



Specifications: DropFilter

In addition to the obvious specifications illustrated in the UML class diagram, the DropFilter class must satisfy the following specifications.

1. public methods must not have any side effects. That is, they must not change the parameters that they are passed in any way (e.g., the List that is passed to the apply() method must not be changed in any way) and they must not change attributes that are not “owned” (i.e., attributes that are aliases) in any way.
2. You may assume that the apply() method is passed a List that does not contain any null elements.
3. The default constructor must construct a DropFilter object that drops the lowest and highest element.
4. The apply() method must construct a new List that is a subset of the List it is passed.
 - 4.1. If the apply() method is passed a List that has an inappropriate size then it must throw a SizeException.
 - 4.1.1. If the apply() method is passed a null List then it must throw a SizeException.
 - 4.1.2. If the apply() method is passed a List that contains fewer elements than are to be dropped then it must throw a SizeException.
 - 4.1.3. If the apply() method is passed a List that contains the same number of elements as are to be dropped then it must throw a SizeException.
 - 4.2. If the apply() method is passed a list that has an appropriate size then it must return a new List.
 - 4.2.1. The elements of the new List must be aliases for (**not** copies of) the Grade objects in the List it is passed.
 - 4.2.2. Because each Grade object in the List has a key that can be used to identify it, the new List need not be in the same order as the List it is passed.
 - 4.2.3. The elements in (and size) of the returned List must be based on the values of the parameters that were passed to the constructor when the object was constructed.
 - 4.2.3.1. If shouldDropLowest was true then it must drop exactly one of the elements with the lowest value in the original List.
 - 4.2.3.2. If shouldDropHighest was true then it must drop exactly one of the elements with the highest value in the original List.
 - 4.2.3.3. When dropping the highest and lowest, two elements must always be dropped, even if the highest and lowest have the same value.
 - 4.2.3.4. When determining the highest and/or lowest values it must account for missing (i.e., null) values as in the compareTo() method of the Grade class (i.e., missing values have smaller magnitude than non-missing values and one missing value has the same magnitude as another missing value).