Second Week Homework

1 – IOC and DI means ?

➤ With inversion of control(IOC), the flow depends on the object graph that is instantiated by the assembler and is made possible by object interactions being defined through abstractions.

➤ The binding process is achieved through dependency injection, although some argue that the use of a service locator also provides inversion of control.

➤ Dependency injection is a pattern used to create instances of objects that other objects rely on without knowing at compile time which class will be used to provide that functionality.

➤ IoC relies on dependency injection because a mechanism is needed in order to activate the components providing the specific functionality.

2 – Spring Bean Scopes ?

➤ When you create a bean definition what you are actually creating is a recipe for creating actual instances of the class defined by that bean definition. The idea that a bean definition is a recipe is important, because it means that, just like a class, you can potentially have many object instances created from a single recipe.

➤ You can control not only the various dependencies and configuration values that are to be plugged into an object that is created from a particular bean definition, but also the scope of the objects created from a particular bean definition.

Scope	Description
singleton	Scopes a single bean definition to a single object instance per Spring IoC container.
prototype	Scopes a single bean definition to any number of object instances.
request	Scopes a single bean definition to the lifecycle of a single HTTP request; that is each and every HTTP request will have its own instance of a bean created off the back of a single bean definition. Only valid in the context of a web-aware Spring ApplicationContext.
session	Scopes a single bean definition to the lifecycle of a HTTP Session. Only valid in the context of a web-aware Spring ApplicationContext.
global session	Scopes a single bean definition to the lifecycle of a global HTTP Session. Typically only valid when used in a portlet context. Only valid in the context of a web-aware Spring ApplicationContext.

3 – What does @SpringBootApplication do ?

➤ This annotation used to mark a configuration class that declares one or more @Bean methods and also triggers auto-configuration and component scanning. It's same as declaring a class with @Configuration, @EnableAutoConfiguration and @ComponentScan annotations.

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

➤ Aspect Oriented Programming (AOP) compliments OOPs in the sense that it also provides modularity. But the key unit of modularity is aspect than class.

➤ A cross-cutting concern is a concern that can affect the whole application and should be centralized in one location in code as possible, such as transaction management, authentication, logging, security etc.

➤ It provides the pluggable way to dynamically add the additional concern before, after or around the actual logic.

➤ It is widely used to provide declarative enterprise services such as declarative transaction management and it allows users to implement custom aspects.

5 – What is Singleton and where to use it ?

➤ A singleton is a class that allows only a single instance of itself to be created and gives access to that created instance. It contains static variables that can accommodate unique and private instances of itself.

➤ It is used in scenarios when a user wants to restrict instantiation of a class to only one object. This is helpful usually when a single object is required to coordinate actions across a system.

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

➤ Spring Boot Actuator is a sub-project of the Spring Boot Framework. It includes a number of additional features that help us to monitor and manage the Spring Boot application.

➤ 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.

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

➤ Spring is an open-source lightweight framework, Spring Boot is a microservice-based framework.

➤ Most important feature of the Spring Framework is dependency injection(removes dependencies from computer code, making the application easier to maintain and test) and for the Spring Boot it is Autoconfiguration(speed up and simplify development by removing the need to define some beans).

➤ Spring framework helps to create a loosely coupled application and Spring Boot helps to create a stand-alone application.

➤ In Spring application it requires a deployment descriptor and does not provide support for the in-memory database.

➤ Spring Boot doesn’t need deployment descriptor and provides support for the in-memory database such as H2.

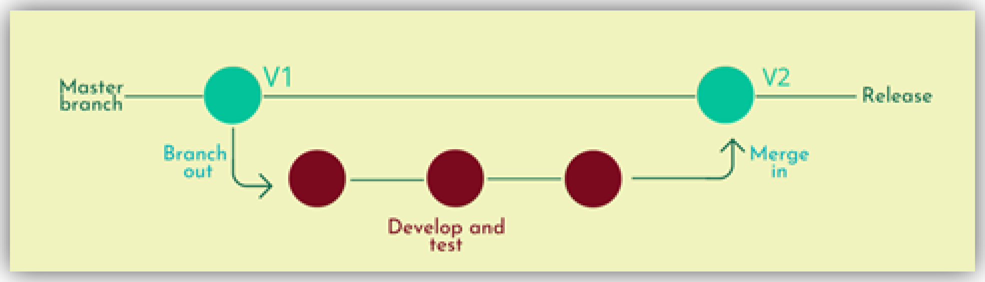
➤ In Spring framework you have to build configurations manually but in Spring Boot there are default configurations. It is easier to build and reduces the lines of code.

8 – Why to use VCS ?

➤ It is a abbreviation for version control systems and they are category of software tools that helps in recording changes made to files by keeping a track of modifications done to the code.

➤ Allow us to easily track the project and keeping the history by storing older versions.

➤ It is simple to collaborate with teammates and provides an environment for CI/CD tools.



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

➤ SOLID is a abbreviation of initials and has five principles.

➤ S - Single responsibility principle > Classes(same can be said for methods) have only one defined responsibility. Example: We have a Spring project for Student Management System which contains course contents and grades. We will have a class name Student and it can only have student related datas. If we add method name 'grades' it breaks the principle.

➤ O - Open closed principle > Classes (same can be said for methods) should be closed to change but open to adding new behaviors. Example: In this project, we will have course constructor method in the Course class. The contents must not change and new feature can be easily added to this method.

➤ L - Liskov substitution principle > We should be able to use subclasses in place of superclasses without making any changes to the code. Example: We will have super class for academics. This super class will have subclasses named Lecturer, Professor etc. We have to sustain subclasses without changing the superclass.

➤ I - Interface segregation principle > Interfaces should be split into a set of actions only what responsibility it requires. Example: We can make interface for academics and this interface can't have ony other feature than academician properties.

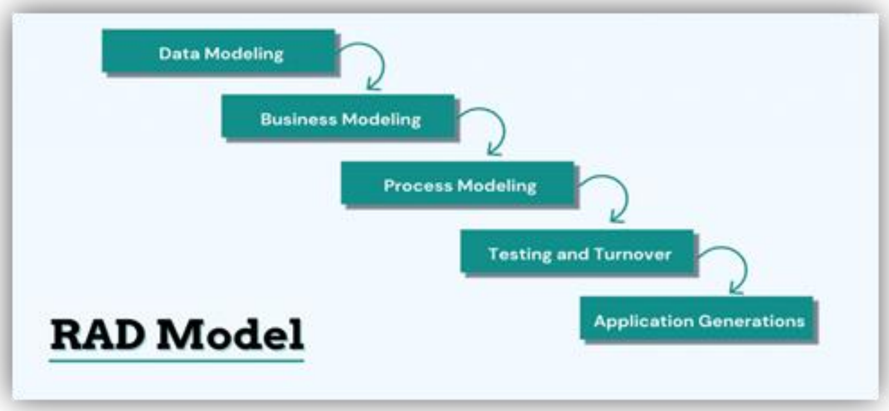
➤ D — Dependency Inversion Principle > Dependencies between classes should be as low as possible, and superclasses should not depend on subclasses. Example: Code addition on the Lecturer class shouldn't be effect the superclass.

10 - What is RAD model ?

➤ RAD Model or Rapid Application Development model is a software development process based on prototyping without any specific planning.

➤ In RAD model, there is less attention paid to the planning and more priority is given to the development tasks. It targets at developing software in a short span of time.

➤ It focuses on input-output source and destination of the information. It emphasizes on delivering projects in small pieces; the larger projects are divided into a series of smaller projects.



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

➤ Spring Boot Starters are dependency descriptors that can be added under the <dependencies> section in pom.xml. There are around 50+ Spring Boot Starters for different Spring and related technologies.

➤ It increase the productivity by decreasing the Configuration time for developers.

➤ Tested, Production-ready and supported dependency configurations.

➤ No need to remember the name and version of the dependencies.

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

➤ Caching is the process of storing copies of files in a cache, or temporary storage location, so that they can be accessed more quickly.

➤ We can enable caching in the Spring Boot by using the annotation @EnableCaching.It is defined in org.springframework.cache.annotation package. It is used together with @Configuration class.

➤ The auto-configuration enables caching and setup a CacheManager, if there is no already defined instance of CacheManager.

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

➤ Log is a file that records either events that occur in an operating system or other software runs, or messages between different users of a communication software.

➤ Logging is the act of keeping a log. In the simplest case, messages are written to a single log file.

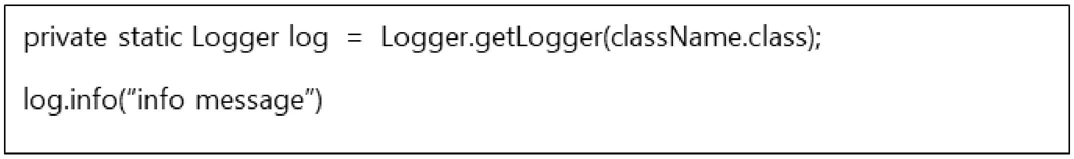
➤ A log can be useful for keeping track of computer use, emergency recovery, and application improvement. Each software program that is capable of creating a log has different methods of starting or stopping the log creation.

➤ In Spring Boot we can use Java Util Logging, Commons Logging, Log4J, or SLF4J.

➤ There are different types of logs for the cases: ERROR, WARN, INFO, DEBUG or TRACE.

➤ If the log is the only detailed source of information available to the support team when trying to diagnose a failure, then it probably needs to be quite detailed and should be used.

➤ After adding the depency all you have to do is create a Logger object.



```
private static Logger log  =  Logger.getLogger(className.class);

log.info("info message")
```

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

➤ Swagger is an open source set of rules, specifications and tools for developing and describing RESTful APIs. The Swagger framework allows developers to create interactive, machine and human-readable API documentation.

➤ All we have to do is add Spring Swagger depency(called SpringFox) and enabling Swagger in the class with @EnableSwagger2 annotation.

➤ Configure the Swagger and navigate to the “*http://localhost:8080/swagger-ui.html”.*