A3:

1 →

a. When a variable is declared final, it's value can't be modified, so the variable becomes constant, hence we always need to initialize it. Further, if the variable points to the reference, then the variable can't be rebound to point to another object, however the internal state of the object pointed by the final variable can be changed, meaning if the final variable is an array type we can add and remove elements from it.
b. A final class cannot be extended(inherited). It also creates an immutable class.
c. A final method cannot be overridden.

A4:

Nested class of the outer class can be made static, which can be instantiated without instantiating outer class, and can only access the static members of outer class.

For eg:

```
public class OuterClass {
    public static String msg = "Outer Message";
    public static class NestedStaticClass {
        public void printMsg() {
            System.out.println("Message from inside of static nested class: " + msg);
        }
    }
}
```

```
public static void main(String[] args) {
    OuterClass.NestedStaticClass instanceNestStaticClass = new OuterClass.NestedStaticClass();
    instanceNestStaticClass.printMsg();
}
```

A5:

1 → I would probably exclude numSides parameter from "Shape" class.

2 → Constructor chaining is the process of calling one constructor from another constructor with respect to the current object. Constructor chaining can be done in two ways:

Within same class: It can be done using this() keyword for constructors in same class
From base class: by using super() keyword to call constructor from the base class.

Constructor chaining occurs through inheritance. A subclass constructor's task is to call super class's constructor first. This ensures that creation of the sub class's object starts with the initialization of the data members of the super class. There could be any number of classes in the inheritance chain. Every constructor calls up the chain till class at the top is reached.

A6:

1 →

ArrayList	LinkedList	Vector
Uses dynamic array to store elements	Uses doubly linked list to store elements	Uses dynamic array to store elements
Implements List Interface	Implements both List and Deque interface	Implements List Interface
Better for random access of element and storing element to the end (get, insertLast, deleteLast)	Better for manipulating stored data (getFirst, getLast, insertFirst, insertLast)	Same as AarrayList
Manipulating takes longer time, since if we want to remove any element, the memory bits needs to be shifted	No concept of memory bits shifting. List is traversed and reference link is changed	Same as ArrayList
Not Synchronized		Synchronized and Thread Safe

2 →

Hashset	LinkedHashset	Treeset
Internally uses HashMap for storing objects	Internally uses HashMap for storing objects	Internally uses TreeMap for storing objects
Used when you want to store unique objects without maintaining any order.	Used when you want to store unique objects maintaining the insertion order.	Used when you want to sort the elements using some Comparator.
O(1) : insert, remove, retrieve	O(1) : insert, remove, retrieve	O(logn) : insert, remove, retrieve
Allows one Null	Allows one Null	No Null
HashSet obj = new HashSet();	LinkedHashSet obj = new LinkedHashSet();	TreeSet obj = new TreeSet();

3 →

Hashmap	Linkedhashmap	Treemap
O(1) : insert, lookup	O(1) : insert, lookup	O(logn) : lookup
Allows one Null key Multiple Null Value	Allows one Null key Multiple Null Value	No Null key Multiple Null Value
No order	Order of insertion	Maintains ascending order

4 →

Hashmap	Hashtable
Non-synchronized, Thread unsafe	Synchronized, Thread safe
Allows one Null key Multiple Null Value	No Null key or value

A7 →

1. Checked Exceptions
2. Error subclass has the least exception, because these are basically unchecked exceptions and are thrown because of external problems, and there can be just so many of them.
3. Throwable is a class not interface, because we can also directly instantiate this because they have state, and class hierarchy checks are cheaper then interface checks.
4. I think all except D can be thrown, because I don't think it is a subclass of Throwable.
5. Because the exception thrown has not been caught.
6. Multi-catch feature
7. Exception contains basically both Runtime exception and checked exceptions, however errors contains unchecked exceptions thrown by external problems.
8. Checked exception: Compile Time Exception, Eg: FileNotFound Exception.
   Unchecked exception: Run Time Exception

A8 →

1. Process is a self contained execution environment and has its own memory space, while thread exists within a process and shares process resources (memory, open files). Process has at least one thread, and can have multiple of them. Threads are used to concurrently run different tasks.
2. Threads can be implemented either by extending Thread class, or by implementing Runnable Interface. Thread class implementation should be used when we want to modify the members of Thread class, otherwise we can simply stick with Runnable Interface.

3. start(): This starts the thread process.
   run(): It is the method in a thread class, that is executed when thread is started.
   Yes, we can call run() method of Thread class.

4. wait(): causes current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object
   sleep(): causes currently executing thread to sleep for specified time

5. Threads communicate in 3 ways:
    1. wait()
    2. notify()
    3. notifyAll()