> Will display the loaded production units which will be used for the optimization

Private methods

- SetOperationPoint() -> will be called to set the operation point on each boiler
- CalculateHeatUnitsRequired() -> is the main part of the optimizer as it organises the data and sets the appropriate computed values
- GetDataPoints() -> Loads data from the SDM
- GetProductionUnits() -> Loads production units (eventually from AM)

How the logic works Once the optimizer starts optimizing:

- each dataPoint loaded needs to be optimized
- production Units need to be stored sorted by efficiency

1. Determines how many boilers need to be activated
2. Based on that number the correct boilers are selected
3. Once the boilers are selected their operation point is set
4. All the information is added to an object ResultHolder which is added to a local list
5. After the foreach loop is completed the results are added to the public property
6. Now the netProductionCosts can be computed

First iteration of the OPT module

Everything under here is left as history, nothing applies anymore.

Scenario 1

The optimizer class has three private fields.

- `_productionUnits` holds the production units objects
- `_energyDataEntries` hold the data eventually provided by the SDM, which reads them from the CSV
- `_resultEntries` currently holds how many boilers need to be activated for any given time period

Optimize scenario 1

`OptimzeScenario1`does what it sounds like. It optimizes the first scenario, there are only heat boilers, no electricity has to be taken into consideration

1. After getting the production units and the time series data which holds how many Mwh are to be produced it orders the production units by their production cost
2. For each "hour" it calls `CalculateHeatingUnitsRequired` which calculates how many heating units have to be activated to satisfy the heating demand

BEWARE `CalculateHeatingUnitsRequired` returns how many heating units need to be activated. This number references the index of the productionUnit list, meaning that 0 means only the production unit with index 0. 2 would mean all the production units with index 0, 1, 2

Net Production cost

Since heat only boilers have no further income or expenses it useful to know what the cost of production is. Once the `OptimzeScenario1()` has run `NetProductionCost()` can be executed. Currently it just calculates it and displays it on screen. Once RDM is implemented it will correctly format the data and make it available to the RDM.

General expedients

- The class is not completed, this is only the initial implementation
- Most classes with which it works with, are going to be changed calling for major refactoring
- It tried to adhere to the SOLID principle as much as possible, refactoring should work ok