14.What is AVL Tree?

Answer: AVL trees are well-balanced. In an AVL tree, the difference between the heights of two subtrees for every node is 0 or 1. It can be shown that the maximum height of an AVL tree is *O*(log *n)*. The process for inserting or deleting an element in an AVL tree is the same as in a regular binary search tree, except that you may have to rebalance the tree after an insertion or deletion operation.

15. What is merge sort, bubble sort and bucket sort?

Answer: **The merge sort** algorithm divides the array into two halves and applies a merge sort on each half recursively. After the two halves are sorted, merge them.

The bubble sort algorithm makes several passes through the array. On each pass, successive neighboring pairs are compared. If a pair is in decreasing order, its values are swapped; otherwise, the values remain unchanged.

Bucket sort can be exceptionally fast because of the way elements are assigned to buckets, typically using an array where the index is the value. This means that more auxiliary memory is required for the buckets at the cost of running time than more comparison sorts. It runs in O(n+k) time in the average case where n is the number of elements to be sorted and k is the number of buckets. While bucket sort is a distribution sort, it typically uses a comparison sort to sort the buckets after they have been allocated.

16.What is an event handler?

Answer: The EventHandler class provides support for dynamically generating event listeners whose methods execute a simple statement involving an incoming event object and a target object. The EventHandler class is intended to be used by interactive tools, such as application builders, that allow developers to make connections between beans.

17.What is Adapter class?

Answer: Java adapter classes provide the default implementation of listener interfaces. If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces. So it saves code. The adapter classes are found in java.awt.event, java.awt.dnd and javax.swing.event packages.

18.What is Stream? How you classify them?

Answer: A Java I/O object is called a stream. An object for reading data is called an input stream and an object for writing data is called an output stream.

19.What is Serialization & Deserialization?

Answer: **Serialization in Java** is a mechanism of *writing the state of an object into a byte stream*.

It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies.

The reverse operation of serialization is called *deserialization*.

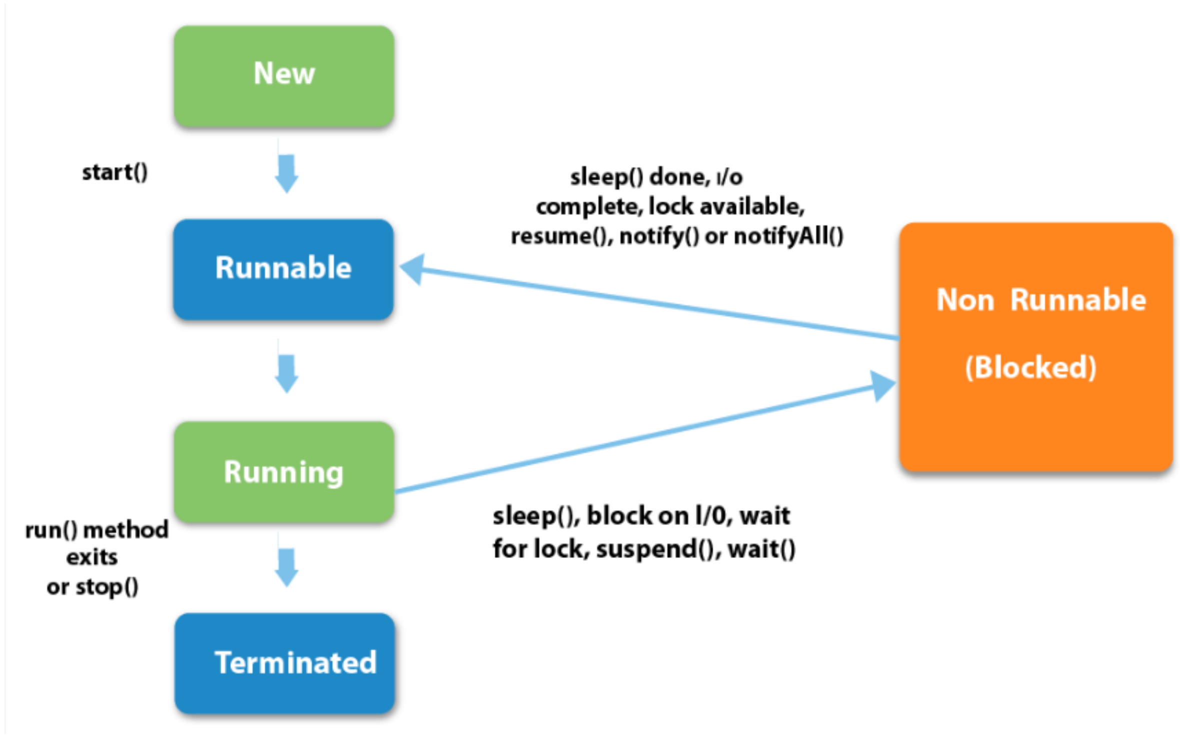
A serializable object is an instance of the **java.io.Serializable** interface, so the object's class must implement **Serializable**.

20. Write the Lifecycle method of a thread?

Answer: A thread can be in one of the five states. According to sun, there is only 4 states in **thread life cycle in java** new, runnable, non-runnable and terminated. There is no running state.

The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

1. New
2. Runnable
3. Running
4. Non-Runnable (Blocked)
5. Terminated


New
start()
Runnable
sleep() done, I/o complete, lock available, resume(), notify() or notifyAll()
Non Runnable (Blocked)
Running
sleep(), block on I/0, wait for lock, suspend(), wait()
run() method exits or stop()
Terminated

1) New

The thread is in new state if you create an instance of Thread class but before the invocation of start() method.

2) Runnable

The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.

3) Running

The thread is in running state if the thread scheduler has selected it.

4) Non-Runnable (Blocked)

This is the state when the thread is still alive, but is currently not eligible to run.

5) Terminated

A thread is in terminated or dead state when its run() method exits.

21. What do you mean by MVC?

Answer: MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

- **Model** - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
- **View** - View represents the visualization of the data that model contains.
- **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

22. What are the type of events?

Answer: Your program may need to respond to many different kinds of events—from menus, from buttons, from the mouse, from the keyboard, and from a number of other components. To have a structured approach to handling events, the events are broken down into subsets. At the topmost level, there are two broad categories of events in Java:

❑ **Low-Level Events**—These are system-level events that arise from the keyboard or from the mouse, or events associated with operations on a window, such as reducing it to an icon or closing it. The meaning of a low-level event is something like “the mouse was moved,” “this window has been closed,” or “this key was pressed.”

❑ **Semantic Events**—These are specific component-related events such as pressing a button by clicking it to cause some program action or adjusting a scrollbar. They originate, and you interpret them, in the context of the GUI you have created for your program. The meaning of a semantic event is typically along the lines of “the OK button was pressed,” or “the Save menu item was selected.” Each kind of component, a button or a menu item, for example, can generate a particular kind of semantic event.

23.What is collection? Write the type of collection.

Answer: The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.

All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion, etc. can be achieved by Java Collections.

The Java Collections Framework supports two types of containers:

■ One for storing a collection of elements is simply called a *collection*.

■ The other, for storing key/value pairs, is called a *map*.

Maps are efficient data structures for quickly searching an element using a key. We will introduce

maps in the next chapter. Now we turn our attention to the following collections.

■ **Set**s store a group of nonduplicate elements.

■ **List**s store an ordered collection of elements.

■ **Stack**s store objects that are processed in a last-in, first-out fashion.

■ **Queue**s store objects that are processed in a first-in, first-out fashion.

■ **PriorityQueue**s store objects that are processed in the order of their priorities.

24.What are the differences between Swing and AWT?

Answer: The java.awt package was the primary repository for classes you would use to create a GUI way back in Java 1.1—awt being an abbreviation for **A**bstract **W**indowing **T**oolkit—but many of the classes this package defines have been superseded in Java 2 by javax.swing. Most of the classes in the javax.swing package define GUI elements, referred to as **Swing components**, that provide much-improved alternatives to components defined by classes in java.awt. You’ll be looking into the JButton class in the Swing set that defines a button, rather than the Button class in java.awt.

25.What is different between JFrame and JWindow?

Answer: **JFrame** is used as the basic Java application window. An object of this class has a title bar and provision for adding a menu. You can also add other components to it. You will usually subclass this class to create a window class specific to your application. You'll then be able to add GUI components or draw in this window if required, as you'll see.

JWindow is an object of this class type is a window with no title bar or window management icons. One typical use for a JWindow object is for a subsidiary application window that is displayed on a secondary display where two or more displays are attached to a system.