# **MedianList: Assert, Try-Catch, Testing**

Suppose you are creating a MedianList data structure. Integers can be added to the data structure in any order but must be removed median first (if there is an even count of numbers then the number closest to and greater than the median is returned).

In other words, if the following numbers are added: 2,7,4,2,2,6,1

Then they would be removed in this order

**Removed** **Input Order** **Sorted Order**

2,7,4,2,2,6,1 1,2,2,**2**,4,6,7

2 7,4,2,2,6,1 1,2,2,**4**,6,7

4 7,2,2,6,1 1,2,**2**,6,7

2 7,2,6,1 1,2,**6**,7

6 7,2,1 1,**2**,7

2 7,1 1,**7**

7 1 **1**

1 [ ] [ ]

Printing the data structure must print out the numbers in the order in which they were added, so you can't sort the list.

Create two classes with the following methods and attributes:

|  |
| --- |
| MedianListTest |
|  |
| + main(String[] args) : void  + isMedian(int x, ArrayList<Integer> values) : boolean  + testCorrectMedian() : Boolean  + testCorrectString() : boolean |

|  |
| --- |
| MedianList |
| - values : ArrayList<Integer> |
| + <<constructor>> MedianList()  + push(int x) : void  + pop() : int //remove and return median value  + peek() : int //reveal the next value to be popped  + isEmpty() : boolean  + toString () : String |

Additionally, write the **pop** method to throw an exception if the user tries to pop an empty list.

Don’t be surprised when you get an UncaughtException error from the compiler. Once you throw the new Exception, you will need to declare that the method throws an Exception and you will need a try catch around any calls to pop.

In **push**, after a new value has been added, if the new value is smaller than the old median, assert that the new median is less than or equal to the old. If the new value is larger than the old median, assert that the new median is greater than or equal to the old. You will need at least one local variable for this.

Fill in the code for a test case in the **testCorrectMedian** method that makes sure the median revealed by peek is correct after a series of pushes and pops.

Fill in the code for a test case in the **testCorrectString** method that makes sure the String returned from toString is correct after a series of pushes and pops.