**Test Cases: Unique Minimum Number Zip Code Range Algorithm**

Please go to “nisum-coding-test\src\test\java\com\amolzore\nisum\codingtest\service\ZipCodeRangeServiceTest.java”

**Code Test Cases**

**Test Case 1:-**

**Description**: No over lapping is there.

Metod:- testGetUinqueMinimumNumberOfRangesWithNoOverLapping()

**Test Case 2:-**

**Description**: Over lapping is there.

Metod:- testGetUinqueMinimumNumberOfRangesWithOverLapping()

**Test Case 3:-**

**Description**: Same zip code ranges are provided.

Metod:- testGetUinqueMinimumNumberOfRangesFromSameRanges()

**Test Case 4:-**

**Description**: Zip code ranges having one unique zip code range after consolidation.

Metod:- testgetUinqueMinimumNumberOfRangesForOneUinqueRange()

**Test Case 5:-**

**Description**: Arbitrary Zip code ranges.

Metod:- testGetUinqueMinimumNumberOfRangesWithArbitraryRanges()

**Test Case 6:-**

**Description**: Zip code ranges with mix of special characters.

Metod:- testGetUinqueMinimumNumberOfRangesWithSpecialCharInZipCodeRanges()

**Test Case 7:-**

**Description**: Zip code ranges with mix of incorrect length zip codes.

Metod:- testGetUinqueMinimumNumberOfRangesWithIncorrectLengthOfZipCodeRanges()

**Test Case 8:-**

**Description**: Zip code ranges with mix of incorrect length zip codes and with special chars.

Metod:- testGetUinqueMinimumNumberOfRangesWithMixIncorrectZipCodes()

**Performance Test Cases**

**Test Case 1:-**

**Description**: to measure the performance one hundred thousand (100000) zip code ranges used.

Method: - testGetUinqueMinimumNumberOfRangesWithOneHundredThousandZipCodeRange()

**Test Case 2:-**

**Description**: to measure the performance one hundred thousand (100000) correct zip code ranges used.

Method: - testGetUinqueMinimumNumberOfRangesWithOneHundredThousandCorrectPatternZipCodeRange()

**Test Case 3:-**

**Description**: to measure the performance one Million (1000000) zip code ranges used.

Method: - testGetUinqueMinimumNumberOfRangesWithOneMillionZipCodeRange()