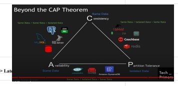
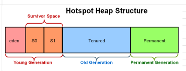
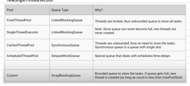
	1. OOPs 2. Encapsulation, Polymorphism(Dynamic Binding), overrriding, overloading JVM, JRE and JDK, BYTE CODE interface and abstract class Object class and its methods transient Serialization, thread Calonable 3. static 4. final and finally 5. Exception handling 6. Check for ++ and -- 7. voletile & transient. Servlet 1. Servlet LifeCycle 2. Difference between servlet context and servlet config 3. Difference betwwen GET and POST method, delete, put 4. How to set session timeout in session 5. Filters usage in application 6. What are the listeners in servlet 7. forward and redirect method Marshling and demarshling	011+91+(phone number)	
	What is Collection ? What is a Collections Framework ? What are the benefits of Java Collections Framework ?		
	What is the difference between Collection and Collections ?		
	How HashMap works in Java ?		
	How HashSet internally Works ?		
	What is the difference between Iterator and Enumeration ?		
	What is the difference between HashMap and Hashtable ?		
	What is the difference between LinkedList and ArrayList in Java ?		
	What are Comparable and Comparator interfaces ? List the difference between them ?		
	Why Map interface does not extend the Collection interface in Java Collections Framework ?		
	How to create immutable object ?		
	We have equals() to compare two methods then why comparator again ?		
	How to create Immutable Class and Object in Java?		
	What is difference between StringBuffer and StringBuilder in Java ?		
	System.Exit()		
	Can we declare a class as static?	https://www.journaldev.com/996/java-inner-class	
	Can we overload main method?		
	Can we have multiple public classes in a java source file?		
	What are access modifiers?		
	What is static import?		
	Java 7 feature is try-with-resources / AutoCloseable	https://www.journaldev.com/592/java-try-with-resources	
	What is Marker interface?	**A marker interface in Java is an interface that does not contain any methods or fields. Its primary purpose is to mark or tag a class with a particular characteristic or behavior. Marker interfaces provide metadata to the Java compiler and JVM, indicating that the objects of the class implementing the marker interface should be treated differently** **Serializable: Indicates that a class can be serialized.** **Cloneable: Indicates that a class allows field-for-field copying.**	
	What is Java Reflection API? Why it's so important to have?	https://www.journaldev.com/1789/java-reflection-example-tutorial	https://www.journaldev.com/2366/core-java-interview-questions-and-answers#reflection-api
	What is composition in java?		
	What is the benefit of Composition over Inheritance?		
	How to sort a collection of custom Objects in Java?		
	What is Classloader in Java?	**Bootstrap Class Loader** – It loads JDK internal classes. It loads rt. jar and other core classes for example java.lang.* package classes. **Extensions Class Loader** – It loads classes from the JDK extensions directory, usually $JAVA_HOME/lib/ext directory. **System Class Loader** – This classloader loads classes from the current classpath. We can set classpath while invoking a program using -cp or -classpath command line option.	https://www.journaldev.com/349/java-classloader
	Shellow & Deep Copy ?		
	What is instanceof keyword?		
	What is difference between Heap and Stack Memory?		
	Why String is immutable or final in Java	https://www.journaldev.com/1321/java-string-interview-questions-and-answers	
	Why Char array is preferred over String for storing password?	https://www.journaldev.com/1321/java-string-interview-questions-and-answers	
	What does String intern() method do?		
	Write code to interate map in Java 8.		
	Difference between Intermediate and terminal operations in Stream?	https://java2blog.com/java-8-interview-questions/#23_Difference_between_Intermediate_and_terminal_operations_in_Stream	
	LinkedHashSet vs TreeSet vs HashSet	https://javaconceptoftheday.com/hashset-vs-linkedhashset-vs-treeset-in-java/	
	Difference between HashMap, LinkedHashMap and TreeMap	https://www.geeksforgeeks.org/differences-treemap-hashmap-linkedhashmap-java/	
	Java - Runnable vs Callable	**Runnable** : If you have a fire and forget task then use Runnable. Put your code inside a Runnable and when the run() method is called, you can perform your task. The calling thread really does not care when you perform your task. **Callable** : If you are trying to retrieve a value from a task, then use Callable. Now callable on its own will not do the job. You will need a Future that you wrap around your Callable and get your values on future.get (). Here the calling thread will be blocked till the Future comes back with results which in turn is waiting for Callable's call() method to execute.	https://medium.com/javarevisited/java-runnable-vs-callable-786aa706775d
	Enumeration Vs Iterator In Java :		https://javaconceptoftheday.com/differences-between-enumeration-vs-iterator-in-java/
	Java 8 Stream and lambda	https://www.youtube.com/playlist?list=PLTyWtrsGknYdqY_7lwcbJ1z4bvc5yEEZl	
		https://www.youtube.com/playlist?list=PLqq-6Pq4lTTa9YGfyhyW2CqdtW9RtY-I3	
		https://www.youtube.com/playlist?list=PLsyeobzWxl7otduRddQWYTQezVul0xIX6	
		https://www.youtube.com/watch?v=CwOfjXPL6-Q	

Topic	Description	Reference
SOLID---- **1. Duplicacy** **2. Easy to maintain** **3. Easy to understand** **4. Flexiblity** **5. Reduce Complexity**	**Single Responsibility Principle (SRP)** : A class should have one and only one reason to change, meaning that a class should have only one job. **Open-Closed Principle (OCP)** : Objects or entities should be open for extension but closed for modification. **Liskov Substitution Principle (LSP) :** It applies to inheritance in such a way that the derived classes must be completely substitutable for their base classes. In other words, if class A is a subtype of class B, then we should be able to replace B with A without interrupting the behavior of the program. **Interface Segregation Principle (ISP) :** The principle states that the larger interfaces split into smaller ones. Because the implementation classes use only the methods that are required. We should not force the client to use the methods that they do not want to use. **Dependency Inversion Principle (DIP)** : The principle states that we must use abstraction (abstract classes and interfaces) instead of concrete implementations. High-level modules should not depend on the low-level module but both should depend on the abstraction.	https://www.javatpoint.com/solid-principles-java https://www.youtube.com/watch?v=yxf2spbpTSw https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design
JAVA 8	forEach() method in Iterable interface default and static methods in Interfaces Functional Interfaces and Lambda Expressions Java Stream API for Bulk Data Operations on Collections Java Time API Collection API improvements Concurrency API improvements Java IO improvements Miscellaneous Core API improvements	https://www.digitalocean.com/community/tutorials/java-8-features-with-examples
	Java 8 Stream Intermediate And Terminal Operations Intermediate Operations : map(), filter(), distinct(), sorted(), limit(), skip() Terminal Operations : forEach(), toArray(), reduce(), collect(), min(), max(), count(), anyMatch(), allMatch(), noneMatch(), findFirst(), findAny()	https://javaconceptoftheday.com/java-8-stream-intermediate-and-terminal-operations/
JAVA 11	Running Java File with single command **New utility methods in String class :** repeat(int), strip(), stripLeading(), stripTrailing(), isBlank(), lines() **Collection Enhancements in Java 11:** toArray() **Files :** readString(), writeString() Local-Variable Syntax for Lambda Parameters Nested Based Access Control JEP 321: HTTP Client JEP 328: Flight Recorder	https://www.digitalocean.com/community/tutorials/java-11-features
Collections		https://www.javatpoint.com/java-collections-interview-questions
Hashmap		https://howtodoinjava.com/java/collections/hashmap/how-hashmap-works-in-java/ https://www.youtube.com/watch?v=CojCE-ojdGY
ConcurrentHashMap		https://www.geeksforgeeks.org/concurrenthashmap-in-java/ https://www.youtube.com/watch?v=6Zzm4esAi7A
Volatile vs AtomicInteger		https://www.youtube.com/watch?v=WH5UvQJizH0
HashMap works in Java		https://www.youtube.com/watch?v=c3RVW3KGIIE
Design Patterns		https://www.javatpoint.com/design-patterns-in-java . https://www.youtube.com/playlist?list=PLsyeobzWxl7r2ZX1fl-7CKnayxHJA_1ol . https://www.youtube.com/watch?v=VGLjQuEQgkI&list=PLt4nG7RVVk1h9lxOYSOGI9pcP3I5oblbx&index=1
Transaction propagation and isolation	REQUIRED Propagation SUPPORTS Propagation MANDATORY Propagation NEVER Propagation NOT_SUPPORTED Propagation REQUIRES_NEW Propagation NESTED Propagation	https://www.baeldung.com/spring-transactional-propagation-isolation
Software Architectural Patterns	Layered pattern Client-server pattern Master-slave pattern Pipe-filter pattern Broker pattern Peer-to-peer pattern Event-bus pattern Model-view-controller pattern Blackboard pattern Interpreter pattern	https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013
DSA		https://www.youtube.com/playlist?list=PLUcsbZa0qzu3yNzzAxgvSgRobdUUJvz7p
Design Patterns	**Singleton Design Pattern :** **Factory Method Pattern :** A Factory Pattern or Factory Method Pattern says that just define an interface or abstract class for creating an object but let the subclasses decide which class to instantiate. In other words, subclasses are responsible to create the instance of the class. **Builder Design Pattern :** Builder Pattern says that "construct a complex object from simple objects using step-by-step approach" **Adapter Pattern** : converts the interface of a class into another interface that a client wants **Composite Pattern :** A Composite Pattern says that just "allow clients to operate in generic manner on objects that may or may not represent a hierarchy of objects" **Prototype Design Pattern :** Prototype Pattern says that cloning of an existing object instead of creating new one and can also be customized as per the requirement.	https://www.youtube.com/playlist?list=PLsyeobzWxl7r2ZX1fl-7CKnayxHJA_1ol
12-Factor Approach for Microservices	**I. Codebase** : One codebase tracked in revision control, many deploys **II. Dependencies** : Explicitly declare and isolate dependencies **III. Config** : Store config in the environment **IV. Backing services** : Treat backing services as attached resources **V. Build, release, run** : Strictly separate build and run stages **VI. Processes** : Execute the app as one or more stateless processes **VII. Port binding** : Export services via port binding **VIII. Concurrency** : Scale out via the process model **IX. Disposability** : Maximize robustness with fast startup and graceful shutdown **X. Dev/prod parity** : Keep development, staging, and production as similar as possible **XI. Logs** : Treat logs as event streams **XII. Admin processes :** Run admin/management tasks as one-off processes	https://12factor.net/ https://www.youtube.com/watch?v=N45VJ20wvCw
CAP theorem	The CAP theorem is a belief from theoretical computer science about distributed data stores that claims, in the event of a network failure on a distributed database, it is possible to provide either consistency or availability—but not both. **Consistency :** All reads receive the most recent write or an error. **Availability :** All reads contain data, but it might not be the most recent. **Partition tolerance:** The system continues to operate despite network failures (ie; dropped partitions, slow network connections, or unavailable network connections between nodes.)	https://www.bmc.com/blogs/cap-theorem/ https://www.youtube.com/watch?v=8UryASGBiR4  Beyond the CAP Theorem C onsistency Same Data A vailability P artition Tolerance Couchbase redis SQL Server Some Data Isolated Data Amazon DynamoDB Tech _ Primers
PACELC theorem	The PACELC theorem is an extension of the CAP theorem, stating that if there is partitioning (P) in the network, you should choose between availability (A) and consistency (C), else (E), you should select between latency (L) and consistency (C).	**PACLC : Partition > Availablity > Consitency > Lat**
	String s1 = "hello"; String s2 = new String("hello"); String s3 = new String(s2); 1. s1 == s2 : False 2. s1.hashCode() == s2.hashCode() : True 3. s2 == s3 : False 4. s1 == "he"+"llo" : True 5. s1 == ("hello123").substring(0,5) : False 6. s1 == s1.toString() : True	1. s1 == s2 : 2. s1.hashCode() == s2.hashCode() : 3. s2 == s3 : 4. s1 == "he"+"llo" : 5. s1 == ("hello123").substring(0,5) : 6. s1 == s1.toString() :

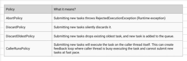
	List a = new ArrayList();
	a.add(null); a.add(null); a.add(null);
	a.size() returns ? :	

	Map m = new HashMap();
	m.put("A", "B");
	m.put("A", m.put("A", "C"));
	
	value of A? :
		B

	Qualifier, Transactional, Transient, PostMapping, PostConstruct	
LinkedHashSet vs TreeSet vs HashSet	**https://www.geeksforgeeks.org/difference-and-similarities-between-hashset-linkedhashset-and-treeset-in-java/**	https://www.youtube.com/watch?v=s3IlKsDyD6U
Difference between HashMap, LinkedHashMap and TreeMap		https://www.geeksforgeeks.org/differences-treemap-hashmap-linkedhashmap-java/
Java - Runnable vs Callable	**Runnable** : If you have a fire and forget task then use Runnable. Put your code inside a Runnable and when the run() method is called, you can perform your task. The calling thread really does not care when you perform your task. **Callable** : If you are trying to retrieve a value from a task, then use Callable. Now callable on its own will not do the job. You will need a Future that you wrap around your Callable and get your values on future.get (). Here the calling thread will be blocked till the Future comes back with results which in turn is waiting for Callable's call() method to execute.	https://medium.com/javarevisited/java-runnable-vs-callable-786aa706775d https://javarevisited.blogspot.com/2016/04/difference-between-ExecutorServie-submit-vs-Executor-execute-method-in-Java.html
Enumeration Vs Iterator In Java :	https://www.geeksforgeeks.org/difference-between-iterator-and-enumeration-in-java-with-examples/	https://javaconceptoftheday.com/differences-between-enumeration-vs-iterator-in-java/
Memory Management	 Hotspot Heap Structure Survivor Space eden S0 S1 Tenured Permanent Young Generation Old Generation Permanent Generation	
SAGA	https://microservices.io/patterns/data/saga.html	https://www.youtube.com/watch?v=WnZ7IcaN_JA . https://www.youtube.com/watch?v=69kqVlvp4p8
mutable class	declare the class as final so it cannot be extended all class members should be private so they cannot be accessed outside of class shouldn't contain any setter methods to change the value of class members the getter method should return the copy of class members class members are only initialized using constructor	
	1,3,5,9,2,4,16,15	
10 Tricky Core Java Interview Coding Questions	https://javaconceptoftheday.com/tricky-core-java-interview-coding-questions/	
	Marker Interface Detailed Study	
	Spring Boot Packaging	
Java is Pass by Value and Not Pass by Reference	https://www.digitalocean.com/community/tutorials/java-is-pass-by-value-and-not-pass-by-reference	
How does implementing Cloneable interface allows cloning of objects, as it is a Marker Interface and doesn't have any methods? [duplicate]	https://stackoverflow.com/questions/52833401/how-does-implementing-cloneable-interface-allows-cloning-of-objects-as-it-is-a#:~:text=Cloneable%20is%20just%20a%20marker,on%20the%20Object%20class%20instead.	
	https://stackoverflow.com/questions/40480/is-java-pass-by-reference-or-pass-by-value	
Reentrant Lock in Java	**https://www.geeksforgeeks.org/reentrant-lock-java/**	https://www.youtube.com/watch?v=7VqWkc9o7RM
Java ExecutorService	https://www.youtube.com/playlist?list=PLhfHPmPYPPRl0LntrCBnQD5ln6lnqqoms	https://javarevisited.blogspot.com/2016/04/difference-between-ExecutorServie-submit-vs-Executor-execute-method-in-Java.html#:~:text=A%20main%20difference%20between%20the,because's%20return%20type%20is%20void.
	newFixedThreadPool	
	newCachedThreadPool	
	newScheduledThreadPool	
	newSingleThreadExecutor	



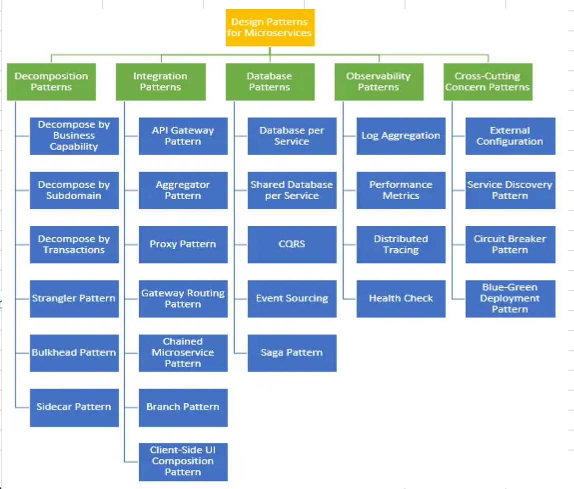
Pool	Queue Type	Why?
FixedThreadPool	LinkedBlockingQueue	Threads are limited, thus unbounded queue to store all tasks.
SingleThreadExecutor	LinkedBlockingQueue	*Note: Since queue can never become full, new threads are never created.*
CachedThreadPool	SynchronousQueue	Threads are unbounded, thus no need to store the tasks. Synchronous queue is a queue with single slot
ScheduledThreadPool	DelayedWorkQueue	Special queue that deals with schedules/time-delays
Custom	ArrayBlockingQueue	Bounded queue to store the tasks. If queue gets full, new thread is created (as long as count is less than maxPoolSize).



Policy	What it means?
AbortPolicy	Submitting new tasks throws RejectedExecutionException (Runtime exception)
DiscardPolicy	Submitting new tasks silently discards it.
DiscardOldestPolicy	Submitting new tasks drops existing oldest task, and new task is added to the queue.
CallerRunsPolicy	Submitting new tasks will execute the task on the caller thread itself. This can create feedback loop where caller thread is busy executing the task and cannot submit new tasks at fast pace.

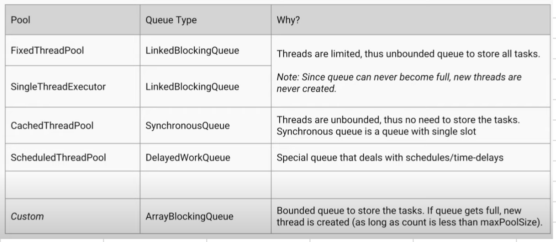
Difference between ExecutorService.submit() and Executor.execute() methods in Java?	https://javarevisited.blogspot.com/2016/04/difference-between-ExecutorServie-submit-vs-Executor-execute-method-in-Java.html#:~:text=A%20main%20difference%20between%20the,because's%20return%20type%20is%20void.
Flat MAP	https://www.javaprogramto.com/2020/07/nested-lists-with-streams-in-java-8.html
Genrics in JAVA	https://www.javatpoint.com/generics-in-java

Linked List			
Mutables Class			
Singleton Pateern		https://medium.com/@kevalpatel2106/how-to-make-the-perfect-singleton-de6b951dfdb0	
Microservice Design Patterns		https://www.openlegacy.com/blog/microservices-architecture-patterns	
		https://microservices.io/tags/pattern	
		Aggregator	
		Backend for frontend	
		API gateway	
		Blue-green deployment	
		Circuit breaker	
		CQRS (command query responsibility segregation)	
		Database per service	
		Shared database	
		Log aggregation	
		Saga	
		Service discovery	
		Event sourcing	
		Strangler application : https://www.redhat.com/architect/pros-and-cons-strangler-architectur	
DDD		https://www.youtube.com/playlist?list=PLZBNtT95PIW3BPNYF5pYOi4MJjg_boXCG	
How to create Immutable class in Java?		The class must be declared as final so that child classes can't be created. Data members in the class must be declared private so that direct access is not allowed. Data members in the class must be declared as final so that we can't change the value of it aft A parameterized constructor should initialize all the fields performing a deep copy so that data Deep Copy of objects should be performed in the getter methods to return a copy rather than	https://www.geeksforgeeks.org/create-immutable-class-java/
		The 12 factors outline various aspects of application design, deployment, and operation, with the goal of promoting scalability, maintainability, and portability. Here's a summary of each factor: Codebase: Each application should have a single codebase, version-controlled by a revision control system like Git. Dependencies: Explicitly declare and isolate dependencies. The application should not rely on globally installed libraries or packages. Config: Configuration should be stored in the environment and separate from the codebase. This allows for easier configuration management and portability across different environments. Backing Services: Treat backing services (databases, caches, message queues) as attached resources that can be accessed via a URL or connection string. Avoid hard-coding their details into the application. Build, Release, Run: Separate the build, release, and run stages of application lifecycle. Each stage should have distinct processes and dependencies. Processes: Applications should be executed as stateless processes that share nothing. Any required state should be stored in a backing service. Port Binding: Applications should be self-contained and export services via a port binding mechanism. They should not rely on specific hostnames or ports. Concurrency: Scale out by adding more instances of the application instead of scaling up a single instance. Applications should be designed to run effectively in a distributed environment. Disposability: Applications should be designed for fast startup and graceful shutdown. Processes can be started or stopped at any time without impacting the overall system. Dev/Prod Parity: Aim for maximum parity between development, staging, and production environments. Differences between environments should be minimized to reduce issues during deployment. Logs: Treat logs as event streams, and applications should produce logs in a structured format. Centralized logging allows for easier debugging and monitoring. Admin Processes: Administrative and management tasks should be run as one-off processes, rather than being tightly coupled with the main application.	

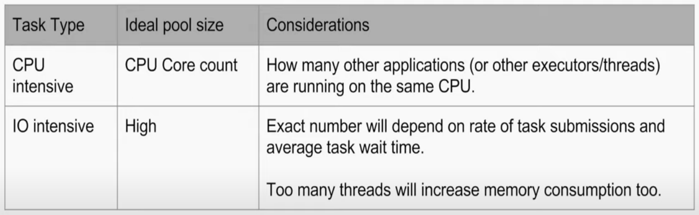


Topic	Reference
Threading	https://www.youtube.com/playlist?list=PLhfHPmPYPPRl0LntrCBnQD5ln6lnqqoms
Java 8 Stream and lambda	https://www.youtube.com/playlist?list=PLqq-6Pq4lTTa9YGfyhyW2CqdtW9RtY-I3
	https://www.youtube.com/playlist?list=PLTyWtrsGknYdqY_7lwcbJ1z4bvc5yEEZl
Design Patterns	https://www.youtube.com/playlist?list=PLsyeobzWxl7r2ZX1fl-7CKnayxHJA_1ol
Solid Principle	https://www.youtube.com/watch?v=yxf2spbpTSw
Using volatile vs AtomicInteger	https://www.youtube.com/watch?v=WH5UvQJizH0
Hashmap	https://www.youtube.com/watch?v=CojCE-ojdGY
ConcurrentHashMap internal implementation	https://www.youtube.com/watch?v=6Zzm4esAi7A
LinkedHashSet vs TreeSet vs HashSet	https://www.youtube.com/watch?v=s3lIKsDyD6U
Saga Design Pattern	https://www.youtube.com/watch?v=WnZ7IcaN_JA
	https://www.youtube.com/watch?v=69kqVIvp4p8
Reentrant Lock in Java	https://www.youtube.com/watch?v=7VqWkc9o7RM

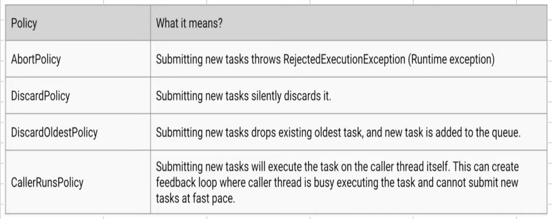
Using volatile vs AtomicInteger in Java concurrency	https://www.youtube.com/watch?v=WH5UvQJizH0
Java ExecutorService	https://www.youtube.com/playlist?list=PLhfHPmPYPPRl0LntrCBnQD5ln6lnqqoms
	newFixedThreadPool
	newCachedThreadPool
	newScheduledThreadPool
	newSingleThreadExecutor



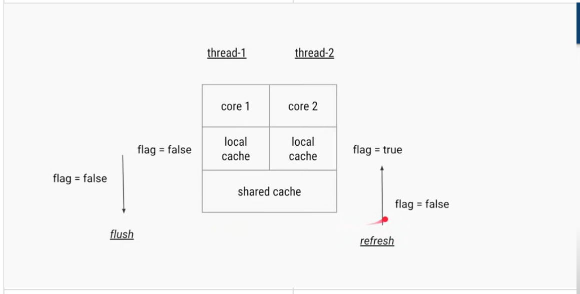
Pool	Queue Type	Why?
FixedThreadPool	LinkedBlockingQueue	Threads are limited, thus unbounded queue to store all tasks. *Note: Since queue can never become full, new threads are never created.*
SingleThreadExecutor	LinkedBlockingQueue	
CachedThreadPool	SynchronousQueue	Threads are unbounded, thus no need to store the tasks. Synchronous queue is a queue with single slot
ScheduledThreadPool	DelayedWorkQueue	Special queue that deals with schedules/time-delays
Custom	ArrayBlockingQueue	Bounded queue to store the tasks. If queue gets full, new thread is created (as long as count is less than maxPoolSize).

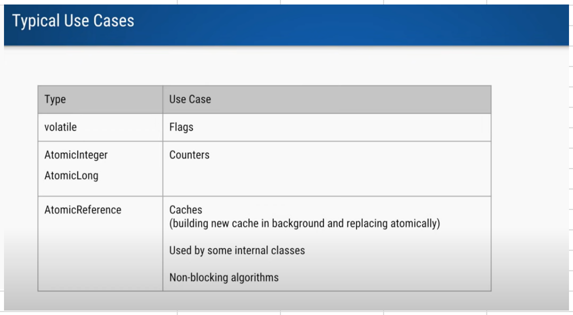


Task Type	Ideal pool size	Considerations
CPU intensive	CPU Core count	How many other applications (or other executors/threads) are running on the same CPU.
IO intensive	High	Exact number will depend on rate of task submissions and average task wait time. Too many threads will increase memory consumption too.



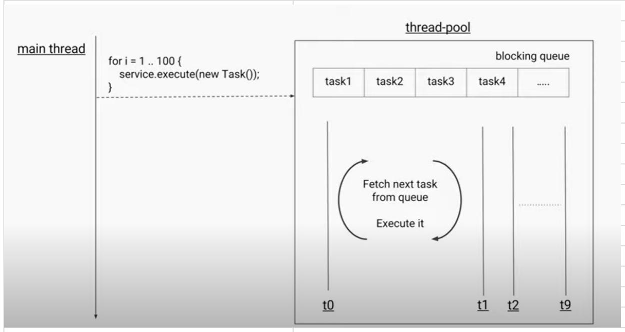
Policy	What it means?
AbortPolicy	Submitting new tasks throws RejectedExecutionException (Runtime exception)
DiscardPolicy	Submitting new tasks silently discards it.
DiscardOldestPolicy	Submitting new tasks drops existing oldest task, and new task is added to the queue.
CallerRunsPolicy	Submitting new tasks will execute the task on the caller thread itself. This can create feedback loop where caller thread is busy executing the task and cannot submit new tasks at fast pace.

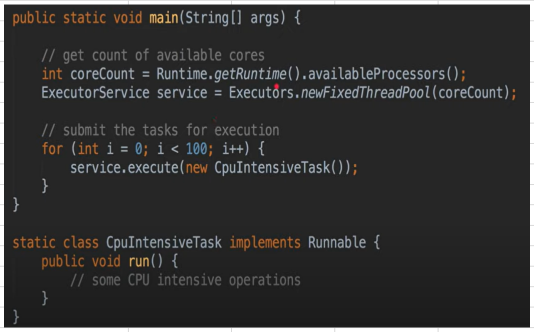




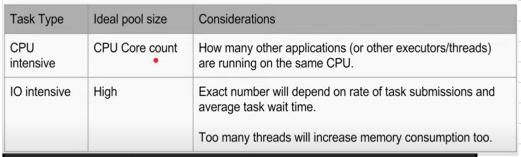
Typical Use Cases

Type	Use Case
volatile	Flags
AtomicInteger AtomicLong	Counters
AtomicReference	Caches (building new cache in background and replacing atomically) Used by some internal classes Non-blocking algorithms

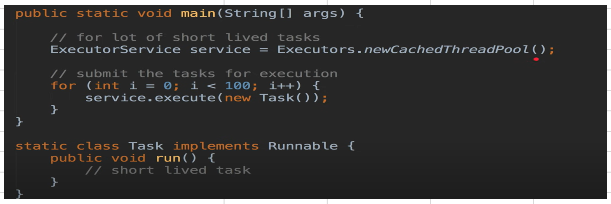




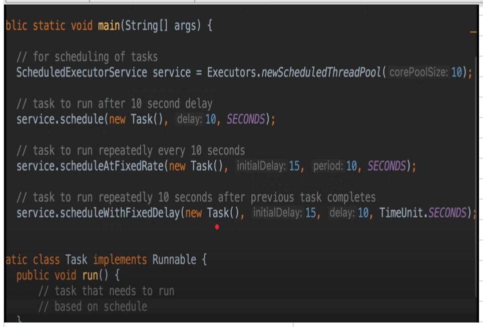
```
public static void main(String[] args) {

    // get count of available cores
    int coreCount = Runtime.getRuntime().availableProcessors();
    ExecutorService service = Executors.newFixedThreadPool(coreCount);

    // submit the tasks for execution
    for (int i = 0; i < 100; i++) {
        service.execute(new CpuIntensiveTask());
    }
}

static class CpuIntensiveTask implements Runnable {
    public void run() {
        // some CPU intensive operations
    }
}
```



Task Type	Ideal pool size	Considerations
CPU intensive	CPU Core count	How many other applications (or other executors/threads) are running on the same CPU.
IO intensive	High	Exact number will depend on rate of task submissions and average task wait time. Too many threads will increase memory consumption too.



```
public static void main(String[] args) {

    // for lot of short lived tasks
    ExecutorService service = Executors.newCachedThreadPool();

    // submit the tasks for execution
    for (int i = 0; i < 100; i++) {
        service.execute(new Task());
    }
}

static class Task implements Runnable {
    public void run() {
        // short lived task
    }
}
```



```
blic static void main(String[] args) {

  // for scheduling of tasks
  ScheduledExecutorService service = Executors.newScheduledThreadPool( corePoolSize: 10);

  // task to run after 10 second delay
  service.schedule(new Task(),  delay: 10, SECONDS);

  // task to run repeatedly every 10 seconds
  service.scheduleAtFixedRate(new Task(),  initialDelay: 15,  period: 10, SECONDS);

  // task to run repeatedly 10 seconds after previous task completes
  service.scheduleWithFixedDelay(new Task(),  initialDelay: 15,  delay: 10, TimeUnit.SECONDS);


atic class Task implements Runnable {
  public void run() {
      // task that needs to run
      // based on schedule
```

	What are the advantages of Hibernate over JDBC?
	Difference between get() vs load() method in Hibernate?
	What is the difference between save() and persist() method in Hibernate?
	What is the requirement for a Java object to become Hibernate entity object?
	What are different types of caches available in Hibernate?
	What is the difference between first and second level cache in Hibernate?
	Does Hibernate Session interface is thread-safe in Java?
	Does SessionFactory is thread-safe in Hibernate?
	What is different between Session and Sessionfactory in Hibernate?
	When do you use merge() and update() in Hibernate?
	The difference between sorted and ordered collection in Hibernate?
	What are the three states of a Hibernate Persistence object can be?

	What is autowiring in spring ?				
	AOP and IOC				
	What is IOC and DI? What is the role of IOC container in spring? What are the types of IOC container in spring? BeanFactory ApplicationContext	https://www.geeksforgeeks.org/spring-difference-between-inversion-of-control-and-dependency-injection/			
	What is the difference between BeanFactory and ApplicationContext?				
	-- BeanFactory is the basic container whereas ApplicationContext is the advanced container. ApplicationContext extends the BeanFactory interface. ApplicationContext provides more facilities than BeanFactory such as integration with spring AOP, message resource handling for i18n etc. -- What is autowiring in spring? What are the autowiring modes? What are the different bean scopes in spring? What are the advantages of JdbcTemplate in spring What is AOP? --- What is the front controller class of Spring MVC? What does @RequestMapping annotation? What does the ViewResolver class? ---				
	Qualifier, Transactional, Transient, PostMapping, PostConstruct				
	https://www.edureka.co/blog/interview-questions/spring-boot-interview-questions/				
	https://www.edureka.co/blog/interview-questions/spring-interview-questions/				

	Architecture					
		https://www.simplilearn.com/tutorials/kubernetes-tutorial/kubernetes-interview-questions				

IAM Section	21	38
EC2	40	
S3	272	
EBS	97	
ELB	131	
Auto Scalling	162	
RDS	169	
Route 53	203	
Elastic Beanstalk	268	
AWS CloudFront	329	
AWS Global Accelerator	350	
Amazon SQS	382	
CloudWatch vs CloudTrail vs Config	580	
AWS STS – Security Token Service	582	
AWS Organizations	594	
Resource Access Manager	611	
Virtual Private Cloud	654	655

	Define ACID Properties?

	1. Restore view 2. Apply request values; process events 3. Process validations; process events 4. Update model values; process events 5. Invoke application; process events 6. Render response

	Subtasks For User Story
	Requirement Analysis
	Requirement Analysis review
	Impact analysis & Design
	Impact analysis & Design review
	Java subtask - Bussiness Logic Handing, - Logging, - Exception Handling, - Persistance Layer Integration
	DB subtask (Optional)
	Java Code review
	DB Code review
	Test Case Writing JUNIT : Integration test cases, Functional test cases, System test cases Selenuim : End to End Testing
	Test Case Review
	Test Case Execution
	Addtional Subtaks (System & Code Setup, Deployement Activities)
	Publish at One API
	Load Testing
	Core Java
	Design Patterns
	Communication
	Coding
	Core Java
	AWS
	Devops
	Microservices
	Design Patterns
	Communication

	What do you mean by ‘Hot deployment’ ?