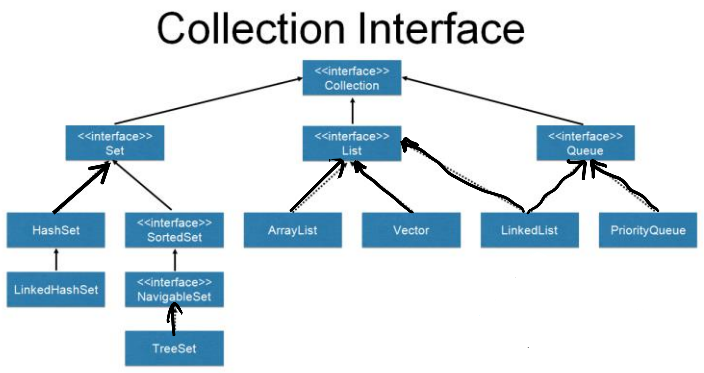
Gebze Technical University
Department of Computer Engineering
CSE 241/505
Object Oriented Programming
Fall 2020
Homework # 6
Generics and Collections in Java

Due date Jan 25th 2021
(No late submission for this HW)

As we discussed in the lectures, Java has a very structured Collections library. We can develop a similar simplified library for Java that uses arrays underneath. We will implement some the interfaces some abstract class, some concrete classes, and some of the helper classes such as iterators.



The above figure shows a simplified version of the Collections. We will write corresponding generic classes for Collection, Set, List, Queue, which are all interfaces. HashSet, ArrayList, and LinkedList are concrete classes. Note that LinkedList uses multiple inheritance. Following table defines the functions for each class

Class Name	Public Method Name	Definition
`Collection`	This is a generic class with one generic parameter which is the generic type E.	
	`iterator()`	Returns an iterator over the collection
	`add(E e)`	Ensures that this collection contains the specified element
	`addAll(Collection c)`	Adds all of the elements in the specified collection to this collection
	`clear()`	Removes all of the elements from this collection
	`contains(E e)`	Returns true if this collection contains the specified element.
	`containsAll(Collection c)`	Returns true if this collection contains all of the elements in the specified collection.
	`isEmpty()`	Returns true if this collection contains no elements.
	`remove(E e)`	Removes a single instance of the specified element from this collection, if it is present

	`removeAll(Collection c)`	Removes all of this collection's elements that are also contained in the specified collection
	`retainAll(Collection c)`	Retains only the elements in this collection that are contained in the specified collection
	`size()`	Returns the number of elements in this collection.
`Set`	A collection that contains no duplicate elements. There is no order for this collection. In other words, you don't have to keep the insertion order of the elements.	
`List`	An ordered collection (also known as a sequence). The user of this interface has precise control over where in the list each element is inserted.	
`Queue`	Queues order elements in a FIFO (first-in-first-out) manner. There is no random access with this Collection. Some functions throw exceptions.	
	`add(E e)`	Inserts the specified element into this queue
	`element()`	Retrieves, but does not remove, the head of this queue.
	`offer(E e)`	Inserts the specified element into this queue
	`poll()`	Retrieves and removes the head of this queue, or throws if this queue is empty.
`HashSet`	Implements Set functions	
`ArrayList`	Implements List functions	
`LinkedList`	Implements both List and Queue functions. Your class does not have to have a linked list to implement these.	
`Iterator`	**`hasNext()`**	Returns true if the iteration has more elements.
	`next()`	Returns the next element in the iteration and advances the iterator.
	`remove()`	Removes from the underlying collection the last element returned by this iterator. This method does not work for Queues, it throws an exception.

Your Java Collections hierarchy should use only Java arrays in the concrete classes to implement all the methods.

You will test each method of each concrete class with generic parameters of int and string.

Notes:

- Do error and range checking for any parameters. Throw exceptions and test them in your client code. Do not forget to define the throw lists for your functions.
- For each class you will write appropriate class documentation for Javadoc. You will also submit the Javadoc files.
- As expected, you should follow all object-oriented programming principles.
- You should submit your work to the moodle page.
- You should submit the images of drawn shapes.