Day1

What is JDK,JRE,or JVM?

JDK stands for java development kit. It includes a java runtime environment(JRE), loader, a compiler, a jar, and documentation in the generator.

JRE stands for java runtime environment which includes java virtual machines. It includes a Java virtual machine and library. JRE provides the runtime environment for which java bytecode can be executed.

JVM stands for java virtual machine. It is an abstract machine that provides a runtime environment in which java bytecode can be executed.

What are the other ways to create an object other than creating a new object?

new keyword,

newInstance();

clone();

deserialization();

using newInstance() of constructor class.

Explain java.lang.OutOfMemoryError

**runtime error in Java** which occurs when the Java Virtual Machine (JVM) is unable to allocate an object due to insufficient space in the Java heap

**Is JVM a compiler or interpreter ?**

**Java virtual machines are interpreters and compilers. It interprets source code into bytecode and compiles and executes bytecode.**

**Difference between loadClass and Class.forName ?**

**Class.forName:it initializes the class by calling the static block of the class.**

**loadClass:it will not call the static block of the class for initialization.**

**Should we override the finalize method ?**

**yes we should override the finalize method, because sometimes our class has resources that garbage collection won't be able to clean up such as file handler and database connect.**

**Which kind of memory is used for storing object member variables and function local variables ?**

**local variables are stored in stack memory.**

**object member variables are stored in heap memory.**

**Why do member variables have default values whereas local variables don't have any default value ?**

**Because member variables are loaded into heap memory, they are initialized once the instance of a class object is created; in case of the local variables, it is stored in stack memory until they are being used.**

**Why Java don't use pointers ?**

**Because the pointer is not safe, by directly using the pointer to allocated main memory, in java it is done by jvm automatically. pointer is used to access the main memory.**

**What are various types of Class loaders used by JVM ?**

**There are 3 types of class loaders.**

**1)bootstrap class loader**

This is the first class loader which is the super class of the extension classloader, The bootstrap class loader loads the core Java libraries

**2)extension class loader**

This is the sub class loader of the bootstrap class loader and it is the super class of the system classloader. The Java Class Loader is a part of the [Java Runtime Environment](https://en.wikipedia.org/wiki/Java_Runtime_Environment) that [dynamically loads](https://en.wikipedia.org/wiki/Dynamic_loading) [Java classes](https://en.wikipedia.org/wiki/Java_class) into the [Java Virtual Machine](https://en.wikipedia.org/wiki/Java_Virtual_Machine).[[1]](https://en.wikipedia.org/wiki/Java_Classloader#cite_note-1) Usually classes are only loaded [on demand](https://en.wikipedia.org/wiki/Lazy_initialization).

**3)System class loader**

**This is the child class of the extension class loader; it loads the class file from the classpath.**

**How are Classes loaded by JVM?**

**In JVM the classes only load classes when they are needed.**

**Which memory areas do instance and static variables use ?**

**Instance variables using heap memory once an instance of the class is created.**

**Static variables memory allocation for static variables only happens once the class is being loaded by the class loader in memory.**

**What is PermGen or Permanent Generation ?**

**It is a special type of heap space. It is separate from the main memory (heap). JVM uses PermGen to keep track of loaded class metadata. All the static content is stored by the** [**JVM**](https://www.javatpoint.com/jvm-java-virtual-machine) **in this memory section.**

**What is metaspace ?**

**It has replaced the older PermGen memory space, to keep track of loaded class metadata.**

**What is the difference between metaspace and permGen?**

**Metaspace by default auto increases its size, but you cannot make PermGen auto-increase.**

**PermGen always has a fixed maximum size. You can set a fixed maximum for Metaspace with JVM parameters, but you cannot make PermGen auto-increase,**

**Describe what happens when an object is created in Java.**

1. **Memory is allocated.**
2. **Fields are** [**initialized to their default values**](http://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.12.5)**.**
3. **The "first line" of the chosen constructor is invoked, unless it's an Object. By first line I mean** [**either explicit call to super() or this(), or an implicit call to super()**](http://docs.oracle.com/javase/specs/jls/se7/html/jls-8.html#jls-8.8.7)**.**
4. **The** [**instance initializer**](http://docs.oracle.com/javase/specs/jls/se7/html/jls-8.html#jls-8.6) **is executed and the** [**fields are initialized to their requested values**](http://docs.oracle.com/javase/specs/jls/se7/html/jls-8.html#jls-8.3.2) **(actually field initialization is usually compiled as an inline part of the instance initializer).**
5. **The rest of the constructor code is executed**

**Different types of memory used by JVM ?**

1. **Class(Method) Area**
2. **Heap**
3. **Stack**
4. **Program Counter Register**
5. **Native Method Stack**

**Why is a static block executed before the main method ?**

**Static blocks are executed before the main method because sometimes we want to do some operations before the main method creates an object of the class, for example, we want to have a database connection before any class is being created.Static blocks are executed when the class is loaded in memory(JVM).**

**Can we override the static method ?**

**No we can not override the static method, because method overriding is based on dynamical binding at runtime, where the static method is bound using static binding at compile time. so we can not override the static method.**

**Can we overload a static method in Java?**

**Yes we can overload a static method in java**

**Can we override a private method in Java?**

**no, we can not override a private method in java, because private access modifiers only allow access in the same class.**

**What is method hiding in Java?**

**When a child class defines a static method with the same signature as a static method in the parent class, then the child's method *hides* the one in the parent class.**

**Why use this keyword**

**This keyword is referred to in the same class, the purpose of using this keyword is to differentiate the parameter variable and class attributes.**

s.input(m,n); //actual parameter or argument

public void input(int a, int b)//formal parameter.

**When Need of super keyword ?**

**is to differentiate the super class and subclass, most of the time we use the super keyword to call the parent method or constructor.**

**What is the difference between this. (this dot) and this() (this off).**

**This dot is used to refer to member variables; this() refers to the constructor in the same class.**

**Does Java Pass by Value or Pass by Reference?**

**Java is pass by value, in Java we don’t have pass by reference.**

**What is a Constructor?**

**Constructor is used to initialize the class member, special method, same name as class name, can have a default constructor or with argument constructor.**

**Do we have a Copy Constructor in Java?**

**Yes, we do have a copy constructor is a constructor creates an object using another object of the same java class.**

**What is Constructor Chaining ?**

**Constructor chaining means the process of calling one constructor from another constructor with respect to the current object.**

**Can we call a subclass constructor from a superclass constructor?**

**no, we can't call a subclass constructor from the parent class.**

**What is No-arg constructor?**

**A constructor that has no parameter is known as the default constructor**

**How is a no – argument constructor different from a default Constructor?before java 7?**

**When we write a class without any constructor(not declared explicitly) then at compilation time java compiler creates a default constructor in our class**

**What are private constructors and where are they used?**

**used in restricting object creation. It is a special instance constructor used in static member-only classes**

**When do we need Constructor Overloading?**

**If we want to have different ways of initializing an object using a different number of parameters,**

**Do we have destructors in Java?**

**There is no concept of destructor in Java. In place of the destructor, Java provides the garbage collector that works the same as the destructor**

**What are native methods? in java**

**The method is implemented in "native" code. That is, code that does not run in the JVM.**

**================================================================**

**Day2**

**OPPs question:**

**Explain these OOPS concepts :**

**Polymorphism**

**In java polymorphism means multiple form, in simple words, one interface for many implementations.**

**There are two types of polymorphism,**

**1)Compile time polymorphism**

**compile-time polymorphism is method overloading(also known as method static binding or early binding.)**P3 p=new P3();

p.displayP(); //static binding or early binding

//compiletime binding

**Upcasting: Upcasting is the** [**typecasting**](https://www.geeksforgeeks.org/type-conversion-java-examples/) **of a child object to a parent object. Upcasting can be done implicitly.**

**Downcasting: Similarly, downcasting means the typecasting of a parent object to a child object.**

**2)Runtime polymorphism**

**runtime polymorphism is method overriding(also known as dynamic binding or late binding.)**

P3 pp=new C3();

pp.displayP();

pp.m1(); //late binding or dynamic binding or runtime binding

//runtime polymorphism

**Inheritance(extends)**

**Inheritance in java is the concept that properties of one class are inherited by another class. It helps to reuse the code and establish the relationship between two different classes. In inheritance there is a common class known as parent, super, and base class and child class or derived class or more specific class.**

**What are the types of Inheritance?**

* **Single Inheritance.**
* **Multiple Inheritance.(not available in Java) one child inherited 2 or more parents**
* **Multi-Level Inheritance.**
* **Hierarchical Inheritance. 2 children inherited from same parents**
* **Hybrid Inheritance.**

diamond problem - using default method - how to avoid that

.super.specific method name

**Abstraction**

**Basically hiding the implementation. Hide the details of something, and only show the functionality to the user.Thus you can say that abstraction in Java is the process of hiding the implementation details from the user and revealing only the functionality to them.**

**Encapsulation**

**Encapsulation is the mechanism where you bind your data and method as a single unit.( or binding properties with functionality) Here the data is hidden from the outside world. only able to access it through the current class method. This helps to protect the data from unnecessary modification. we can achieve this by:**

**Defined data field(class variable) as private**

**Implement public the setter and getter methods.**

**Interface(Implements)**

**Is the Blueprint of a class, or we can say is a collection of abstract methods. but it does not contain a constructor and all methods inside of the interface are body implemented.**

**concrete methods**

**method with body**

**What is the difference between interface and abstraction?**

**An abstract class allows you to create functionality that subclasses can implement or override. An interface only allows you to define functionality, not implement it. And whereas a class can extend only one abstract class, it can take advantage of multiple interfaces.**

**Association(Has-a relationship)**

**Relationships exist between two classes.**

**OnetoOne, OnetoMany,ManytoOne, ManytoMany.**

**When a relationship is established. Containing object(Object that has another object as member) communicates with contained objects to use their properties or characteristics.**

**Composition**

**Composition is known as a strong association. Composition is tily couple mean When two objects are bounded. When one object is destroyed another object will also get destroyed.**

**In composition, one class contains another class, which contained class depends on containing class in such a way that it can not exist independently. Allow all 4 relationships. static or compile time binding. When we use it we don’t want to add more features to the existing project, because when we change one class it can affect other classes as well.**

Composition allows us to reuse the existing code thereby reducing the code complexity and also reduces the bugs in the code. Using composition we can reuse only what we need from the object and also control its visibility.

**Aggreagation**

**Aggregation is known as a weak association. In aggregation, a child class can exist independently without a parent class. child classes have their own ownership. one object gets destroyed another will not get destroyed.**

**only allow OnetoOne relationship, dynamic binding and run time.**

**What is the default class modifier?**

**When no modifier is defined explicitly.**

**What are the different method access modifiers?**

**public, private, default, protected**

**What is the use of a final modifier in a class?**

**using final modifier on class is to make sure class can not be inherited**

**What is the use of a final modifier on a method?**

**using the final modifier on the method is to make sure the method is not able to overrider.**

**What is the use of a final modifier on a variable?**

**When a variable is declared with the final keyword, its value can’t be modified, essentially, a constant. This also means that you must initialize a final variable.**

**Which package is always imported by default?**

**java.lang package**

**Can I import the same package/class twice? Will the JVM load the package twice at runtime?**

**Yes, we can import the same package twice, but JVM loads the package only once.**

**Does importing a package imports the sub packages as well? E.g. Does importing com.bob. also import com.bob.code.?**

**No you will have to import the sub packages explicitly**

**What is a Java package and how is it used?**

**Making searching/locating and usage of classes, interfaces, enumerations and annotations easier.**

**Explain the usage of Java packages.???**

**We have two types of packages in Java:**

* **built-in packages**
* **the packages we can create (also known as user defined packages)**

**Are the imports checked for validity at compile time?**

**yes, check semantic validity.**

**Difference between Public, Private, Default and Protected ?**

**default:within the package only**

**Can we reduce the visibility of the overridden method ?**

**No, private methods can not be overridden.**

**Which access specifier can be used with Class ?**

**public, private, protected, default.**

**Can we reduce the visibility of the inherited or overridden method ?**

**No, we can not reduce the visibility of the inherited or overridden method.**

**What will happen if we make the constructor private ?**

**Yes we can, but we can only create objects within this class. You cannot access its objects from outside the constructor class. if all methods are static we can use a private constructor,**

**Can we instantiate the object of the derived class if the parent constructor is protected ?**

**Yes, protected access is in a subclass.**

**Can we declare an abstract method private ?**

**Private methods are not polymorphic, this makes no sense to make private method as abstract**

**What is the strictfp keyword and when do you use it?**

**strictfp is a modifier that stands for strict floating-point. It is used in java for restricting floating-point calculations and ensuring the same result on every platform while performing operations in the floating-point variable.**

**What is an instanceOf operator?**

***instanceof is* a binary operator used to test whether the object is an instance of the given type.**

**Day3**

**What is String in Java? String is a data type?**

**String in java is a sequence of characters. String is a reference data type.**

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

**String str= “name”;**

**String str2 = new String(“name2”);**

**How can we make String uppercase or lowercase?**

**String str = “abc”;**

**str.toUpperCase(); str.toLowerCase();**

**What is the String subSequence method?**

**Is a built-in function that returns charSequences.**

**How to compare two Strings in a java program?**

**String overrides equal() to compare the content of two strings.**

**== or compareTo**

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

**Character.toString(ch);**

**String.valueOf(ch);**

**-----------------------**

**String s=String.valueOf(c);**

**String s=Character.toString(c);**

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

**byte[] byteArr = str.getBytes();**

**Can we use String in a switch case?**

**Yes, we can**

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

**String is immutable and thread safe,**

**StringBuffer is thread-safe, slow, mutable.**

**string builder, mutable, non-thread safe and faster performer.**

**Why are strings immutable?**

**because sync**

**and a single object is cast into a string pool.**

**How to make class as immutable?**

* **The class must be declared as final so that child classes can’t be created.**
* **Data members in the class must be declared private and final**
* **provide a deep copy of an object for all data fields, so data members can’t be modified with an object reference.**
* **can not have a setter method. for a modified data field.**
* **Perform** [**cloning**](https://www.journaldev.com/60/java-clone-object-cloning-java) **of objects in the getter methods to return a copy rather than returning the actual object reference**

**How to Split String in java?**

**We could use the split() method which provides some delimiter to split string into a string array.**

**Why Char array preferred over String for storing passwords?**

**Since Strings are immutable there is no way the contents of Strings can be changed because any change will produce a new String, while if you char[] you can still set all its elements as blank or zero. So Storing the password in a character array clearly mitigates(减轻 )/reduce security risk of stealing passwords.**

**What is a String Pool?**

**String pool is in heap memory and it stores string literal**

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

**The method intern() creates an exact copy of a String object in the heap memory and stores it in the String constant pool. Note that, if another String with the same contents exists in the String constant pool, then a new object won't be created and the new reference will point to the other String**

**I am performing lots of string concatenation and string modification in my code. which class among string, StringBuffer and StringBuilder improves the performance of my code. Remember I also want thread safe code?**

**string buffer**

**What is a string intern?**

**string intern() method is used to return a string from a string pool.**

**for example: we have String declared as String str=new String(“name”);**

**str.intern();//return name from string constant pool**

**In how many ways you can create string objects in java?**

**string literal and new keyword**

**How does the substring method work in Java?**

**Returns a new string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index endIndex – 1**

**Difference between using StringTokenizer and String.split( )**

**StringTokenizer does not support regular expression, split string into tokens one at a time. don’t handle empty strings.**

**split() . split string into string array.**

**What is a String Tokenizer and its usage?**

**to split string into tokens based on the provided delimiter.**

**Methods in stringtokenizer class**

**booleanhasMoreTokens()**

**String nextToken()**

**String nextToken(String delim)**

**Day4 Exception**

### **What is an Exception in Java?**

**An exception is an error event that can happen during the execution of a program and disrupts its normal flow. The exception can arise from different kinds of situations such as wrong data entered by the user, hardware failure, network connection failure, etc.**

**Java Exception handling framework is used to handle runtime errors only, compile-time errors are not handled by exception handling framework**

### **What are the Exception Handling Keywords in Java?**

**throw**

**throws**

**try-catch**

**finally.**

### **Explain Java Exception Hierarchy?**

**Throwable is the parent class of the Java Exceptions Hierarchy and it has two child objects – Error and Exception. Exceptions are further divided into checked exceptions and unchecked exceptions.**

### **What are the important methods of Java Exception Class?**

### **Explain Java 7 ARM Feature and multi-catch block?**

### **multi-catch block where we can catch multiple exceptions in a single catch block. vertical slash.**

### **catch(IOException | SQLException | Exception ex){**

### **logger.error(ex);**

### **throw new MyException(ex.getMessage());**

### **}**

Most of the time, we use finally block just to close the resources and sometimes we forget to close them and get runtime exceptions when the resources are exhausted

### **What is the difference between Checked and Unchecked Exceptions in Java?**

1. **Checked Exceptions should be handled in the code using try-catch block or else the method should use the throws keyword to let the caller know about the checked exceptions that might be thrown from the method. Unchecked Exceptions are not required to be handled in the program or to mention them in the throws clause of the method.**
2. **Exception is the superclass of all checked exceptions whereas RuntimeException is the superclass of all unchecked exceptions. Note that RuntimeException is the child class of Exception.**

### **What is the difference between the throw and throws keyword in Java?**

**throws keyword is used with method signature to declare the exceptions that the method might throw whereas**

**throw keyword is used to disrupt the flow of the program and hand over the exception object to runtime to handle it。**

### **How to write custom exceptions in Java?**

**We can extend the Exception class or any of its subclasses to create our custom exception class. The custom exception class can have its own variables and methods that we can use to pass error codes or other exception-related information to the exception handler.**

### **What are the different scenarios causing “Exception in thread main”?**

**Exception in thread main**

**java.lang.UnsupportedClassVersionError**

**java.lang.NoClassDefFoundError**

**java.lang.NoSuchMethodError**

**java.lang.ArithmeticException**

### **What is the difference between final, finally, and finalize in Java?**

**final and finally are keywords in java whereas finalize is a method.**

**final keyword can be used with class variables so that they can’t be reassigned, with the class to avoid extending by classes and with methods to avoid overriding by subclasses, finally keyword is used with try-catch block to provide statements that will always get executed even if some exception arises, usually finally is used to close resources. finalize() method is executed by Garbage Collector before the object is destroyed, it’s a great way to make sure all the global resources are closed.**

### **What happens when an exception is thrown by the main method?**

**When an exception is thrown by a main() method, Java Runtime terminates the program and prints the exception message and stack trace in the system console.**

**Can we have an empty catch block?**

**We can have an empty catch block but it’s an example of bad programming. We should never have an empty catch block because if the exception is caught by that block, we will have no information about the exception and it wil be a nightmare to debug it.**

**Provide some Java Exception Handling Best Practices?**

**Use multi-catch block for cleaner close,Use Specific Exceptions for ease of debugging,Use custom exceptions to throw a single type of exception from your application API**

**Day5**

**What is an IO stream?**

**IO stream is process input and produce output, Stream is sequence of byte data.**

**What is the necessity of two types of streams – byte streams and character streams?**

**byte streams mainly perform on input and output 8 bit byte stream**

**character streams mainly perform on input and output for 16 unicode streams.**

**What are the super most classes of all streams?**

**InputStream for input and OutputStream for output**

**Input and Output Stream**

**What are FileInputStream and FileOutputStream?**

**FileOutputStream writes byte and character data into a file.**

**and FileInputStream reads bytes and characters from a file.**

**What is the functionality of SequenceInputStream?**

**What is PrintStream and PrintWriter?**

**printStream is to write data to another stream, automatically flush();**

**printWriter is used to print formatted object representation to txt output**

**PrintStream**

**Which streams are advised to use to have maximum performance in file copying?**

**BufferedInputStream and BufferedOutputStream**

**What is File class?**

**What is RandomAccessFile?**

**Creates a random access file stream to read from, and optionally to write to**

**What is the difference between the Reader/Writer class hierarchy and the InputStream/OutputStream class hierarchy?**

**What is Scanner class used for ?**

**Scanner is used to read input from the keyboard and break input into tokens using a delimiter.**

**Tell me something about BufferedWriter ? What are flush() and close() used for ?**

**Which class is used to read streams of characters from a file?**

**fileReader**

**Which class is used to read streams of raw bytes from a file?**

**FileInputStream Class**

**Difference between Scanner and BufferedReader ? Which one is faster and Why ?**

**BufferedReader is a bit faster as compared to scanner because scanner does parsing of input data and BufferedReader simply reads sequence of characters**

**How and when the Buffer is cleared when using Bufferedwriter Classes ?**

**Why is it important to close files and streams ?**

**to release file descriptor held by this class**

**Day6**

After starting a thread we are not allowed to restart the

same thread once again otherwise we will get runtime exception

**What is thread?**

Multithreading in [Java](https://www.javatpoint.com/java-tutorial) is a process of executing multiple threads simultaneously. A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

However, we use multithreading rather than multiprocessing because threads use a shared memory area. They don't allocate a separate memory area so saves memory, and context-switching between the threads takes less time than process.

class Thread

{

Public void start(){

1. Register this thread with thread scheduler

2. run()

}}

**What is the difference between Process and Thread?**

**thread is lightweight and process is not. One operating system can have multiple processes and in each process can have multiple threads. threads are using shared memory compared to processes which are not. threads don't allocate separate memory. Threading saves time. Thread doesn’t affect other threads. thread work independently**

**Important method inside of multithread**

* public void run(): is used to perform action for a thread(thread job).
* public void start(): starts the execution of the thread.JVM calls the run() method on the thread.
* public void sleep(long milliseconds): Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.
* public void join(): waits for a thread to die.
* public void join(long milliseconds): waits for a thread to die for the specified milliseconds.
* yield(): to pause currently executing thread for giving chance to remaining waiting

thread of the same priority.

**What are the benefits of multi-threaded programming?**

**multi-thread perform faster and save memory, in which threads share memory we don’t allocate separate memory.**

**What is the difference between user Thread and daemon Thread?**

**JVM doesn’t wait for the daemon thread to finish but it waits for the User Thread.**

**The User threads are high priority as compared to the daemon thread.**

**How can we create a Thread in Java?**

**There are two ways to create threads in Java.**

**1)By extending thread class.**

**2)By implementing a runnable interface.**

**What are the different states in the life cycle of Thread?**

**There are 5 states of thread class,**

**new-> runnable->no-runnable>running->terminated.**

**Can we call the run() method of a Thread class?**

**Yes we can, but the run() method will run as normal.**

class Thread

{

Public void start(){

1. Register this thread with thread scheduler

2. run()

}}

**How can we pause the execution of a Thread for a specific time?**

**Yes, we can use the built in method sleep() to make the thread pause for millions of seconds.**

**What do you understand about Thread Priority?**

**There are 3 types of priority: 1 is least and 10 is highest.**

Thread.MIN\_PRIORITY====>1

Thread.NORM\_PRIORITY====>5

Thread.MAX\_PRIORITY====>10

**Higher priority threads will execute first, thread scheduler will be based on priority.**

**Difference between Runnable vs Thread?**

Runnable is the preferred way to.

* **Java only supports single inheritance, so you can only extend one class.**
* **Implementing Runnable makes your class more flexible. If you extend Thread then the action you’re doing is always going to be in a thread. However, if you implement Runnable it doesn’t have to be**

**What is Thread Scheduler and Time Slicing?**

**The Thread Scheduler in java is the part of the JVM that decides which thread should run. no guarantee which thread will be chosen to run by thread scheduler**

**What is context-switching in multi-threading?**

**This is the process of storing the CPU state so that thread execution can be resumed from the same pointer at a later point of time.**

**How can we make sure main() is the last thread to finish in a Java Program?**

**We could use the thread join() method to make sure all the threads created by the program are dead before finishing the main function.**

**What if we call the run() method directly instead of the start() method?**

**Each thread starts in a separate call stack.**

**Invoking the run() method from the main thread, the run() method goes onto the current call stack rather than at the beginning of a new call stack.**

**How do threads communicate with each other?**

**threads communicate with each other using three methods, wait(), notifly(),notifyAll().**

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

**all are applied to the thread. If these methods are in the thread class, we are not able to apply them to the thread class. So in order to use those methods in a thread we have these 3 methods in object class.So these three methods at object class can be applied to different thread classes. They also need separate instances of class.**

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

**IllegalMonitorStateException**

**if we call one of the *wait()*, *notify(),* or *notifyAll()* methods of the *Object* class outside of a** [***synchronized***](https://www.baeldung.com/java-synchronized) **block**

**Why Thread sleep() and yield() methods static?**

**The sleep() and yield() methods work with the current execute thread. so there is no point for invoking another thread.**

**How can we achieve thread safety in Java?**

**We can make a synchronize method or block , using volatile keywords and final.**

**What is a volatile keyword in Java**

**volaties are used to modify the value of a variable from main memory by different threads. Also make class thread safe.**

**Which is more preferred – Synchronized method or Synchronized block?**

**Synchronized block**

**How to create a daemon thread in Java?**

**Creating a thread as a daemon in Java is as simple as calling the setDaemon() method. A setting of true means the thread is a daemon; false means it is not. By default, all threads are created with an initial value of false.**

**What is ThreadLocal?**

**allows us to store data that will be accessible only by a specific thread.**

**What is a Thread Group? Why is it advised not to use it?**

**ThreadGroup is a class which is intended to provide information about a thread group. The ThreadGroup API is weak and it doesn’t have any functionality that is not provided by Thread. It has two main features – to get the list of active threads in a thread group and to set the uncaught exception handler for the thread. But Java 1.5 has added the setUncaughtExceptionHandler*(UncaughtExceptionHandler eh)* method using which we can add an uncaught exception handler to the thread. So ThreadGroup is obsolete and hence not advised to use anymore.**

**What is Java Thread Dump, How can we get Java Thread dump of a Program?**

**A thread dump is a list of all the threads active in the JVM, thread dumps are very helpful in analyzing bottlenecks in the application and analyzing deadlock situations. There are many ways using which we can generate Thread dump – Using Profiler, Kill -3 command, jstack tool,**

**What is Deadlock? How to analyze and avoid a 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**

**To analyze a deadlock, we need to look at the java thread dump of the application, we need to look out for the threads with state as BLOCKED and then the resources it’s waiting to lock, every resource has a unique ID using which we can find which thread is already holding the lock on the object.**

**Avoid Nested Locks, Lock Only What is Required and Avoid waiting indefinitely are common ways to avoid deadlock situations, read this post to learn how to** [**analyze deadlock in java**](https://www.journaldev.com/1058/deadlock-in-java-example) **with a sample program.**

**What is Java Timer Class? How to schedule a task to run after a specific interval?**

**What is a Thread Pool? How can we create a Thread Pool in Java?**

**What will happen if we don’t override the Thread class run() method?**

**What is atomic operation? What are atomic classes in Java Concurrency API?**

**What is the Lock interface in Java Concurrency API? What are its benefits over synchronization?**

**What is the Executors Framework?**

**What is BlockingQueue? How can we implement Producer-Consumer problem using Blocking Queue?**

**What is Callable and Future?**

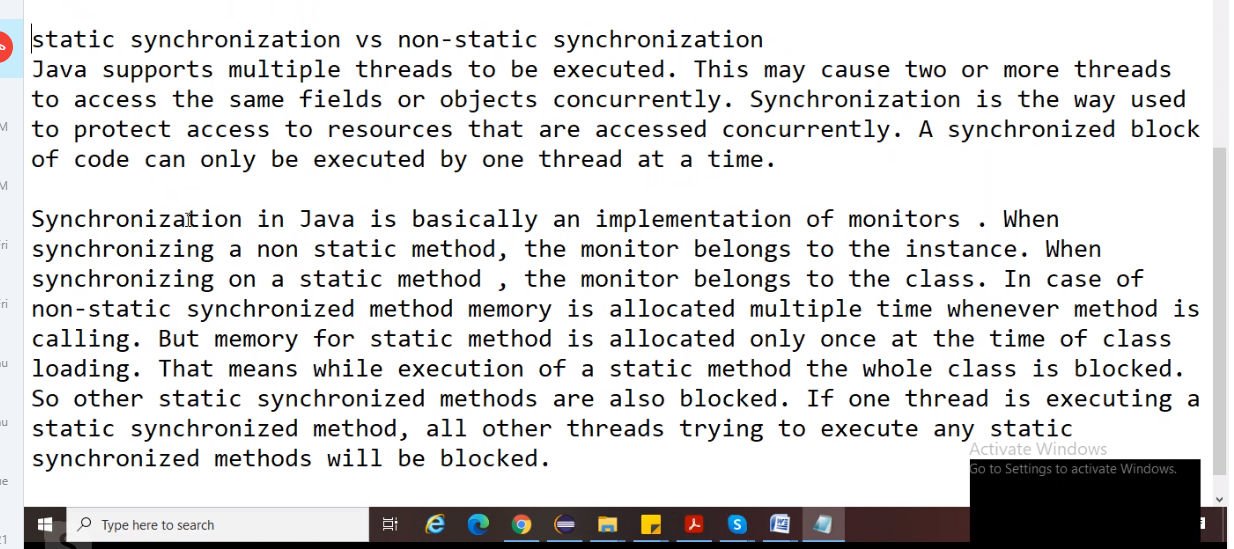
**What is the FutureTask class?**

**What are Concurrent Collection Classes?**

**What is Executors Class?**

**What are some of the improvements in Concurrency API in Java 8?**

**Object Class notify,notifyAll,wait ?**



# **Object level lock vs Class level lock in Java**

**Deadlock: in Synchronized program some methods are waiting for each other to release the resource**

### How to avoid ConcurrentModificationException in a multi-threaded environment?

To avoid the ConcurrentModificationException in a multi-threaded environment, we can follow the following ways-

1. Instead of iterating over the collection class, we can iterate over the array. In this way, we can work very well with small-sized lists, but this will deplete the performance if the array size is very large.
2. Another way can be locking the list by putting it in the synchronized block. This is not an effective approach as the sole purpose of using multi-threading is relinquished by this.
3. JDK 1.5 or higher provides ConcurrentHashMap and CopyOnWriteArrayList classes. These classes help us in avoiding concurrent modification exceptions.

### How to avoid ConcurrentModificationException in a single-threaded environment?

By using the iterator's remove() function, you can remove an object from an underlying collection object.

# **Java Thread Pool**

Java Thread pool represents a group of worker threads that are waiting for the job and reuse many times. I

#### Advantage of Java Thread Pool

Better performance It saves time because there is no need to create new thread.

Disadvantages of Java Thread Pool

Threadpool threads are **not suitable for long running operations**, as it can easily lead to thread starvation. thread starvation in java when we in thread pool

Starvation describes **a situation where a thread is unable to gain regular access to shared resources and is unable to make progress**. ... If one thread invokes this method frequently, other threads that also need frequent synchronized access to the same object will often be blocked

ExecutorService

high-level API for concurrency(java.util.concurrent), One of the features of this API is the Executor interfaces that provide an alternative (better) way to launch and manage threads.

void shutdown()

List<Runnable> shutdownNow()

The shutdown() method tells the executor to stop accepting new tasks, but the previous tasks are allowed to continue until the finish. During this time, the method isTerminated() will return false until all tasks are completed, while the method isShutdown() will return true at all times.

The shutdownNow() method will also tell the executor to stop accepting new tasks but it will TRY to stop all executing tasks immediately (by interrupting the threads, but if the thread doesn't respond to interrupts, it may never terminate) and return a list of the tasks that were never started.

Synchronization: it is capability to control the access of multiple threads to any shared resources

Synchronization is used 1) to prevent the thread interference(once a thread is finished another thread will join)

2) to prevent inconsistency.

1.Synchronized method

2.Synchronized block

3.Static Synchronized

Deadlock: in Synchronized program some methods are waiting got each other to release the resource

Mutual Exclusive: It helps keep threads from interfering with one another while sharing the data.

Concept of Lock:

Synchronization is built around an internal entity known as lock (or moniter)

ReentrantLock

ReentrantLock belong to java.util.concurrent

Two key feature of ReentrantLock, which provides more control on lock acquisition is trying to get a lock with the ability to interrupt, and a timeout on waiting for a lock, these are key for writing responsive and scalable systems in Java.

1) Ability to lock interruptibly.

2) Ability to timeout while waiting for lock.

3) Power to create fair lock.

4) API to get list of waiting thread for lock.

5) Flexibility to try for lock without blocking.

## **Difference between ReentrantLock and synchronized keyword in Java**

| **Sr. No.** | **Key** | **ReentrantLock** | **Synchronized** |
| --- | --- | --- | --- |
| 1 | Acquire Lock | Reentrant lock class provides lock() methods to get a lock  on the shared resource by thread | You need to just write synchronized keyword to acquire a lock |
| 2 | Release Lock | To release lock , programmers have to call unlock() method | It is done implicitly |
| 3 | Ability to interrupt | lockInterruptibly() method can be used to interrupt the thread | There is no way to interrupt the thread |
| 4 | Fairness | Constructor of this class has fairness parameter. If it is set to true then locks favor granting access to the longest-waiting  \* thread | Lock does not guarantee any particular  access order |
| 5 | Lock Release Order | Lock can be released in any order | Lock should be released in the same order in which they were acquired |

Daemon Thread in Java

Daemon thread in java is a service provider thread that provides services to the user thread. Its life depends on the mercy of user threads i.e. when all the user threads dies, JVM terminates this thread automatically.

Volatile keyword:

A volatile keyword is used to modify the value of a variable by different threads.

Difference between synchronization and volatile keyword

Volatile keyword is not a substitute for a synchronized keyword, but it can be used as an alternative in certain cases. There are the following differences are as follows:

assume that two threads are working on the same class. Both threads run on different processors where each thread has its local copy of var. If any thread modifies its value, the change will not reflect in the original one in the main memory. It leads to data inconsistency because the other thread is not aware of the modified value.

| **Volatile Keyword** | **Synchronization Keyword** |
| --- | --- |
| Volatile keyword is a field modifier. | Synchronized keyword modifies code blocks and methods. |
| The thread cannot be blocked for waiting in case of volatile. | Threads can be blocked for waiting in case of synchronized. |
| It improves thread performance. | Synchronized methods degrade the thread performance. |
| It synchronizes the value of one variable at a time between thread memory and main memory. | It synchronizes the value of all variables between thread memory and main memory. |
| Volatile fields are not subject to compiler optimization. | Synchronize is subject to compiler optimization. |

Important points

on Volatile keyword in Java

1. The volatile keyword in Java is only applied to a variable and using a volatile keyword with class and method is illegal.  
  
2. volatile keyword in Java guarantees that the value of the volatile variable will always be read from main memory and not from Thread's local cache.  
  
3. In Java reads and writes are [atomic](http://javarevisited.blogspot.sg/2012/02/what-is-race-condition-in.html) for all variables declared using Java volatile keyword (including long and double variables).  
  
4. Using the volatile keyword in Java on variables reduces the risk of memory consistency errors because any write to a volatile variable in Java establishes a happens-before relationship with subsequent reads of that same variable.  
  
5. From Java 5 changes to a volatile variable are always visible to other threads. What's more, it also means that when a thread reads a volatile variable in Java, it sees not just the [latest change to the volatile variable](http://java67.blogspot.sg/2012/08/what-is-volatile-variable-in-java-when.html) but also the side effects of the code that led up the change.  
  
6. Reads and writes are atomic for reference variables are for most primitive variables (all types except long and double) even without the use of volatile keyword in Java.  
  
7. Access to a volatile variable in Java never has a chance to block, since we are only doing a simple read or write, so unlike a synchronized block we will never hold on to any lock or wait for any [lock](http://javarevisited.blogspot.sg/2010/10/what-is-deadlock-in-java-how-to-fix-it.html).  
  
8. Java volatile variable that is an object reference may be null.  
  
9. Java volatile keyword doesn't mean atomic, its common misconception that after declaring volatile ++ will be atomic, to make the operation atomic you still need to ensure exclusive access using synchronized method or block in Java.  
  
10. If a variable is not shared between multiple threads, you don't need to use volatile keyword with that variable.

================================================================

Java 8

What are the features of Java 8?

Lambda expressions,

Functional interfaces,

Static methods,

deaflut method,

Method references,

Functional interfaces.(**Single Abstract Method interfaces**)

stream api,

Date Time API,

Functional interface

* As discussed above, only one abstract method is allowed in any functional interface. Second abstract method is not not permitted in a functional interface. If we remove @FunctionInterface annotation then we are allowed to add another abstract method, but it will make the interface non-functional interface.
* A functional interface is valid even if the @FunctionalInterface annotation would be omitted. It is only for informing the compiler to enforce single abstract method inside interface.
* Conceptually, a functional interface has exactly one abstract method. Since default methods have an implementation, they are not abstract. Since default methods are not abstract you’re free to add default methods to your functional interface as many as you like.
* **Predicate<T>**
* **Consumer<T>**
* **Function<T, R>**
* **Supplier<T>**
* **UnaryOperator<T>**

How to combine two Predicates? and, or, negate

Predicate<Integer> p1= x->(x>10);

Predicate<Integer> p2=x->(x%2==0);

System.out.println(p1.and(p2).text(8));

default Predicate<T> and(Predicate<? super T> other)  
default Predicate<T> or(Predicate<? super T> other)  
default Predicate<T> negate()

isEqual to compare two Predicates.

static <T> Predicate<T> isEqual(Object targetRef)

What is the difference between normal date time(Util) and sql date time?

1. sql date time is the child of java.util.date
2. sql is a wrapper class on the topic of util date time.
3. We use sql date time for Database.
4. If we use normal date time it will provide us date and time information all together, if we use sql date time it will only provide us date.

What is the difference between normal date time(Util) and Joda date time?

1. Normal date time thread safety is not there, it has multiple thread issues, data consistency not there. Joda is thread safety.
2. Normal date time is poorly designed with less library available, compared to Joda date time it provides you with a wide variety of libraries.
3. Joda date time facilitates us with zone date time which until date time doesn’t.

What are the types of functionality in stream api?

Intermediate operations:You can easily identify intermediate operations; they always return a new stream. This allows the operations to be connected. An important feature of intermediate operations is that they don't process the elements until a terminal operation is invoked.

filter(), map(), distinct(), sorted(), limit()

Terminal operations: You can also easily identify terminal operations, they always return something other than a stream.

forEach(), toArray(), reduce(), collect(), min(), max(), count(),

Difference between intermediate operation and terminal operations?

1. Terminal operations are when intermediate operations return a stream as a result and terminal operations return non-stream values like primitive or object or collection or may not return anything.
2. As intermediate operations return another stream as a result, they can be chained together to form a pipeline of operations. Terminal operations can not be chained together
3. Pipeline of operations may contain any number of intermediate operations, but there has to be only one terminal operation, that too at the end of the pipeline.
4. Intermediate operations are lazily loaded. When you call intermediate operations, they are actually not executed. They are just stored in the memory and executed when the terminal operation is called on the stream.

How to convert String, Collection, Array into stream?

First we have to cover it into stream first.

Array.asList(new Integer[]{12,3,4,5}).stream.forEach(System.out::println);

Strin s=”hello”;

s.chars().mapToObj(x->(char)x).forEach(System.out::println);

This operator is covered s into characters then mapping each element into x and type casting. We have a character type of string.

Remove duplicate string

s.chars().mapToObj(x-> (char)x).distinct.forEach(System.out::println);

Integer[] array = {1,2,3,4,5};

Stream.of(array).forEach(System.out::println);