Homework 4

1. Question List (write necessary code to answer the following questions)
   1. Coding:

create a list of students, Student Class has name, age, score three fields.

List<Student> list = new ArrayList<>();

* use stream api to find all the students’ name starting with ‘A’
* use stream api to get the sum of all the students score
* use stream api to find all the students whose sore >= 60
* use stream api to retrieve all students name
* use stream api to count the frequency of each age

A screenshot of a computer program

AI-generated content may be incorrect.

1. intermediate operation vs terminal operation

the intermediate operation will return the Stream type of the object, and the intermediate operation such as filter, map, distinct, flatmap.

The terminal operation will return normal datatype, and the interface such as collect, forEach, reduce, min.

1. thread lifecycle, how does thread transfer from one state to another

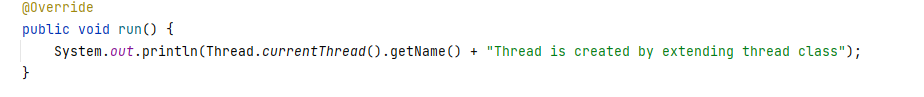
new state, runnable state, block state, waiting state, timed waiting state, terminated.

The new thread will put in runnable state when it ready to start. The thread will be blocked state when it trying to acquire the lock but currently the lock is acquired by the other thread. The thread will move from the block state to runnable state when it acquire the lock. The thread will be in waiting state when it call wait method or join method. And it will move to the runnable state. When other state will notify or will be terminated. A thread will lies in timing waiting state when it calls a method time-out parameter. Thread will lies in until the timeout is completed or until a notification received. When it go to terminated state there is two reason one is the thread finish their job and exists. The other reason is there has some error occur.

1. how to create a thread( 4 ways, write code)

there is 4 day to create a thread.

The first one is the extended the thread and overwrite the run method.



The second one is by implementing the runnable class and overwrite the the run method

A computer code on a white background

AI-generated content may be incorrect.

The third one is by implementing the the callable class and overwrite the call method

A screen shot of a computer code

AI-generated content may be incorrect.

The last one is create a threadpool object.

A white background with colorful text

AI-generated content may be incorrect.

1. how does thread pool work

threadpool is the thread management tool which can save the computing power for the system. If there is multi task coming, the thread pool will allow limit of task into runnable state and , some will store in thread pool until the pool size fix. The other task will be put into the working queue wait for notification or if there is the space empty for them to join.

1. what is the potential problem for the newCachedThreadPool and newFixedThreadPool and why

So in terms of resource, the newFixedThreadPool will keep all the thread running until they are explicitly terminated. In the newCachedThreadPool threads that have not been used for sixty seconds are terminated and removed from the cache.

This resource consumption will depend on very much in situation. If we have a huge number of long running task, we should use the fixedThreadpool. And if we have shor-live asynchronous task we should choose the newcachedThreadPool which can improve the performance.

1. what is Future

Future is an abstraction used in concurrent or asynchronous programming to represent a value that may not be available yet because its computation still in progress.

1. what is CompletableFuture

CompletableFuture is used for deal with asynchronous programming, when we run the completableFuture it will start a new thread to run the program. And the main thread don’t need to wait and it can do the something else until the completablefuture return the result. And also we can chain the result by using the completableFuture.

1. Future vs CompletableFuture

Both future and completableFutre are both work on asynchronous programming.

Future only provide the basic mechanism to check if a computation is complete, wait for its result, and cancel the computation. CompletableFuture is build on Future and implement the completion stage interface make it much more flexible and it support for non-blocking operation.

1. Lock vs synchronized

Lock is we lock some process, such as some block of code, some method or some class. There is two way to use the lock the one way is we need to use synchronized keyword. It can lock the code block, static method, method also some class. Ant the other way is we using the lock interface such as Reentrantlock, it is more flexible. Why this more flexible because it has some lock method and unlock method also trylock method. It has different method to lock object so it is easily to manage the object.

Synchronized is when two process come together so we need to lock one process to make other process to be excute.

1. what is wait(), notify(), notifyAll(), join()

wait method is call object monitor pause the current thread until the another thread signals change. It release a lock on a object, and allowing other thread to enter a synchronized block on the same object.

notify method wake up one waiting thread.The choice of which thread is generally not specified, so it may not be the thread you want.

Notifyall method should be wake up all waiting threads, allowing each one to compete for the lock once it’s release. This particularly useful when multiple thread might be waiting for a condition to change.

Join method when we call thread.join on a thread instance, the calling thread will block until the target thread finish its execution. This helpful for ensuring that a thread has completed its task before processing with subsequent operations.

Homework 5

1. Question List (write necessary code to answer the following questions)
   1. what is DeadLock,

DeadLock is like two thread are waiting for each other. Such like when we has one thread need to execute and it is waiting for thread to unlock it but the thread2 also need thread1 notify to be excute.

* 1. how to create deadlock(write the code by using ReentrantLock),

1. import java.util.concurrent.locks.ReentrantLock;  
     
   public class ReentranLockDemo {  
    private static final ReentrantLock *lock1* = new ReentrantLock();  
    private static final ReentrantLock *lock2* = new ReentrantLock();  
     
    public static void main(String[] args) {  
    Thread thread1 = new Thread(() -> {  
    try{  
    *lock1*.lock();;  
    System.*out*.println("Thread 1: Acquire lokc1");  
    Thread.*sleep*(100);  
    System.*out*.println("Thread1 is waitting for thread2");  
    *lock2*.lock();  
    System.*out*.println("Thread 1: Acquire lock 2");  
    } catch (InterruptedException e) {  
    e.printStackTrace();  
    } finally {  
    if(*lock2*.isHeldByCurrentThread()) {  
    *lock2*.unlock();  
    }  
    *lock1*.unlock();  
    }  
    });  
    Thread thread2 = new Thread(() -> {  
    try {  
    *lock2*.lock();  
    System.*out*.println("Thread 2: Acquired lock");  
    Thread.*sleep*(100);  
    System.*out*.println("Thread 2 is waitting for thread 1");  
    *lock1*.lock();  
    System.*out*.println("Thread 1: Acquired lock");  
     
    } catch (InterruptedException e) {  
    e.printStackTrace();  
    } finally {  
    {  
    if(*lock1*.isHeldByCurrentThread()) {  
    *lock1*.unlock();  
    }  
    *lock2*.unlock();  
    }  
    }  
    });  
    thread1.start();  
    thread2.start();  
     
    }  
   }
   1. how to prevent deadlock and why

So the first way to prevent deadlock we need to avoid the nested locks. The second way to prevent the dead lock is we need to avoid the unnecessary lock we need to avoid to use those data structure such as vector, stack, hashtable because they are all threadsafe and they will provide the lock automatically.

* 1. CompletableFuture vs Future

The future only provide the basic mechanism to check if the computation is complete or not. And the completableFuture is base on the future but it is more flexible because of it is implementation. And it support the non-blocking operation.

* 1. CompletableFuture common API

They are some API, such as runAsync which give us no return value, and it should be call by the future.get.

The other common api is has return value such as supplyAsync. Also by those return value we can chain all result.

* 1. use completableFuture implements the following logic
     1. initial Num = 1
     2. async: num += 10
     3. sync: num \*=4
     4. async: consume result and print
     5. exception: if exception, handle it

1. System.*out*.println("Main Thread starts");  
   CompletableFuture<Integer> future = CompletableFuture.*supplyAsync*(() -> {  
    int num = 1;  
    return num + 10;  
   }).thenApply(val -> val \* 4 ).thenAcceptAsync(result -> {  
    System.*out*.println("Final result: "+ result);  
   }).handle((val, ex) -> {  
    System.*out*.println("Get info the handle method");  
    if(ex == null) {  
    System.*out*.println("task done");  
    return null;  
    } else {  
    System.*out*.println("Exception throw" + ex.getMessage());  
    return 0;  
    }  
   });  
   future.get();

A white background with black text

AI-generated content may be incorrect.

* 1. write the producer and consumer model (by using synchronized keyword)

1. class ProducerConsumerModel {  
    private Queue<Integer> queue = new LinkedList<>();  
    private final int capacity = 3;  
     
    private Random myRandom = new Random();  
     
    public void put() throws InterruptedException {  
    synchronized (queue) {  
    while(queue.size() == capacity) {  
    System.*out*.println(Thread.*currentThread*().getName() + "wait, queue is full");  
    queue.wait();  
    }  
    int tempVal = myRandom.nextInt(100);  
    queue.offer(tempVal);  
    System.*out*.println(Thread.*currentThread*().getName() + "put" + tempVal);  
    queue.notifyAll();  
    }  
    }  
    public void take() throws InterruptedException {  
    synchronized (queue) {  
    while (queue.isEmpty()) {  
    System.*out*.println(Thread.*currentThread*().getName() + "wait producer to produce product");  
    queue.wait();  
    }  
    int tempVal = queue.poll();  
    System.*out*.println(Thread.*currentThread*().getName() + "take" + tempVal);  
    queue.notifyAll();  
    ;  
    }  
    }  
   }
   1. synchronized normal method vs synchronized static method

the normal synchronized method when it executed, it will lock the object itself. Like if we have an object call shareObject. And it will block this object. On the other hand when we use synchronizated static method, it willblock the class of that object. So it means it will block the shareObjectClass.

Homework 6

1. Question List (write necessary code to answer the following questions)
   1. write out the optimized Singleton Version and explain each line of code
2. class Singleton2 {  
    *//volatitle to prevent instruction reorder* private static volatile Singleton2 *instance*;  
     
    *//private constructor to prevent instantiation from other class* private Singleton2() {}  
     
     
    *//not thread safe* public static Singleton2 getInstance() {  
    if (*instance* == null) { *//first check(no locking for the instance)* synchronized (Singleton2.class) {  
    if(*instance* == null) { *//second check with locking  
    instance* = new Singleton2();  
    *//1 create instance reference  
    //2 new singleton  
    //instance reference points to instance object,  
    // 1 -> 2 -> 3  
    // 1 -> 3 -> 2* }  
    }  
     
    }  
    return *instance*;  
    }  
   }
   1. what are the use case for singleton

the use case for singleton when we have hardware interface access, or the Logger method. And when we write the configuration file input and output file.

* 1. do research on Factory, Builder, Observer, Decorator, static and dynamic Proxy pattern. what are their use cases, pros and cons

Factory Patter: Creating object without specifying exact class.Pros: Decouples client mode, centralizes creation logic. Cons: Can add extra classes and complexity.

Builer Pattern: Step-by-step construction of complex object. Pros: Improves clarity, support immutability. Cons: Can be verbose for simple objects.

Observer Pattern: Implementation event handling system, MVC architectures. Pros: Subjects and observers are lossely coupled. Observers can be added or removed at run time. Cons: It will cause memory leak. Over-notification can occur if not managed carefully, leading to performance issues.

Decorator Pattern: When we want to add responsibilities or behaviors to objects at runtime. Pros: Offer flexible enhancements without change original object.

Cons: Can result in many small class and nested wrapper.

Static Proxy Pattern: Compile-time proxies adding fixed behaviors. Pros: Simple, and type-safe. Cons: Requires extra boilerplate and maintenance for each interface

Dynamic Proxy Pattern: Use Case: Runtime proxying for flexible cross-cutting concerns (e.g., AOP). Pros: Reduces boilerplate; adapts dynamically.

Cons: Uses reflection (may impact performance) and can be more complex to debug.

* 1. what is reflection

the reflection is we can modify the behavior of method, class and interface at a run time.

* 1. how does annotation work in spring

There is serval step in Annotation Processing. Component Scanning. The spring container scans the specific packages for classes annotated with @Component, @Service, @Repository @Controller. And each annotation has their on responsibility. Annotation work with @ sign.

Homework 7

1. Question list
   1. what are http status code, 200/ 201/202/ 204/ 307/ 301/ 400/ 401/ 403/ 404/ 500, explain them by your own words

200 -> ok

201 ->Created

202 ->Accept

204 -> No content

307 -> temporary redirect

301 -> move permanently

400 -> bad request

401 -> unauthrization

403 -> forbidden

404-> page not found

500 -> internal server error

* 1. what is http

the http is hyper text transform protocol

* 1. what is get, post, put, delete, patch method

the get method is to make the request to our target website and they will sent us an response. It always be a json type.

The post method is when we try to create something to there database, then we use the post method.

The put method is when we need to update the existing profile in their database so we need to use put method.

The delete method is we try to delete something from their server.

Patch methos it is kind of similar to the put method, but the patch method it only be modify some specific item not a whole object.

* 1. post vs patch

the post is we create a new thing to the server, and it is no safe also no idempotent and no cacheable.

Patch is we update some specific data in the server, it is also not safe, no idempotent and not cacheable

* 1. post vs put

post is we create a new thing to the server.

Put is we update existing data in the server and it is idempotent.

* 1. TCP 3 way handshaking

The client send TCP request with SYN flag with an initial sequence number to the server. This signals the client’s intent to initial a connection and start process of synchronizing sequence number.

Upon receiving the SYN, the server respond with a segment that has both SYN and ACK flags set. This step confirm receipt of the client’s request and indicates the server is ready to establish the connection.

The client send a final segment with ACK flag set. This completes the handshake process, and the connection is fully established, allowing data to be exchanged reliably.

* 1. TCP vs UDP

` TCP has low speed and error detection, and it has three way shaking

UDP has fast speed, connectionless protocol prioritizing speed over reliability. They don’t care about whether data packet was lost or arrive out of order.

* 1. what is Tomcat

Tomcat is an open-source, light-weight web server and servlet container. It is widely used for hosting Java-based applications and implementations java servlet and java server page specification.

* 1. what are the basic components for tomcat

there are three tomcat basic components: servlet container, JSP engine, and Http connector. They work together to handle the web request, process the servlets and compile javaserver page.

* 1. what is Spring IOC

The Spring IOC is the inversion of the control. And it is the core that manages objects, their dependencies, and their lifecycles, promoting loose coupling and modulartiy

* 1. what is IOC Container

An IOC container is a framework or tool that manages the liftcycle and dependencies of object in a software application, allowing developers to focus on building logic rather than object creation and dependency management.

* 1. what is the advantage come with IOC

it can improved the modularity. Components become more modular and reusable because they don’t need to know how to create their dependencies.

Loose coupling Dependency injection, a key mechanism used by IOC containers, results in loose coupled components, making the system more maintainable and testable.

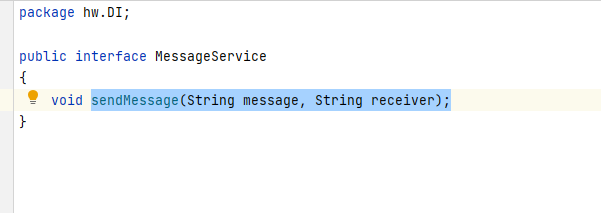
Configurability: IOC containers allow for easy change to the application’s behavior without modifying the source code by using configuration metadata.

Lifecycle Management: The container manages the lifecycle of beans, ensuring that they are created, configured, and destroyed in a controller manner.

* 1. what is Dependency Injection (DI)

DI is a design pattern where an object’s dependencies are provided by an external entity rather than being created by the object itself. This approach makes your code more modular testable and easier to maintain

* 1. write a demo code to show what is Dependency Injection (give screenshot)



A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

* 1. how to do dependency injection

There is serval way to do it. First it is Constructor injection, dependencies are provided through a class constructor.

Second way it is setter injection, dependencies are provided through the setter method. This is useful when dependencies are optional or when you need to change them after the object creation.

The third one is Field injection. The field injection directly assigns dependencies to fields, typically using annotations. This method is commonly used with dependency injection framework like Spring.

* 1. @Component vs @Bean

Those are both Spring annotation used for managing beans.

@Component is a class-level annotation that marks a class as a spring-managed Component. It is auto-detection and configuration. We have limited control on it because it is auto manage by spring component

@Bean it is a method-level annotation used within @Configuration classes to explicitly declare a Spring Bean. It is explicitly bean declaration. We have more control on creation and configuration

* 1. what is @Configuration

The @Configuration is a class-level annotation that is part of the Spring Framework. It indicates that a class serves as a source of bean definitions.

* 1. what is AOP

The solution for cross-cutting concerns. It is a programming paradigm designed to increase modularity by separating cross-cutting from the core logic. Instead of scattering code for concerns like logging, security or transaction management throughout your application. AOP allows you to encapsulate these concerns into separate units call aspects.

* 1. what is JointPoint and Aspect in AOP

A join point is a specific point in the execution of a program where additional behavior can be interested. Common join points include method call, method execution, or even exception handling.

An aspect is a collection of behavior and the rules that determine when and where those behaviors should be applied

* 1. spring bean scope

Springbean scope in Spring FrameWork Define the lifecycle and visibility of a bean within the Spring Container. Here are some key scope:

Singleton Scope, Prototype, Request, Session, Application, webSocket.