Due to the print book page limit, we cannot inlcude all good CheckPoint questions in the physical book. The CheckPoint on this Website may contain extra questions not printed in the book. The questions in some sections may have been reordered as a result. Nevertheless, it is easy to find the CheckPoint questions in the book on this Website. Please send suggestions and errata to Dr. Liang at y.daniel.liang@gmail.com. Indicate the book, edition, and question number in your email. Thanks!

Chapter 24 Check Point Questions

Section 24.2

▼24.2.1

Suppose list is an instance of MyList, can you get an iterator for list using list.iterator()?

Yes. Because MyList is Iterable.

Hide Answer

▼24.2.2

Can you create a list using new MyList()?

No. Because MyList is an interface.

Hide Answer

▼24.2.3

What methods in Collection are overridden as default methods in MyList?

The methods isEmpty(), contains(Object e), add(E e), remove(Object e), size(), containsAll(Collection<?> c), addAll(Collection<? extends E> c),removeAll(Collection<?> c), retainAll(Collection<?> c), toArray(), and toArray(T[] are overridden in MyList.

Hide Answer

▼24.2.4

What are the benefits of overriding the methods in Collection as default methods in MyList?

Default methods is a new feature in Java 8. The default methods providedefualt implementation for some methods in the interface.

Hide Answer

Section 24.3

▼24.3.1

What are the limitations of the array data type?

An array is a fixed-size data structure. Once an array is created, its size cannot be changed.

Hide Answer

▼24.3.2

MyArrayList is implemented using an array, and an array is a fixed-size data structure. Why is MyArrayList considered a dynamic data structure?

MyArrayList is implemented using an array and an array is a fixed-size data structure. But MyArrayList is considered as a dynamic data structure, because its storage size changes behind the scene and hidden from the user.

Hide Answer

▼24.3.3

Show the length of the array in MyArrayList after each of the following statements is executed.

```
1  MyArrayList<Double> list = new MyArrayList<>();
2  list.add(1.5);
3  list.trimToSize();
4  list.add(3.4);
5  list.add(7.4);
6  list.add(17.4);
```

After line 1, list's length is 16. After line 2, list's length is 16. After line 3, list's length is 1. After line 4, list's length is 3. After line 5, list's length is 3. After line 6, list's length is 7.

Hide Answer

▼24.3.4

What is wrong if lines 11-12 in Listing 24.2, MyArrayList.java,

```
    for (int i = 0; i < objects.length; i++)
      add(objects[i]);
```

are replaced by

```
data = objects;
size = objects.length;
```

If

```
for (int i = 0; i < objects.length; i++)
  add(objects[i]);
```

are replaced by

```
super(objects);
```

When constructing an ArrayList using new ArrayList(objects), the super class' constructor is invoked first to add element in objects to data. However, data has not been initialized yet. data will be initialized after the body of the superclass' constructor is executed. So you will get a NullPointerException when attempting to add an element to data. See Supplement III.I, "Initialization Block," for reference.If

```
for (int i = 0; i < objects.length; i++)
  add(objects[i]);
```

are replaced by

```
data = objects;
size = objects.length;
```

Then data and objects refer to the same array. This is a security hole. You may change ArrayList by directing changing the array elements through objects.

Hide Answer

▼24.3.5

If you change the code in line 33 in Listing 24.2, MyArrayList.java, from

```
E[] newData = (E[])(new Object[size * 2 + 1]);
```

to

```
E[] newData = (E[])(new Object[size * 2]);
```

the program is incorrect. Can you find the reason?

(Hint: To find the bug, perform trimToSize() on an empty list, then add a new element to the list.)When an empty array list is trimmed, its size becomes 0. If you create a new array by doubling its size, the new array size is still 0. Adding a new element now would cause an ArrayIndexOutOfBounds exception.

Hide Answer

▼24.3.6

Will the MyArrayList class have memory leak if the following code in line 41 is deleted?

```
data = (E[])new Object[INITIAL_CAPACITY];
```

Yes.

Hide Answer

▼24.3.7

The get(index) method invokes the checkIndex(index) method (lines 59-63 in Listing 24.2) to throw an IndexOutOfBoundsException if the index is out of bounds. Suppose the add(index, e) method is implemented as follows:

```
public void add(int index, E e) {
  checkIndex(index);

  // Same as lines 23-33 in Listing 24.2 MyArrayList.java
}
```

What will happen if you run the following code?

```
MyArrayList<> list = new MyArrayList<>();
list.add("New York");
```

list.add(e) invokes list.add(list.size(),e), which will throw an exception, because size is now 0.

Hide Answer

Section 24.4

▼24.4.1

If a linked list does not contain any nodes, what are the values in head and tail?

head and tail are null.

Hide Answer

▼24.4.2

If a linked list has only one node, is head == tail true? List all cases in which head == tail is true.

Yes. When the list is empty, head == tail is also true.

Hide Answer

▼24.4.3

Draw a diagram to show the linked list after each of the following statements is executed.

```
MyLinkedList<Double> list = new MyLinkedList<>();
list.add(1.5);
list.add(6.2);
list.add(3.4);
list.add(7.4);
list.remove(1.5);
list.remove(2);
```

Omitted.

Hide Answer

▼24.4.4

When a new node is inserted to the head of a linked list, will the head and the tail be changed?

The head will be changed. The tail will also be changed if the list is empty before the insertion.

Hide Answer

▼24.4.5

When a new node is appended to the end of a linked list, will the head and the tail be changed?

The head will be changed if the list is empty before the insertion. The tail will always be changed.

Hide Answer

▼24.4.6

Simplify the following code in Listing 24.5 using a conditional expression.

```
if (current != null) {
  result.append(", "); // Separate two elements with a comma
}
else {
  result.append("]"); // Insert the closing ] in the string
}
```

result.append((current != null) ? ", " : "]");

Hide Answer

▼24.4.7

Simplify the code for the removeLast() method by invoking the removeFirst() method when the size is less than or equal to 1. Is the new code more efficient in execution time?

```
public E removeLast() {
  if (size <= 1) {
    return removeFirst();
  }
  else {
    Node<E> current = head;
```

```
      for (int i = 0; i < size - 2; i++) {
        current = current.next;
      }

      Node<E> temp = tail;
      tail = current;
      tail.next = null;
      size--;
      return temp.element;
    }
  }
```

This new code is simpler and better. It reuses the existing code and makes the code easy to maintain. The time complexity for the new code is O(1), which is the same as before.
(Thanks to the UT Dallas students for their contribution. 10/14/2015)

Hide Answer

▼24.4.8
What is the time complexity of the addFirst(e) and removeFirst() methods in MyLinkedList?

O(1)

Hide Answer

▼24.4.9
What would be the time complexity for the size() method if the size data field is not used in MyLinkedList?

O(n)

Hide Answer

▼24.4.10
Suppose you need to store a list of elements. If the number of elements in the program is fixed, what data structure should you use? If the number of elements in the program changes, what data structure should you use?

If the number of elements is fixed in the program, use array is more efficient. If the number of elements changes in the program, you may use MyArrayList or MyLinkedList.

Hide Answer

▼24.4.11
If you have to add or delete the elements at the beginning of a list, should you use MyArrayList or MyLinkedList? If most of the operations on a list involve retrieving an element at a given index, should you use MyArrayList or MyLinkedList?

If you have to add or delete the elements anywhere in a list, use MyLinkedList. If most of operations on a list involve retrieving an element at a given index, use MyArrayList.

Hide Answer

▼24.4.12
Both MyArrayList and MyLinkedList are used to store a list of objects. Why do we need both types of lists?

Both MyArrayList and MyLinkedList are used to store a list of objects. Why do we need two? MyLinkedList is more efficient for deletion and insertion at the beginning of the list. MyArrayList is more efficient for all other operations.

Hide Answer

▼24.4.13

Implement the removeLast() method in a doubly linked list in O(1) time.

Note that in a singly linked list, to find the second-to-last node, you have loop all the way from head to the second-to-last node, which takes O(n) time. Here is the copy of the code from the text:

```
 1  public E removeLast() {
 2    if (size == 0) return null; // Nothing to remove
 3    else if (size == 1) { // Only one element in the list
 4      Node<E> temp = head;
 5      head = tail = null; // list becomes empty
 6      size = 0;
 7      return temp.element;
 8    }
 9    else {
10      Node<E> current = head;
11
12      for (int i = 0; i < size - 2; i++)
13        current = current.next;
14
15      Node<E> temp = tail;
16      tail = current;
17      tail.next = null;
18      size--;
19      return temp.element;
20    }
21  }
```

In a doubly linked list, the second-to-last node can be found in O(1) time using tail.previous. Here is the complete code:

```
 1  public E removeLast() {
 2    if (size == 0) return null; // Nothing to remove
 3    else if (size == 1) { // Only one element in the list
 4      Node<E> temp = head;
 5      head = tail = null; // list becomes empty
 6      size = 0;
 7      return temp.element;
 8    }
 9    else {
10      Node<E> temp = tail;
11      tail = tail.previous;
12      tail.next = null;
13      size--;
14      return temp.element;
15    }
```

Hide Answer

Section 24.5

▼24.5.1

You can use inheritance or composition to design the data structures for stacks and queues. Discuss the pros and cons of these two approaches.

Using inheritance: You can declare the stack class by extending the array list class, and the queue class by extending the linked list class. Using composition: You can declare an array list as a data field in the stack class, and a linked list as a data field in the queue class. Both designs are fine, but using composition is better because it enables you to declare a complete new stack class and queue class without inheriting the unnecessary and inappropriate methods from the array list and linked list.

Hide Answer

▼24.5.2

If LinkedList is replaced by ArrayList in lines 2-3 in Listing 24.6 GenericQueue.java, what will be the time complexity for the enqueue and dequeue methods?

The time complexity for enqueue will be O(1) and for dequeue will be O(n).

Hide Answer

▼24.5.3

Which lines of the following code are wrong?

```
1  List<> list = new ArrayList<>();
2  list.add("Tom");
3  list = new LinkedList<>();
4  list.add("Tom");
5  list = new GenericStack<>();
6  list.add("Tom");
```

Line 5 will be wrong, because GenericStack is not a subtype of MyList.

Hide Answer

Section 24.6

▼24.6.1

What is a priority queue?

In a priority queue, elements are assigned with priorities. When accessing elements, the element with the highest priority is removed first.

Hide Answer

▼24.6.2

What are the time complexity of the enqueue, dequeue , and getSize methods in MyProrityQueue?

For enqueue and dequeue in a priority queue, the complexity is O(logn). For getSize(), the complexity is O(1).

Hide Answer

▼24.6.3

Which of the following statements are wrong?

```
1  MyPriorityQueue<Object> q1 = new MyPriorityQueue<>();
```

```
2  MyPriorityQueue<Number> q2 = new MyPriorityQueue<>();
3  MyPriorityQueue<Integer> q3 = new MyPriorityQueue<>();
4  MyPriorityQueue<Date> q4 = new MyPriorityQueue<>();
5  MyPriorityQueue<> q5 = new MyPriorityQueue<>();
```

Lines 1 and 2 are wrong, because Object and Number don't implement Comparable.

Hide Answer