**Chapter 15**

**Short Answer Q**

1. **Do linked lists take more storage space than arrays of the same size?**

Yes, for two reasons. You need to store the node references, and each node is a separate object. (There is a fixed overhead to store each object in the virtual machine.)

1. **Why don’t we need iterators with arrays?**

An integer index can be used to access any array location.

1. **define LinkedListIterator ?**

private inner class of LinkedList .

1. **Trace through the addFirst method when adding an element to an empty list .**

When the list is empty, first is null. A new Node is allocated. It’s data instance variable is set to the newly inserted object. It’s next instance variable is set to null because first is null. The first instance variable is set to the new node. The result is a linked list of length 1.

1. **Why does the add method have two separate cases?**

If position is null, we must be at the head of the list, and inserting an element requires updating the first reference. If we are in the middle of the list, the first reference should not be changed.

1. **What is the advantage of viewing a type abstractly?**

You can focus on the essential characteristics of the data type without being distracted by implementation details .

1. **How much slower is the binary search algorithm for a linked list compared to the linear search algorithm?**

To locate the middle element takes *n* / 2 steps. To locate the middle of the subinterval to the left or right takes another *n* / 4 steps. The next lookup takes *n* / 8 steps. Thus, we expect almost *n* steps to locate an element. At this point, you are better off just making a linear search that, on average, takes *n* / 2 steps.

1. **Why wouldn’t you want to use a stack to manage print jobs?**

Stacks use a “last in, first out” discipline. If you are the first one to submit a print job and lots of people add print jobs before the printer has a chance to deal with your job, they get their printouts first, and you have to wait until all other jobs are completed.