HW2

1 – IOC and DI means ?

Inversion of Control is a principle in software engineering which transfers the control of objects or portions of a program to a container or framework. We most often use it in the context of object-oriented programming.

In contrast with traditional programming, in which our custom code makes calls to a library, IoC enables a framework to take control of the flow of a program and make calls to our custom code. To enable this, frameworks use abstractions with additional behavior built in. If we want to add our own behavior, we need to extend the classes of the framework or plugin our own classes.

The advantages of this architecture are:

- decoupling the execution of a task from its implementation
- making it easier to switch between different implementations
- greater modularity of a program
- greater ease in testing a program by isolating a component or mocking its dependencies, and allowing components to communicate through contracts

We can achieve Inversion of Control through various mechanisms such as: Strategy design pattern, Service Locator pattern, Factory pattern, and Dependency Injection (DI).

Dependency injection is a pattern we can use to implement IoC, where the control being inverted is setting an object's dependencies.

Connecting objects with other objects, or “injecting” objects into other objects, is done by an assembler rather than by the objects themselves.

Both IoC and DI are simple concepts, but they have deep implications in the way we structure our systems, so they're well worth understanding fully.

2 – Spring Bean Scopes ?

Spring Bean Scopes allows us to have more granular control of the bean instances creation. Sometimes we want to create bean instance as singleton but in some other cases we might want it to be created on every request or once in a session.

There are five types of **spring bean scopes**:

- **Singleton** – only one instance of the spring bean will be created for the spring container. This is the default spring bean scope. While using this scope, make sure bean doesn’t have shared instance variables otherwise it might lead to data inconsistency issues.
- **Prototype** – A new instance will be created every time the bean is requested from the spring container.
- **Request** – This is same as prototype scope, however it’s meant to be used for web applications. A new instance of the bean will be created for each HTTP request.
- **Session** – A new bean will be created for each HTTP session by the container.
- **Global Session** – This is used to create global session beans for Portlet applications.

3 – What does @SpringBootApplication do ?

Many Spring Boot developers like their apps to use auto-configuration, component scan and be able to define extra configuration on their "application class". A single **@SpringBootApplication** annotation can be used to enable those three features, that is:

- **@EnableAutoConfiguration**: enable Spring Boot's auto-configuration mechanism.
- **@ComponentScan**: enable **@Component** scan on the package where the application is located (see the best practices).
- **@Configuration**: allow to register extra beans in the context or import additional configuration classes.

The **@SpringBootApplication** annotation is equivalent to using **@Configuration**, **@EnableAutoConfiguration**, and **@ComponentScan** with their default attributes.

4 – What is Spring AOP ? Where and How to use it ?

One of the key components of Spring Framework is the **Aspect oriented programming (AOP)** framework. Aspect-Oriented Programming entails breaking down program logic into distinct parts called so-called concerns. The functions that span multiple points of an application are called **cross-cutting concerns** and these cross-cutting concerns are conceptually separate from the application's business logic. There are various common good examples of aspects like logging, auditing, declarative transactions, security, caching, etc.

The key unit of modularity in OOP is the class, whereas in AOP the unit of modularity is the aspect. Dependency Injection helps you decouple your application objects from each other and AOP helps you decouple cross-cutting concerns from the objects that they affect. AOP is like triggers in programming languages such as Perl, .NET, Java, and others.

Spring AOP module provides interceptors to intercept an application. For example, when a method is executed, you can add extra functionality before or after the method execution.

5 – What is Singleton and where to use it ?

The singleton design pattern restricts the instantiation of a class to a single instance. This is done in order to provide coordinated access to a certain resource, throughout an entire software system. Through this design pattern, the singleton class ensures that it's only instantiated once, and can provide easy access to the single instance.

Common **use-cases** for the singleton design pattern include factories, builders, and objects that hold program state. Singletons are sometimes considered to be an alternative to global variables or static classes.

Compared to global variables, singletons have the following benefits:

- Singleton instance fields don't take up space in the global namespace
- Singletons may be lazily initialized (to be discussed further)

Primarily due to the fact that a singleton holds an instantiated object, whereas static classes do not, singletons have the following advantages over static classes:

- Singletons can implement interfaces
- Singletons can be passed as parameters

- Singletons can have their instances swapped out (such as for testing purposes)
- Singletons can be handled polymorphically, so there may exist multiple implementations

6 – What is Spring Boot Actuator and Where to use it ?

In essence, **Actuator** brings production-ready features to our application.

Monitoring our app, gathering metrics, understanding traffic, or the state of our database become trivial with this dependency.

The main benefit of this library is that we can get production-grade tools without having to actually implement these features ourselves.

Actuator is mainly used to **expose operational information about the running application** — health, metrics, info, dump, env, etc. It uses HTTP endpoints or JMX beans to enable us to interact with it.

Once this dependency is on the classpath, several endpoints are available for us out of the box. As with most Spring modules, we can easily configure or extend it in many ways.

To enable Spring Boot Actuator, we just need to add the *spring-boot-actuator* dependency to our package manager.

In Maven:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Note that this remains valid regardless of the Boot version, as versions are specified in the Spring Boot Bill of Materials (BOM).

7 - What is the primary difference between Spring and Spring Boot ?

Spring	Spring Boot
Spring Framework is a widely used Java EE framework for building applications.	Spring Boot Framework is widely used to develop REST APIs.
It aims to simplify Java EE development that makes developers more productive.	It aims to shorten the code length and provide the easiest way to develop Web Applications.
The primary feature of the Spring Framework is dependency injection.	The primary feature of Spring Boot is Autoconfiguration. It automatically configures the classes based on the requirement.
It helps to make things simpler by allowing us to develop loosely coupled applications.	It helps to create a stand-alone application with less configuration.

The developer writes a lot of code (boilerplate code) to do the minimal task.	It reduces boilerplate code.
To test the Spring project, we need to set up the sever explicitly.	Spring Boot offers embedded server such as Jetty and Tomcat, etc.
It does not provide support for an in-memory database.	It offers several plugins for working with an embedded and in-memory database such as H2.
Developers manually define dependencies for the Spring project in pom.xml.	Spring Boot comes with the concept of starter in pom.xml file that internally takes care of downloading the dependencies JARs based on Spring Boot Requirement.

8 – Why to use VCS ?

Software development includes the continuous process of modifying programs and **version control system** makes this task easier. Version control software is used by software developers to maintain documentation and configuration files as well as source code. It helps developers to store different versions of software safely and in an organized manner.

With this, a Software Engineer can quickly identify which version needs to be modified and what commits had been done previously and should be done.

9 – What are SOLID Principles ? Give sample usages in Java ?

SOLID is the acronym for a set of practices that, when implemented together, makes the code more adaptive to change. Bob Martin and Micah Martin introduced these concepts in their book 'Agile Principles, Patterns, and Practices'.

The acronym was meant to help us remember these principles easily. These principles also form a vocabulary we can use while discussing with other team members or as a part of technical documentation shared in the community.

SOLID principles form the fundamental guidelines for building object-oriented applications that are robust, extensible, and maintainable.

For example, if a given class is a model class then it should strictly represent only one actor/entity in the application. This kind of design decision will give us the flexibility to make changes in the class, in future without worrying the impacts of changes in other classes.

Similarly, If we are writing service/manager class then the class should contain only that part of methods and nothing else. The service class should not contain even utility global functions related to the module.

Better to separate the global functions in another globally accessible class. This will help in maintaining the class for that particular purpose, and we can decide the visibility of class to a specific module only.

10 - What is RAD model ?

The **RAD** (Rapid Application Development) model is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product.

Rapid Application Development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.

11 - What is Spring Boot starter ? How is it useful ?

Spring Boot provides a number of starters that allow us to add jars in the classpath. Spring Boot built-in starters make development easier and rapid. Spring Boot Starters are the dependency descriptors.

In the Spring Boot Framework, all the starters follow a similar naming pattern: spring-boot-starter-*, where * denotes a particular type of application. For example, if we want to use Spring and JPA for database access, we need to include the spring-boot-starter-data-jpa dependency in our pom.xml file of the project.

12 – What is Caching ? How can we achive caching in Spring Boot ?

Spring Framework provides caching in a Spring Application, transparently. In Spring, the **cache** abstraction is a mechanism that allows consistent use of various caching methods with minimal impact on the code.

The cache abstraction mechanism applies to Java methods. The main objective of using cache abstraction is to reduce the number of executions based on the information present in the cache. It applies to expensive methods such as CPU or IO bound.

Every time, when a method invokes, the abstraction applies a cache behavior to the method. It checks whether the method has already been executed for the given argument or not.

- If yes, the cached result is returned without executing the actual method.
- If no, first, the method executes, and the result is cached and returned to the user.

Note: This approach works only for the methods that are guaranteed to return the same result for a given input. It does not matter how many times the method executes.

The developers take care of two things while working with cache abstractions.

- Cache Declaration: It identifies the methods that need to be cached.
- Cache Configuration: The backing cache where the data is stored and read from.

Caching is a part of temporary memory (RAM). It lies between the application and persistence database. It stores the recently used data that reduces the number of database hits as much as possible. In other words, caching is to store data for future reference.

The primary reason of why should we use the cache; for using cache is to make data access faster and less expensive. When the highly requested resource is requested multiple times, it is often beneficial for

the developer to cache resources so that it can give responses quickly. Using cache in an application enhances the performance of the application. Data access from memory is always faster in comparison to fetching data from the database. It reduces both monetary cost and opportunity cost.

What data should be cached?

- The data that do not change frequently.
- The frequently used read query in which results does not change in each call, at least for a period.

There are four types of caching are as follows:

- In-memory Caching
- Database Caching
- Web server Caching
- CDN Caching

13 – What & How & Where & Why to logging ?

Logging is a powerful aid for understanding and debugging program's run-time behavior. Logs capture and persist the important data and make it available for analysis at any point in time.

This article discusses the most popular java logging framewloorks, Log4j 2 and Logback, along with their predecessor Log4j, and briefly touches upon SLF4J, a logging facade that provides a common interface for different logging frameworks.

All the logging frameworks discussed in the article share the notion of loggers, appenders and layouts. Enabling logging inside the project follows three common steps:

- Adding needed libraries
- Configuration
- Placing log statements

14 - What is Swagger? Have you implemented it using Spring Boot?

Swagger allows you to describe the structure of your APIs so that machines can read them. The ability of APIs to describe their own structure is the root of all awesomeness in Swagger. Why is it so great? Well, by reading your API's structure, we can automatically build beautiful and interactive API documentation. We can also automatically generate client libraries for your API in many languages and explore other possibilities like automated testing. Swagger does this by asking your API to return a YAML or JSON that contains a detailed description of your entire API. This file is essentially a resource listing of your API which adheres to OpenAPI Specification. The specification asks you to include information like:

- What are all the operations that your API supports?
- What are your API's parameters and what does it return?
- Does your API need some authorization?
- And even fun things like terms, contact information and license to use the API.

Yes i have implemented it using Spring Boot.