

1. The Java `BufferedReader` class supports more operations than the simpler `InputStreamReader` class. Name and describe one of these operations. (What parameters are passed to it? What is the return value type? What does it do?)

The `BufferedReader` class has a zero-parameter `readLine()` method which returns an entire line of input (a sequence of characters terminated by an end of line character) as a Java `String` object.

2. Command line arguments are an easy way to pass a few input values to your Java program's `main()` method with signature

```
public static void main(String[] x)
```

(a) Write a Java `System.out.println()` statement that prints the number of command line arguments received by `main`.

```
System.out.println(x.length);
```

(b) Assume that one or more command line arguments is received by `main`. Write a Java statement (or statements) that declares a variable named `s` and assigns the value of the first command line argument to that variable.

```
String s = x[0];
```

3. Write a recursive Java method for computing  $n!$  ( $n$  factorial) for values of  $n \geq 0$ . Remember that  $0!$  is defined to be 1. Use the signature

```
int factorial(int n)
```

for your method.

```
public int factorial (int n) {  
    if (n == 0)  
        return 1;  
    else  
        return n*factorial(n-1);  
}
```

4. Use the following array as the starting point for parts (a), (b) and (c).

65	38	73	28	53	29	17	82	33	48
----	----	----	----	----	----	----	----	----	----

(a) Show the order of the array elements after one pass with the bubbleSort sorting algorithm.

38	65	28	53	29	17	73	33	48	82
----	----	----	----	----	----	----	----	----	----

(b) Show the order of the array elements after one pass with the insertionSort sorting algorithm.

38	65	73	28	53	29	17	82	33	48
----	----	----	----	----	----	----	----	----	----

(c) Show the order of the array elements after one pass with the selectionSort sorting algorithm.

17	38	73	28	53	29	65	82	33	48
----	----	----	----	----	----	----	----	----	----

5. Write a Java interface named `StackInterface` that describes the standard operations available for a stack.

```
public interface StackInterface {
    public void push(Object x);
    public Object pop();
    public Object top();
    public boolean isEmpty();
    public boolean isFull();
}
```

6. Write a Java interface named `QueueInterface` that describes the standard operations available for a queue.

```
public interface QueueInterface {
    public void enqueue(Object x);
    public Object dequeue();
    public Object front();
    public boolean isEmpty();
    public boolean isFull();
}
```

7. The Java Collections Framework contains sets (which extend the `AbstractSet` class and implement the `Set` interface) and lists (which extend the `AbstractList` class and implement the `List` interface).

(a) How do Java sets and lists differ with respect to allowing a collection to contain duplicate values?

Duplicates are not allowed in sets. Duplicates are allowed in lists.

(b) How do Java sets and lists differ with respect to allowing positional access to the elements of a collection?

Lists offer positional access to elements. Sets do not.

8. All the classes in the Java Collections Framework implement the `Iterable` interface. What capability does this require the Collections classes to have?

This requires the Collection classes to have an `iterator()` method which returns an `Iterator` object.

OR

This requires the classes to provide `Iterators`, which have `next()` and `hasNext()` methods that allow the elements of the collection to be traversed.

9. Write a complete Java class for nodes in a singly linked list. The type of data stored in a node can be any `Object` type. There should be at least one node constructor.

```
public class SLLNode {
    public Object data;
    public SLLNode next;
    public SLLNode() {
        data=null;
        next=null;
    }
    public SLLNode(Object x) {
        data =x;
        next=null;
    }
}
```

10. Most Java classes define constructors as well as attributes (fields/ variables) and methods (subroutines/ operations).

(a) What does a constructor usually do?

It initializes the object's fields.

(b) What Java reserved word is used when a constructor is invoked?

new

(c) What type of value, if any, does a constructor return?

a reference, or pointer, or address  
[as in `SLLNode n = new SLLNode();`]

11. In addition to classes and interfaces, the Java API also contains a number of abstract classes.

(a) What is an abstract class?

An abstract class is a class containing at least one abstract method. An abstract method is a method signature with no code implementing it.

[`public abstract void push(Object x);`]

(b) Why can't you invoke a constructor for an abstract class?

Java can't construct an object that hasn't been completely defined.

(c) What can you do with an abstract class (i.e., what is the ONLY THING you can do with an abstract class)?

You can extend it.

12. An expression used to represent the time necessary to search an unordered linear list containing  $N$  values for a particular value is  $O(N)$ , pronounced "big oh of  $N$ ".

(a) How (informally) is this expression derived as the time complexity of searching an unsorted list?

You may need to look at each of the  $N$  items in the list to complete the search. Thus, the number of steps required is proportional to  $N$ .

(b) If we know that a particular system needs one second to search an unordered list of 100,000 elements, how long should we expect it to take to search an unordered list of 1,000,000 elements?

10 seconds (since the list is 10 times as long)

13. Write a Java method with the signature

```
public void insertFirst(Object x)
```

that inserts a new node containing *x* as its data value at the front of a singly linked list. Assume that the pointer to the first node is named *head*, and that *SLLNode* is the name of the class that describes the list nodes.

```
public void insertFirst(Object x) {
    SLLNode n = new SLLNode(x);
    n.next = head;
    head = n;
}
```

14. Write a Java method with the signature

```
public Object removeFirst()
```

that removes the first node from a singly linked list and returns the data value stored in that node. Assume that the pointer to the first node is named *head*, and that *SLLNode* is the name of the class that describes the list nodes.

```
public Object removeFirst() {
    if (head != null) {
        Object o = head.data;
        head = head.next;
        return o;
    }
    else
        return null;
}
```

15. Write a Java method with the signature

```
public Object getLast()
```

that returns the data value stored in the last node in a singly linked list. The node is not to be removed from the list. Assume that the pointer to the first node is named *head*, and that *SLLNode* is the name of the class that describes the list nodes.

```
public Object getLast() {
    if (head != null) {
        SLLNode c = head;
        while (c.next != null)
            c = c.next;
        return c.data;
    }
    else
        return null;
}
```