**Java Fundamentals:**

* **What is JVM and Is it platform independent?**

Java Virtual Machine (JVM) is the execution engine of java programming language. JVM is responsible for converting byte code into machine readable code. JVM is not platform independent, thats why you have different JVM for different operating systems. We can customize JVM with Java Options, such as allocating minimum and maximum memory to JVM. It’s called virtual because it provides an interface that doesn’t depend on the underlying OS.

* **What is difference between JDK and JVM and JRE?**

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms. JVM, JRE and JDK are platform dependent because configuration of each OS differs. But, Java is platform independent.

The JVM performs following main tasks:

Loads code, Verifies code, Executes code

Provides runtime environment

JRE is an acronym for Java Runtime Environment . It is used to provide runtime environment. It is the implementation of JVM. It physically exists. It contains set of libraries + other files that JVM uses at runtime.

JDK is an acronym for Java Development Kit.It physically exists.It contains JRE + development tools.

* **Why Java is not pure Object Oriented language?**

Java is not a pure object oriented language because it supports Primitive datatype such as int, byte, long? etc, to be used, which are not objects.

* **What are principle concepts of OOPS?**

Abstraction, Encapsulation, Inheritance, Polymorphism

* **What is Java Package and which package is imported by default?**

A java package is a group of similar types of classes, interfaces and sub-packages.Package in java can be categorized in two form, built-in package and user-defined package.java.lang package is default impored package.

* **What is Overloading and Overriding in Java?**

|  |  |
| --- | --- |
| **Overloading** | **Overriding** |
| Method overloading is used to increase readability of the program. | Method overriding is used to provide the specific implementation of the method that is already provided by its super class. |
| Method overloading is performed within class. | Method overriding occurs *in two classes* that have IS-A (inheritance) relationship. |
| In case of method overloading, *parameter must be different*. | In case of method overriding, *parameter must be same*. |
| Method overloading is the example of *compile time polymorphism*. | Method overriding is the example of *run time polymorphism*. |
| In java, method overloading can't be performed by changing return type of the method only. *Return type can be same or different* in method overloading. But you must have to change the parameter. | *Return type must be same or covariant* in method overriding. |

* **What is the difference between an Inner Class and a Sub-Class?**

A class would be called subclass if it extends some another class. This child class would be called subclass and that parent class would be called as super-class. On the other hand, nested classes are those which are defined inside a class.

* **What is the difference between abstract class and interface?**

Simply, abstract class achieves partial abstraction (0 to 100%) whereas interface achieves fully abstraction (100%).

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| Abstract class can **have abstract and non-abstract**methods. | Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.. |
| Abstract class **doesn't support multiple inheritance**. | Interface **supports multiple inheritance**. |
| Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static and final variables**. |
| Abstract class **can provide the implementation of interface**. | Interface **can't provide the implementation of abstract class**. |
| The **abstract keyword** is used to declare abstract class | The **interface keyword** is used to declare interface. |

* **What is static binding and dynamic binding?**

Static binding: When type of the object is determined at compiled time(by the compiler), it is known as static binding. If there is any private, final or static method in a class, there is static binding.

Dynamic binding: When type of the object is determined at run-time, it is known as dynamic binding.

* **What is Data Encapsulation and what’s its significance?**

**Data encapsulation**, also known as **data** hiding, is the mechanism whereby the implementation details of a class are kept hidden from the user. The user can only perform a restricted set of operations on the hidden members of the class by executing special functions commonly called methods.

* **What is Java Bean Class?**

JavaBeans are classes that encapsulate many [objects](https://en.wikipedia.org/wiki/Object_(computer_science)) into a single object (the bean). They are serializable, have a [zero-argument constructor](https://en.wikipedia.org/wiki/Nullary_constructor), and allow access to properties using [getter and setter methods](https://en.wikipedia.org/wiki/Mutator_method).

* **What are Access modifiers?**

Access modifiers are keywords in object-oriented languages that set the accessibility of classes, methods, and other members. Access modifiers are a specific part of programming language syntax used to facilitate the encapsulation of components.

* **What’s the benefit of using inheritance?**

Reusability,facility to use public methods of base class without rewriting the same. Overriding with inheritance, we will be able to override the methods of the base class so that meaningful implementation of the base class method can be designed in the derived class.

* **Why multiple Inheritance not supported in Java?**

Java doesn’t allow multiple inheritance to avoid the ambiguity caused by it. One of the example of such problem is the diamond problem that occurs in multiple inheritance.

* **What is the diamond problem in inheritance?**

The "diamond problem" ("deadly diamond of death") is an ambiguity that arises when two classes B and C inherit from A, and class D inherits from both B and C. If there is a method in A that B and C have [overridden](https://en.wikipedia.org/wiki/Method_overriding_(programming)), and D does not override it, then which version of the method does D inherit: that of B, or that of C?, this is situation that is called as diamond problem.

* **What is the difference between break and continue statement?**

An unlabeled break statement terminates the innermost switch, for, while, or do-while statement, but a labeled break terminates an outer statement.

The continue statement skips the current iteration of a for, while , or do-while loop. The unlabeled form skips to the end of the innermost loop's body and evaluates the boolean expression that controls the loop.

* **What is nested class?**

**Java inner class** or nested class is a class which is declared inside the class or interface. We use inner classes to logically group classes and interfaces in one place so that it can be more readable and maintainable. Additionally, it can access all the members of outer class including private data members and methods.

* **How are this() and super() used with Constructor?**

Constructors use this to refer to another constructor in the same class with a different parameter list.

Constructors use super to invoke the super class's constructor. If a constructor uses super, it must use it in the first line; otherwise, the compiler will complain.

* **What is Serialization and Deserialization**

**Serialization** is the process of converting an object into a stream of bytes in order to store the object or transmit it to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed. The reverse process is called **deserialization**.

* **What is difference between Heap and Stack Memory?**

Java Heap space is used by java runtime to allocate memory to Objects and JRE classes. Whenever we create any object, it’s always created in the Heap space.

Garbage Collection runs on the heap memory to free the memory used by objects that doesn’t have any reference. Any object created in the heap space has global access and can be referenced from anywhere of the application.

* **How garbage collection is done in Java?**

The garbage collector will look for objects which aren't being used anymore, and gets rid of them, freeing up the memory so other new objects can use that piece of memory.

* **How destructors are defined in Java?**

A destructor is a special method called automatically during the destruction of an object. Actions executed in the destructor include the following:

* Recovering the heap space allocated during the lifetime of an object
* Closing file or database connections
* Releasing network resources
* Releasing resource locks

**Array:**

* **What do you mean by an Array? How to create?**

Array is an indexed collection of homogeneous elements, they are fixed in size, to create an array we have to declare, instantiate and initialize the java array together by:

int a[]={33,3,4,5};//declaration, instantiation and initialization

* **Advantages and disadvantages of Array?**

**Advantages:**

Code Optimization**:** It makes the code optimized, we can retrieve or sort the data easily.

Random access: We can get any data located at any index position.

**Disadvantages:**Size Limit: We can store only fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in java.

* **What is the meaning of anonymous array? Explain with an example?**

An array without name is known as anonymous [array in java](https://www.thecrazyprogrammer.com/2015/05/array-in-java-1-d.html).

* Anonymous array is passed as an argument of method.
* Anonymous array is created and initialized in the same line.

class AnonymousArray

{

static void print(int a[])

{

  for(int i=0;i<a.length;++i)

   System.out.print(a[i]+" ");

}

static void print(int a[][])

{

  for(int i=0;i<a.length;++i)

  {

   for(int j=0;j<a[i].length;++j)

    System.out.print(a[i][j]+" ");

   System.out.println("");

  }

}

public static void main(String...s)

{

  //1d anonymous array

  print(new int[]{10,20,30,40});

  System.out.println("n");

  //2d anonymous array

  print(new int[][]{{10,20},{30,40},{50,60}});

}

}

* **What are “jagged” arrays in java?**

It is a new feature supported by Java. In Jagged arrays, each row, in a two-dimensional array, may contain different lengths. Let us design a two-dimensional array with 4 rows where the first row contains 4 elements, the second row with 1 element, the third row with 2 elements and the fourth row with 3 elements.

* **How to copy an array into another array?**

public class A {

public static void main(String args[]) {

int a[] = { 1, 2, 3, 4, 5, 6 };

int b[] = new int[a.length];

for (int i = 0; i < a.length; i++) {

b[i] = a[i];

}

}

}

* **What is difference between ArrayIndexOutfOBounds and ArrayStoreException?**

Public class ArrayIndexOutOfBoundsException extends IndexOutOfBoundsException

Thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.

public class ArrayStoreException extends RuntimeException

Thrown to indicate that an attempt has been made to store the wrong type of object into an array of objects.

Object x[] = new String[3];  
x[0] = new Integer(0);

* **Where does array stored in memory?**

It is stored in the heap. Array is an object, and objects are stored at heap.

**String:**

* **What is String in Java?**

String is built in final class in java.lang package, object of String is used represent group of characters enclosed in “ ”. String object is immutable.

* **What are different ways to create String Object?**

By literal – String s1= “Hello World”;

By new keyword- String s2= new String (“Hello World”);

* **What is String subSequence method?**

Java 1.4 introduced CharSequence interface and String implements this interface, this is the only reason for the implementation of subSequence method in String class. Internally it invokes the [String substring](https://www.journaldev.com/807/java-string-substring) method.

String subSequence method returns a character sequence that is a subsequence of this sequence. An invocation of this method of the form str.subSequence(begin, end) behaves in exactly the same way as the invocation of str.substring(begin, end).

* **How to convert String to char and vice versa?**

char chararray[]=string.toCharArray();//the arrray contains all the characters in the string .

String chartostring = Character.toString(chararrayname);

* **How to convert String to byte array and vice versa?**

String input = "Hello World";

      byte[] bytes = input.getBytes(Charset.forName("UTF-8"));

System.out.println("Input : " + input);

      System.out.println("Input [Byte Format] : " + bytes);

String s = new String(bytes, Charset.forName("UTF-8")); // for UTF-8 encoding

System.out.println("String created from byte array in UTF-8 encoding: " + s);

* **Difference between String, StringBuffer and StringBuilder?**

String : It is an immutable class, which provides lot of utility methods for String object.

StringBuffer: it is synchronized i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.

StringBuffer is less efficient than StringBuilder.

StringBuilder: is non-synchronized i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.

StringBuilder is more efficient than StringBuffer.

* **Why String is immutable or final in Java?**

String is immutable for several reasons, here is a summary:

* **Security**: parameters are typically represented as String in network connections, database connection urls, usernames/passwords etc. If it were mutable, these parameters could be easily changed.
* **Synchronization** and concurrency: making String immutable automatically makes them thread safe thereby solving the synchronization issues.
* **Caching**: when compiler optimizes your String objects, it sees that if two objects have same value (a="test", and b="test") and thus you need only one string object (for both a and b, these two will point to the same object).

**Class loading**: String is used as arguments for class loading. If mutable, it could result in wrong class being loaded (because mutable objects change their state).

* **What is String Pool?**

String Pool in Java corresponds to an allocation of memory in Java heap memory. It consists of a collection of String objects, which are shared and reused among several String object references for same String content.

Note : This capability is gained through the [immutability](http://www.codedjava.com/2017/09/immutable-objects-in-java_19.html) nature of Java String.

* **What does String intern() method do?**

According to [String#intern()](https://docs.oracle.com/javase/9/docs/api/java/lang/String.html" \l "intern--), intern method is supposed to return the String from the String pool if the String is found in String pool, otherwise a new string object will be added in String pool and the reference of this String is returned.

* **Why String is popular HashMap key in Java?**

Since String is immutable, its hashcode is cached at the time of creation and it doesn’t need to be calculated again. This makes it a great candidate for key in a Map and its processing is fast than other HashMap key objects. This is why String is mostly used Object as HashMap keys.

**Collections:**

* **What is Java Collections Framework? List out some benefits of Collections framework?**

A Collection is a group of individual objects represented as a single unit. Java provides Collection Framework which defines several classes and interfaces to represent a group of objects as a single unit.

The Collection interface (**java.util.Collection**) and Map interface (**java.util.Map**) are two main root interfaces of Java collection classes.

Advantages:

Consistent API

Reduces programming effort

Increases program speed and quality

* **What is the benefit of Generics in Collections Framework?**
* There are mainly 3 advantages of generics. They are as follows:
* **1) Type-safety :** We can hold only a single type of objects in generics. It doesn’t allow to store other objects.
* **2) Type casting is not required:** There is no need to typecast the object.
* **3) Compile-Time Checking:** It is checked at compile time so problem will not occur at runtime. The good programming strategy says it is far better to handle the problem at compile time than runtime.
* **What are the basic interfaces of Java Collections Framework?**
* **Collection:** Collection is the root of the collection hierarchy. A collection represents a group of objects known as its elements. The Java platform doesn’t provide any direct implementations of this interface.
* **Set:** Set is a collection that cannot contain duplicate elements. This interface models the mathematical set abstraction and is used to represent sets, such as the deck of cards.
* **List:** List is an ordered collection and can contain duplicate elements. You can access any element from its index. List is more like array with dynamic length.
* **Map:** A Map is an object that maps keys to values. A map cannot contain duplicate keys: Each key can map to at most one value.
* Some other interfaces are Queue, Dequeue, Iterator, SortedSet, SortedMap and ListIterator.
* **What are common algorithms implemented in Collections Framework?**

Java Collections Framework provides algorithm implementations that are commonly used such as sorting and searching. Collections class contain these method implementations. Most of these algorithms work on List but some of them are applicable for all kinds of collections. Some of them are sorting, searching, shuffling, min-max values.

* **Why Collection doesn’t extend Cloneable and Serializable interfaces?**

1.Need for Serializable interface:An object created in java may sometimes have to travel across the network to another remote network [area.So](http://area.so/" \t "_blank) in order to facilitate this, every collection object implements Serializable Interface.

2.Need for Cloneable Interface: After travelling across the network, the object may undergo unforeseen corruption,so they make a shallow copy of the object that and work on it to check for any corrupted data.

Cloneable Interface implements Clone() which makes a shallow copy of the object.This prevents the loss of time and [data.So](http://data.so/" \t "_blank) every collection class implements cloneable interface too.

* **What is difference between Array and ArrayList? When will you use Array over ArrayList?**

Arrays can contain primitive or Objects whereas ArrayList can contain only Objects. Arrays are fixed size whereas ArrayList size is dynamic. Arrays doesn’t provide a lot of features like ArrayList, such as addAll, removeAll, iterator etc.

Although ArrayList is the obvious choice when we work on list, there are few times when array are good to use.

* **What are similarities and difference between ArrayList and Vector?**

ArrayList and Vector are similar classes in many ways:

1. Both are index based and backed up by an array internally.

2. Both maintains the order of insertion and we can get the elements in the order of insertion.

3. The iterator implementations of ArrayList and Vector both are fail-fast by design.

4. ArrayList and Vector both allows null values and random access to element using index number.

These are the differences between ArrayList and Vector:

1. Vector is synchronized whereas ArrayList is not synchronized. However if you are looking for modification of list while iterating, you should use CopyOnWriteArrayList.

2. ArrayList is faster than Vector because it doesn’t have any overhead because of synchronization.

3. ArrayList is more versatile because we can get synchronized list or read-only list from it easily using Collections utility class.

* **Why Map interface doesn’t extend Collection interface?**

If Map extends Collection interface, then where are the elements? Map contains key-value pairs and it provides methods to retrieve list of Keys or values as Collection but it doesn’t fit into the “group of elements” paradigm.

* **What is difference between Enumeration and Iterator interface?**
* Enumeration is twice as fast as Iterator and uses very less memory. Enumeration is very basic and fits to basic needs. But Iterator is much safer as compared to Enumeration because it always denies other threads to modify the collection object which is being iterated by it.
* Iterator takes the place of Enumeration in the Java Collections Framework. Iterators allow the caller to remove elements from the underlying collection that is not possible with Enumeration. Iterator method names have been improved to make its functionality clear.
* **What is difference between Stack and Queue?**
* Both Stack and Queue are used to store data before processing them.

java.util.Queue is an interface whose implementation classes are present in java concurrent package. Queue allows retrieval of element in First-In-First-Out (FIFO) order but it’s not always the case. There is also Deque interface that allows elements to be retrieved from both end of the queue.  
 Stack is similar to queue except that it allows elements to be retrieved in Last-In-First-Out (LIFO) order. Stack is a class that extends Vector whereas Queue is an interface.

* **What is difference between Comparable and Comparator interface?**

Comparable is meant for objects with natural ordering which means the object itself must know how it is to be ordered. For example Roll Numbers of students. Whereas, Comparator interface sorting is done through a separate class.

Logically, Comparable interface compares “this” reference with the object specified and Comparator in Java compares two different class objects provided.

If any class implements Comparable interface in Java then collection of that object either List or Array can be sorted automatically by using Collections.sort() or Arrays.sort() method and objects will be sorted based on there natural order defined by CompareTo method.

To summarize, if sorting of objects needs to be based on natural order then use Comparable whereas if you sorting needs to be done on attributes of different objects, then use Comparator in Java.

* **How HashMap works in Java?**

**On principle of Hashing**

**Hashing** in its simplest form, is a way to assigning a unique code for any variable/object after applying any formula/algorithm on its properties. A true Hashing function must follow this rule:

Hash function should return the same hash code each and every time, when function is applied on same or equal objects. In other words, two equal objects must produce same hash code consistently.

**Multi-Threading and Concurrency:**

* **What is the difference between Process and Thread?**
* A process is a self-contained execution environment and it can be seen as a program or application whereas Thread is a single task of execution within the process. Java runtime environment runs as a single process which contains different classes and programs as processes. Thread can be called lightweight process. Thread requires less resources to create and exists in the process, thread shares the process resources.
* **What are the benefits of multi-threaded programming?**

In Multi-Threaded programming, multiple threads are executing concurrently that improves the performance because CPU is not idle in case some thread is waiting to get some resources. Multiple threads share the heap memory, so it’s good to create multiple threads to execute some task rather than creating multiple processes. For example, Servlets are better in performance than CGI because Servlet support multi-threading but CGI doesn’t.

* **What is difference between user Thread and daemon Thread?**
* When we create a Thread in java program, it’s known as user thread. A daemon thread runs in background and doesn’t prevent JVM from terminating. When there are no user threads running, JVM shutdown the program and quits. A child thread created from daemon thread is also a daemon thread.
* **How can we create a Thread in Java?**

There are two ways to create Thread in Java – first by implementing Runnable interface and then creating a Thread object from it and second is to extend the Thread Class.

* **What are different states in lifecycle of Thread?**

When we create a Thread in java program, its state is New. Then we start the thread that change its state to Runnable. Thread Scheduler is responsible to allocate CPU to threads in Runnable thread pool and Blocked and Dead. Read this post to learn more about life cycle of thread.

* **What is Thread Scheduler and Time Slicing?**
* Thread Scheduler is the Operating System service that allocates the CPU time to the available runnable threads. Once we create and start a thread, its execution depends on the implementation of Thread Scheduler. Time Slicing is the process to divide the available CPU time to the available runnable threads. Allocation of CPU time to threads can be based on thread priority or the thread waiting for longer time will get more priority in getting CPU time. Thread scheduling can’t be controlled by java, so it’s always better to control it from application itself.
* **What is context-switching in multi-threading?**
* Context Switching is the process of storing and restoring of CPU state so that Thread execution can be resumed from the same point at a later point of time. Context Switching is the essential feature for multitasking operating system and support for multi-threaded environment.
* **How does thread communicate with each other?**

When threads share resources, communication between Threads is important to coordinate their efforts. Object class wait(), notify() and notifyAll() methods allows threads to communicate about the lock status of a resource.

* **Why thread communication methods wait(), notify() and notifyAll() are in Object class?**

In Java every Object has a monitor and wait, notify methods are used to wait for the Object monitor or to notify other threads that Object monitor is free now. There is no monitor on threads in java and synchronization can be used with any Object, that’s why it’s part of Object class so that every class in java has these essential methods for inter thread communication.

* **Why wait(), notify() and notifyAll() methods have to be called from synchronized method or block?**

When a Thread calls wait() on any Object, it must have the monitor on the Object that it will leave and goes in wait state until any other thread call notify() on this Object. Similarly when a thread calls notify() on any Object, it leaves the monitor on the Object and other waiting threads can get the monitor on the Object. Since all these methods require Thread to have the Object monitor, that can be achieved only by synchronization, they need to be called from synchronized method or block.

* **Why Thread sleep() and yield() methods are static?**
* Thread sleep() and yield() methods work on the currently executing thread. So there is no point in invoking these methods on some other threads that are in wait state. That’s why these methods are made static so that when this method is called statically, it works on the current executing thread and avoid confusion to the programmers who might think that they can invoke these methods on some non-running threads.
* **How can we achieve thread safety in Java?**
* There are several ways to achieve thread safety in java – synchronization, atomic concurrent classes, implementing concurrent Lock interface, using volatile keyword, using immutable classes and Thread safe classes.
* **What is Deadlock? How to analyze and avoid deadlock situation?**

Deadlock is a programming situation where two or more threads are blocked forever, this situation arises with at least two threads and two or more resources.

Avoid Nested Locks, Lock Only What is Required and Avoid waiting indefinitely are common ways to avoid deadlock situation, read this post to learn how to analyze deadlock in java with sample program.

* **What is Java Timer Class? How to schedule a task to run after specific interval?**
* java.util.Timer is a utility class that can be used to schedule a thread to be executed at certain time in future. Java Timer class can be used to schedule a task to be run one-time or to be run at regular intervals.
* java.util.TimerTask is an abstract class that implements Runnable interface and we need to extend this class to create our own TimerTask that can be scheduled using java Timer class.
* **What is Thread Pool? How can we create Thread Pool in Java?**
* A thread pool manages the pool of worker threads, it contains a queue that keeps tasks waiting to get executed.
* A thread pool manages the collection of Runnable threads and worker threads execute Runnable from the queue.
* java.util.concurrent.Executors provide implementation of java.util.concurrent.Executor interface to create the thread pool in java. Thread Pool Example program shows how to create and use Thread Pool in java. Or read ScheduledThreadPoolExecutor Example to know how to schedule tasks after certain delay.