Lab 1B

For the following function, model the input domain using both the **interface**-**based** and the **functionality**-**based** approach.

/\*\*

\* Given two maps with string keys and integer values, return a new map containing only the keys that exist in both maps.

\* For each such key, the value should be the absolute difference of the two values from the input maps.

\*

\* Example:

\* map1 = {"A": 5, "B": 10, "C": 3}

\* map2 = {"B": 7, "C": 8, "D": 12}

\* Output: {"B": 3, "C": 5}

\*

\* If there are no common keys, return an empty map.

\*/

**public static Map<String, Integer> computeValueDifferences(Map<String, Integer> map1, Map<String, Integer> map2);**

Once you have defined the characteristics, you need to divide them into blocks, and answer the following questions:

а) Is the partitioning of the input parameters such that it ensures that the partitions are disjoint? Why? If not, alter the partitioning to ensure this property is satisfied.

б) Is the partitioning of the input parameters such that it ensures that the partitions cover the entire domain? Why? If not, alter the partitioning to ensure this property is satisfied.

в) Choose a base test and list all the necessary tests to satisfy the **Base Choice Coverage** (**BCC**) criterion. How many tests did you get?

г) Write JUnit tests using the BCC criteria for ISP coverage. Try to use AI tools to help you!

**Interface-based approach**

|  |  |
| --- | --- |
| Characteristic | Partition |
| C1 = map1 is null | {True, False} |
| C2 = map2 is null | {True, False} |
| C3 = map1 is empty | {True, False} |
| C4 = map2 is empty | {True, False} |

**Functionality-based approach**

|  |  |
| --- | --- |
| Characteristic | Partition |
| C5 = Number of common keys | {A (0), B (1), C (>1)} |

\*The partitions are disjoint and they cover the entire domain.

**Base Choice Coverage**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C1 | C2 | C3 | C4 | C5 |  |
| F | F | F | F | C |  |
| T | F | F | F | C | *Excluded* |
| F | T | F | F | C | *Excluded* |
| F | F | T | F | C → A | *Modified* |
| F | F | F | T | C → A | *Modified* |
| F | F | F | F | A |  |
| F | F | F | F | B |  |

**Test Cases**

1. FFFFC
2. FFTFC → FFTFA
3. FFFTC → FFFTA
4. FFFFA
5. FFFFB