# Marking Sheet CODE - <https://github.com/ara-fb/Solution2.git>

Total Marks Claimed = 24

## Identifying Bad Smells

MARKS CLAIMED = 12

### Name: Long Method Marks claimed = 4

Location: Solution2\DataInterpreter\di.py –DataInterpreter - \_\_validate() between lines 62 and 90

Reasons Long:

1. More than 10 lines of code
2. Doing more than one function—washing all fields/validating all fields
3. Difficult to unit test (Test exactly ONE thing and one thing ONLY)

Strategy/ approach: <e.g., extract method>

1. Replace method with method object (Create a “Validator class)
2. In the DataInterpreter class, replace code with a call to the method object
3. Extract method (split method into smaller methods
4. Decompose conditional instead of one long if condition and condition and condition and

### Name: Primitive Obsession Marks claimed = 4

Location: Solution2\DataInterpreter\di.py –DataInterpreter - between lines 12 and 18, 21

Reasons:

1. Record is stored as an array of primitives
2. Even though each record has a fixed number of items and each item has a rule
3. Currently work around this using a couple of static arrays to get indexes by name

Strategy/ approach:

1. Replace Array with Object
   1. Record object stores values for id, gender, age, sales, bmi, income
   2. Record Class also has the rules for each property, that can be accessed publicly
2. Replace all the code that uses static arrays to get values with accessor methods of the record object.

### Name: Long Method Marks claimed = 4

Location: Solution2\DataInterpreter\di.py –DataInterpreter - \_\_add\_data() between lines 64 and 88

Reasons:

1. More than 10 lines
2. Doing a lot of different jobs – e.g. – adding data, generating invalid data message
3. Difficult to read/ understand

Strategy/ approach:

1. Extract Method
2. Replace inline code with call to method.

## Testing

MARKS CLAIMED = 12

* To develop a set of tests for the methods encompassed by the bad smells you previously identified (3 \* N marks)

Claimed = 3 \*3

* Please also use coverage package to generate reports in order to show your code coverage >= 80%. And all tests should be able to be run together by running a single .py file (N marks)

Claimed = 3

* Sourcecode tested = Solution2/DataInterpreter/di
* Look at TestCoverageResults.jpg on GitHub for a screenshot - 90% branch coverage

## Refactoring (4 \* N marks)

MARKS CLAIMED = 12

* Version control via a remote repository and testing

<https://github.com/ara-fb/Solution2.git>

Claimed = 3

* Modification and PEP8 validation

Claimed = 6

* Effectively evaluations -

**look at RefactoringEvaluations.docx on github repo**

Claimed = 3