1. **What do you mean by Generic?**

J2SE 5.0 provides compile-time type safety with the Java Collections framework through generics .This allow us to specify, at compile-time, the types of objects we want to store in a Collection. So we don't need to cast anything.

1. **What is a Thread?**

A thread is the flow of execution of a single set of program statements. The Thread Class allows multitasking (i.e. running several tasks at the same time) by instantiating (i.e. creating) many threaded objects, each with their own run time characteristics.

1. **What are the two ways of creating thread?**

Ans) There are two ways to create a new thread.

a) Extend the Thread

b) Implements the Runnable interface e.g.

1. **What is the difference between wait() and sleep()?**

Ans) 1) wait() is a method of Object class. sleep() is a method of Thread class.

2) sleep() allows the thread to go to sleep state for x milliseconds. When a thread goes into sleep state it doesn’t release the lock. wait() allows thread to release the lock and goes to suspended state. The thread is only active when a notify() or notifyAll() method is called for the same object.

1. **What is deadlock?**

When two threads are waiting for each other and can’t proceed until the first thread obtains a lock on the other thread or vice versa, the program is said to be in a deadlock.

1. **What do you mean by Generic?**

J2SE 5.0 provides compile-time type safety with the Java Collections framework through generics .This allow us to specify, at compile-time, the types of objects we want to store in a Collection. So we don't need to cast anything.

1. **Difference between Comparable and Comparator Interface?**

|  |  |
| --- | --- |
| Comparable | Comparator |
| Comparable is a member of java.lang package. | Comparator is a member of java.util.package. |
| Sorts the objects is natural order. | Sorts the objects is custom order. |
| Its method is compareTo( object) | Its method is compare(object1, object2) |

1. **What is File?**

Files can be classified as either text or binary. A file that can be processed (read, created, or modified) using a text editor such as Notepad on Windows or vi on UNIX is called a *text file*. All the other files are called *binary files*. You cannot read binary files using a text editor—they are designed to be read by programs. For example, Java source programs are text files and can be read by a text editor, but Java class files are binary files and are read by the JVM.

1. **What is buffer, BufferedReader and BufferedWriter?**

Buffer is a container for data of a specific primitive type. A buffer is a linear, finite sequence of elements of a specific primitive type.

BufferedReader Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines. The buffer size may be specified, or the default size may be used. The default is large enough for most purposes.

Writes text to a character-output stream, buffering characters so as to provide for the efficient writing of single characters, arrays, and strings. The buffer size may be specified, or the default size may be accepted. The default is large enough for most purposes.

1. **What is List, Stack, Set? Difference between list and set in Java.**

The List interface extends the Collection interface and defines a collection for storing elements in a sequential order. To create a list, use one of its two concrete classes: ArrayList or LinkedList.

The Stack class represents a last-in-first-out (LIFO) stack of objects. It extends class Vector with five operations that allow a vector to be treated as a stack. The usual push and pop operations are provided, as well as a method to peek at the top item on the stack, a method to test for whether the stack is empty, and a method to search the stack for an item and discover how far it is from the top.

A collection that contains no duplicate elements. More formally, sets contain no pair of elements e1 and e2 such that e1.equals(e2), and at most one null element. As implied by its name, this interface models the mathematical set abstraction.

|  |  |
| --- | --- |
| **List** | **Set** |
| 1) List is an ordered collection it maintains the insertion order, which means upon displaying the list content it will display the elements in the same order in which they got inserted into the list. | 1) Set is an unordered collection, it doesn’t maintain any order. There are few implementations of Set which maintains the order such as LinkedHashSet (It maintains the elements in insertion order). |
| 2) List allows duplicates while Set doesn’t allow duplicate elements. | 2) All the elements of a Set should be unique if you try to insert the duplicate element in Set it would replace the existing value. |
| 3) List implementations: *ArrayList, LinkedList etc.* | 3) Set implementations: *HashSet, LinkedHashSet, TreeSet etc.* |

1. **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.

1. **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 together.

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.

1. **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.

1. **What is Adapter class?**

Answer: Java adapter classes provide the default implementation of listener interfaces. If we inherit the adapter class, we 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.

1. **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.

1. **What is Serialization & Deserialization?**

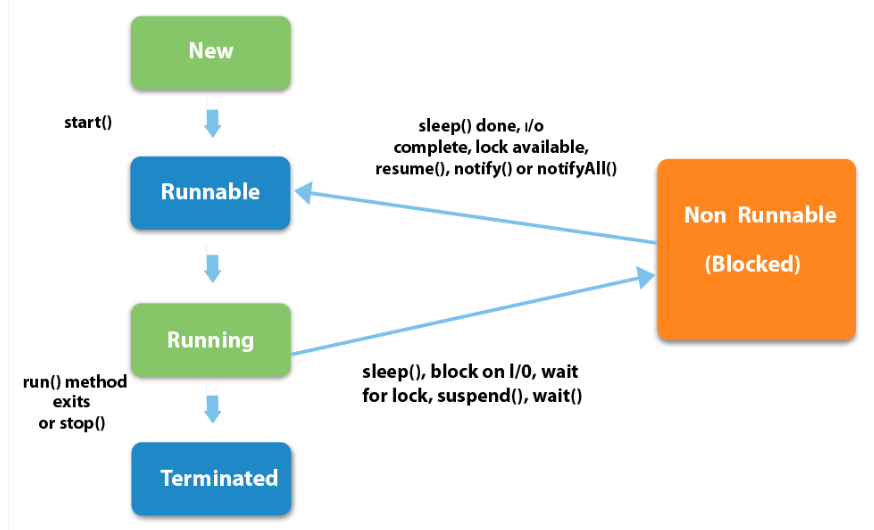
Answer: **Serialization in Java** is a mechanism of *writing the state of an object into a byte stream*. It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies. The reverse operation of serialization is called *deserialization*. A serializable object is an instance of the **java.io.Serializable** interface, so the object’s class must implement **Serializable**.

1. **Write the Lifecycle method of a thread?**

Answer: A thread can be in one of the five states. There are 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



|  |
| --- |
| 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.

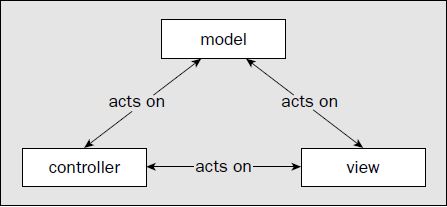
5) Terminated

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

1. **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.



1. **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 we interpret them, in the context of the GUI you have created for our 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.

1. **What is collection? Write the type of collection.**

Answer: The **Collection in Java** is a framework that provides architecture to store and manipulate a 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.

■ **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.

1. **What are the differences between Swing and AWT?**

Answer: The java.awt package was the primary repository for classes we 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 of 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. For example we’ll be looking into the JButton class in the Swing set that defines a button, rather than the Button class in java.awt.

1. **What is different between JFrame and JWindow?**

|  |  |
| --- | --- |
| **JFrame:** | **JWindow:** |
| A resizable, movable window with title bar and close button. | A window without a title bar or move controls. The program can move and resize it, but the user cannot. |
| The entire application is usually a JFrame. | It optionally has a parent JFrame |
| We can also add other components to it. | It is for a subsidiary application window. |