# ISP: Sample Problem

Using the code below, complete the following tables. Your ISP analysis should be based on not needing to fully understand the inner workings of the code beyond what is described in the method comments. Be sure to clearly label your base choice.

```java
/**
 * This class holds an array of Strings and allows additional
 * arrays of strings to be added to it.
 */
public class WeirdArrayWidget {
  private String[] currentArray;

  /**
   * Creates a new instance of a WeirdArrayWidget initializing the current
   * array to a new String array of the specified length.  An initial value
   * that is less than 0 initializes the currentArray to an array of length 0
   */
  public WeirdArrayWidget(int initialSize){
    // Some code would eventually go here....
  }

  /**
   * Takes the passed array of strings, if the length of the passed elements is
   * equal to or longer than the current array it should be placed before the
   * elements in the currentArray, otherwise the elements should be placed after
   * the elements in the curent array.  Thus, the new length of the current array
   * should be equivelent to the sum of the lengths of the current array and the
   * passed array (null is same as a length of 0).
   */
  public void addMoreStrings(String[] moreStrings){
    // Some code would eventually go here....
  }

  /**
   * Prints out the contents of the currentArray in order in the form ["a","b",null],
   * if the current array has a length of zero or all elements in the array are null
   * 'The widget is empty' is returned.
   */
  public String toString(){
    // Some code would eventually go here...
  }
}
```

| Table A: Identify input space and determine characteristics | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Method** | **Params** | **Returns** | **Values** | **Exception** | **Char ID** | **Characteristic** | **Covered by** |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

| Table B: Design Partitioning | | | | | |
|---|---|---|---|---|---|
| Char ID | Characteristic | f1: | f2: | f3: | Partition |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| Table C: Define test requirements for BCC (make sure to indicate your base case) | | | | | |
|---|---|---|---|---|---|
| **Method** | **Characteristics** | **Test Requirements** | **Infeasible TRs** | **Revised TRs** | **# TRs** |
| | | | | | |
| | | | | | |
| | | | | | |

| Test Case design | | | |
|---|---|---|---|
| **addMoreStrings()** | **Case** | **Set up data** | **Steps** |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Solution:

**Just one possible solution, completely depends on your characteristic choices.**

**Table A: Determine characteristics**

| Method | Params | Returns | Values | Exception | Char ID | Characteristic | Covered by |
|---|---|---|---|---|---|---|---|
| WeirdArrayWidget() | initialSize | WeirdArrayWidget | int | | C1 | Initial size non-negative | |
| | | | | | C2 | Length > 0 | |
| addMoreStrings() | moreStrings | void | String[], null | | C3 | Passed array length compared to state | |
| | | | | | C4 | Passed array non-null | |
| | state - currentArray | | length | | | | C2 |
| toString() | state - currentArray | String | "The widget is empty", array representation | | C5 | currentArray only contains non-null values | |
| | | | | | C6 | currentArray only contains null values | |
| | | | | | | | C2 |

**Table B: Design Partitioning**

| Char ID | Characteristic | f1: WeirdArrayWidget() | f2: addMoreStrings() | f3: toString() | Partition |
|---|---|---|---|---|---|
| C1 | Initial size non-negative | X | | | true/false |
| C2 | Length > 0 | X | X | X | true/false |
| C3 | Passed array length compared to state | | X | | less/equal/greater |
| C4 | Passed array non-null | | X | | true/false |
| C5 | currentArray only contains non-null values | | | X | true/false |
| C6 | currentArray only contains null values | | | X | true/false |

**Table C: Define test requirements**

| Method | Characteristics | Test Requirements | Infeasible TRs | Revised TRs | # TRs |
|---|---|---|---|---|---|
| WeirdArrayWidget() | C1C2 | **TT**,FT,TF | FT | FT->FF | 3 |
| addMoreStrings() | C2C3C4 | **TGreaterT**,FGreaterT,TEqualT,TLessT,TGreaterF | TGreaterF | TGreaterF->TLessF | 5 |
| toString() | C2C5C6 | **TFF**,FFF,TTF,TFT | | | 4 |

| Test Case design | | | |
|---|---|---|---|
| Method | | | |
| **WeirdArrayWidget()** | **Case** | **Set up data** | **Steps** |
| criteria: C1C2 | **TT (base)** | w = new WeirdArrayWidget(1) | assertEqual("The widget is empty",w.toString()) |
| | FF | w = new WeirdArrayWidget(-1) | assertEqual("The widget is empty",w.toString()) |
| | TF | w = new WeirdArrayWidget(0) | assertEqual("The widget is empty",w.toString()) |
| | | | |
| **addMoreStrings()** | **Case** | | |
| criteria: C2C3C4 | **TGreaterT (base)** | base, w.addMoreStrings(["bear",null,"dog"]) | assertEqual("["bear",null,"dog","cat",null]", w.toString()) |
| | FGreaterT | w = new WAW(0), w.addMoreStrings([null,"dog"]) | assertEqual("[null,"dog"]", w.toString()) |
| | TEqualT | base, w.addMoreStrings([null,"dog"]) | assertEqual("[null,"dog","cat",null]", w.toString()) |
| | TLessT | base, w.addMoreStrings(["dog"]) | assertEqual("["cat",null,"dog"]", w.toString()) |
| | TLessF | base, w.addMoreStrings(null) | assertEqual("["cat",null]",w.toString()) |
| | | | |
| **toString()** | **Case** | | |
| criteria: C2C5C6 | **TFF (base)** | base | assertEqual("["cat",null]", w.toString()) |
| | FFF | w = new WAW(0) | assertEqual("The widget is empty",w.toString()) |
| | TTF | w = new WAW(0), w.add(["dog"]) | assertEqual("["dog"]", w.toString()) |
| | TFT | w = new WAW(2) | assertEqual("The widget is empty",w.toString()) |
| | | | |
| | | | |
| **Base data** | **w = new WeirdArrayWidget(1), w.addMoreStrings(["cat"]) ["cat",null]** | | |