**Chapter 15 Concept Quiz**

Determine whether each of the following statements is true or false. If a statement is false, please explain why.

1. Compared to ArrayList, LinkedList is more efficient for insertions at a specified location.
2. Use an ArrayList if your application involves frequent insertions and removals.
3. Good programming practice for the implementation of LinkedList using Node is to define the Node class as its own independent public class.
4. In the implementation of the generic LinkedList using Node with ascending order, the generic definition can be completed as <T>.
5. In the implementation of the generic LinkedList using Node, the advancement of the cursor (a Node variable for looping over a LinkedList) can be coded as: cursor++;.
6. In the implementation of the generic LinkedList using Node, cur is a node to be deleted, prev is its predecessor, the code to delete cur node is: prev.next = cur.next;.
7. In the implementation of the generic Iterable LinkedList, the next() method in the implementation of Iterator interface returns the content of the Node succeeding the current node.
8. In the consideration of secured coding, it is good practice for the get() method in the LinkedList to return a deep copy of the content of the node instead of a shallow copy.
9. If a LinkedList implemented with Iterable interface, for-each loop is always recommended to use for iteration.
10. Java Collection’s LinkedList can be used to create instance of Queue ADT.

**Answers to the Quiz**

1. **True**. Linked implementation doesn’t involve shifting all subsequent elements one position to the right.
2. **False**. Use LinkedList is more efficient in this case.
3. **False**. It is recommended to define the Node class a generic, static inner class with private visibility.
4. **False**. In the implementation of the generic LinkedList using Node with ascending order, the generic definition can be completed as <T extends Comparable<T>>.
5. **False**. cursor = cursor.next.
6. **True**.
7. **False**. It returns the content of the current node.
8. **True**.
9. **False**. It is not recommended to use a for-each loop if addition or removal operations are involved during traversal.
10. **True**.