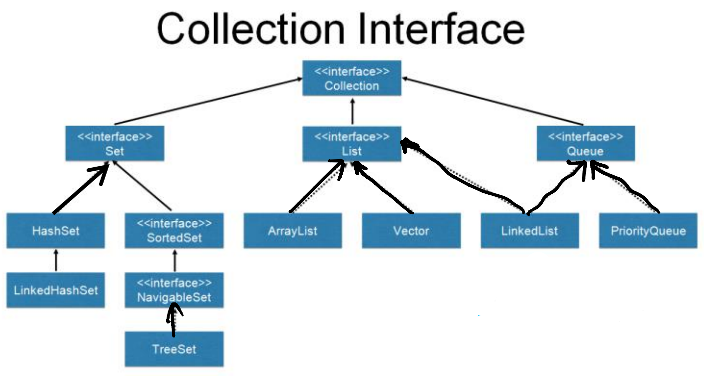
Gebze Technical University
Department of Computer Engineering
CSE 241/505
Object Oriented Programming
Fall 2018
Homework # 6
Templates and STL

Due date Dec 30th 2018

As we discussed in the lectures, Java has a very structured Collections library. We can develop a similar library for C++ that use STL classes underneath. We will implement some the interfaces (abstract classes with pure virtual functions only), 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 templated classes for Collection, Set, List, Queue, which are all abstract classes with all pure virtual functions with no data members. HashSet, ArrayList, and LinkedList are concrete classes. Note that LinkedList uses multiple inheritance. Following table defines the functions for each class

Class Name	Public Function Name	Definition
`Collection`	This is a templated class with two template parameters like the std::stack class of STL. The first parameter is the template type E and the second parameter is the STL container that will do all the work for us.	
	`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

Your C++ Collections hierarchy should use only STL containers in the concrete classes to implement all the functions. Second parameter of the Collection template can be any of **`std::vector`** (default container), **`std::list`**, and **`std::set`**.

You will test each function of each concrete class with template parameters of int and string. You will also use the three possible containers with all combinations.

Notes:

- Define and use your namespace.
- 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 its own header .h file and .cpp file for the separation of interface and implementation.
- As expected, you should follow all object-oriented programming principles and all submission rules.
- You should submit your work to the moodle page.
- You should submit the header and source code files and sample output results.