

1. The standard input stream (usually the keyboard) is available to your Java program when it begins execution. The value stored in `System.in` is, in fact, a reference to an `InputStream` object which is already open and ready to supply input. What do we normally do to make it easier to read input from the keyboard (i.e., to read entire lines of input as `Strings`)?

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) Assume that one or more command line arguments is received by `main`. Write Java code that (1) declares a variable named `s` and (2) assigns the value of the first command line argument to that variable.

(b) Code a `for` loop that prints all the command line arguments.

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.

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

34	57	47	83	53	29	17	12	33	28
----	----	----	----	----	----	----	----	----	----

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

--	--	--	--	--	--	--	--	--	--

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

--	--	--	--	--	--	--	--	--	--

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

--	--	--	--	--	--	--	--	--	--

5. Write a Java interface named `StackInterface` that describes the standard operations available for a stack. Insert brief comments to describe the function each operation performs.

6. Write a Java interface named `QueueInterface` that describes the standard operations available for a queue. Insert brief comments to describe the function each operation performs.

7. The Java Collections Framework includes 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?

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

8. A number of classes in the Java Collections Framework implement the `RandomAccess` interface, the `Serializable` interface and/or the `Cloneable` interface.

(a) What capability does a class have (or claim to have) if it implements the `RandomAccess` interface?

(b) What capability does a class have (or claim to have) if it implements the `Serializable` interface?

(c) What capability does a class have (or claim to have) if it implements the `Cloneable` interface?

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.

10. Most Java classes contain fields, constructors and methods.

(a) What sort of Java code would you expect to see in the body of a constructor?
i.e., What does a constructor do?

(b) Although a constructor doesn't specify a type for a return value, a value is always returned when a constructor is invoked.

(e.g. `BigInteger b = new BigInteger("12345");`).
Describe the return value. What does it represent? What can be done with it?

11. The Java API contains classes, interfaces, and abstract classes.

(a) What is an interface? How is an interface used?

(b) What is the difference between an ordinary class and an abstract class?

12. An expression used to represent the time necessary to sort an array using the bubblesort algorithm is $O(N^2)$, pronounced "big oh of N squared".

(a) How (informally) is this expression derived as the time complexity of bubblesorting an array?

(b) If a particular system needs ten seconds to bubblesort an array of 1,000 elements, how long should we expect it to take to bubblesort an array of 10,000 elements? [Note: the second array is 10 times the size of the first.]

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.

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. If the list is empty, return `null`. 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.

15. Write a Java method with the signature

```
public void insertLast(Object x)
```

that appends a new node containing `x` as its data value to the end 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.