Report: Random Queue

By Sergiy Isakov, Rasmus Lehmann & Andreas Tønder

Implementation of RandomQueue

RandomQueue is using a simple array ***items[],*** which is initialised as a new Object with size 1. Another field is ***size***, initialized as 0, which corresponds to number of items in *RandomQueue*. The constructor for *RandomQueue* is empty, and simply creates an instance with two fields: ***items[]*** and ***size***.

Method overview

isEmpty() returns a Boolean value of statement size==0;

size() returns actual value of size.

enqueue(Item item) first checks if there is enough space in the array, by comparing ***size*** and array length. If they are equal, then a new array is created, with double the size of the original array. The method then adds a new item to the array, and increments the ***size*** variable by 1.

sample() throws a *RuntimeException* if ***size==0***, or returns a random item from the array, using the *StdRandom.uniform(size)* method (I.e. a random integer between 0 and size).

deque() has a similar behaviour to *sample(),* but it also deletes the returned item from array. To do this, it swaps a randomly chosen item with the last item of the array, and then sets the last item to be null, thus removing it. After this, the size variable is decremented.

Implementation of RandomQueueIterator

RandomQueueIterator has one field; *cursor*, which is instantiated as 0.

The constructor shuffles the items in the array using *StdRandom.shuffle(),* to provide random elements sequence in the array.

Method overview

hasNext() returns a Boolean value from the statement *cursor != size*. I.e. if cursor reaches *size* value, there are no next items in array.

remove() has no implementation, and throws an *UnsupportedOperationException(),* as specified in the assignment parameters.

next() throws a *NoSuchElementException* if there are no next items. Else, it returns the *item* in *Items[]* corresponding to the *cursor*, and increments *cursor* by 1.