HOMEWORK2
FATMA BETÜL UYAR

1 – IOC and DI means ?

IOC(Inversion of Control) is a principle in software. It transfers the control of objects or portions of a program to a container or framework.IoC enables a framework to take control of the flow of a program and make calls to our custom code.

Benefits of IoC;

- 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

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

```
public class Store {
    private Item item;

    public Store() {
        item = new ItemImpl1();
    }
}
```

In the example above, we need to instantiate an implementation of the *Item* interface within the *Store* class itself.

By using DI, we can rewrite the example without specifying the implementation of the *Item* that we want

```
public class Store {
    private Item item;
    public Store(Item item) {
        this.item = item;
    }
}
```

```
@Configuration
public class AppConfig {

    @Bean
    public Item item1() {
        return new ItemImpl1();
    }

    @Bean
    public Store store() {
        return new Store(item1());
    }
}
```

Constructor-Based Dependency Injection

The container will invoke a constructor with arguments each representing a dependency we want to set.

```
@Bean
public Store store() {
    Store store = new Store();
    store.setItem(item1());
    return store;
}
```

Setter-Based Dependency Injection

the container will call setter methods of our class after invoking a no-argument constructor or no-argument static factory method to instantiate the bean.

```
public class Store {
    @Autowired
    private Item item;
}
```

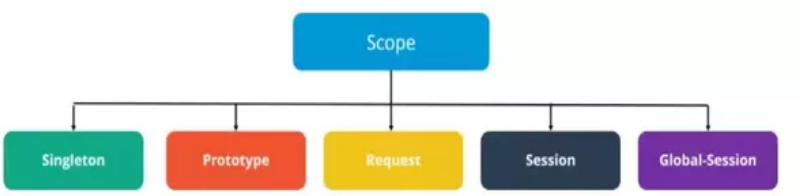
Field-Based Dependency Injection

we can inject the dependencies by marking them with an *@Autowired* annotation

2 – Spring Bean Scopes ?

Bean Scope controls the instance creation of the bean and it is managed by the spring container. The following are the five scopes provided for a bean:

The latest version of the Spring framework defines **6** types of scopes:



- singleton
- prototype
- request
- session
- application
- websocket

By default, the scope of a bean is a **singleton**.

Singleton: Only one instance will be created for a single bean definition per Spring IoC container and the same object will **be shared for each request** made for that bean.

Proptype: A new instance will be created for a single bean definition **every time a request** is made for that bean.

Request: A new instance will be created for a single bean definition **every time an HTTP request** is made for that bean.

Session: Scopes a single bean definition to the lifecycle of an **HTTP Session.**

Global-Session: Scopes a single bean definition to the lifecycle of a **global HTTP Session**.

Note: *Request, Session and Global-Session only valid in the context of a web-aware Spring ApplicationContext.*

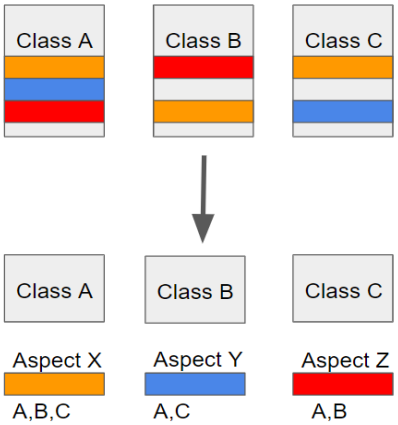
3 – What does @SpringBootApplication do ?

The **entry point** of the Spring Boot Application is the class containing @SpringBootApplication annotation. This class should have the main method to run the Spring Boot application. @SpringBootApplication annotation **includes Auto- Configuration**, **Component Scan,** and **Spring Boot Configuration**.

If you added @SpringBootApplication annotation to the class, you do not need to add the @EnableAutoConfiguration, @ComponentScan and @SpringBootConfiguration annotation.

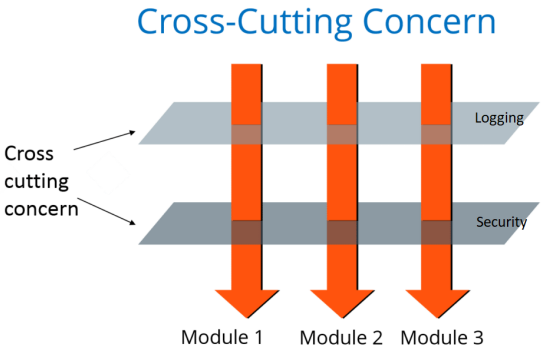
The @SpringBootApplication annotation includes all other annotations.

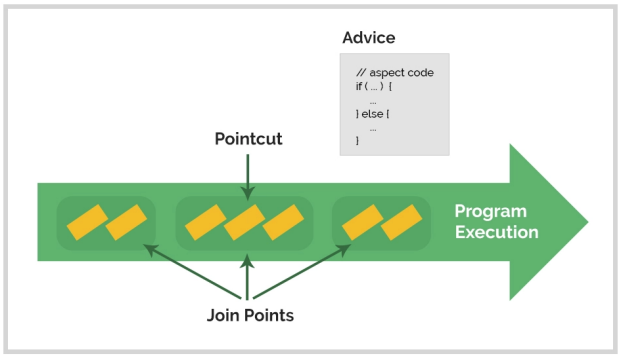
4 – What is Spring AOP ? Where and How to use it ?



AOP is a programming paradigm that aims to increase modularity by allowing the **separation of cross-cutting concerns**.

Cross-cutting concerns are parts of a program that rely on or must affect many other parts of the system such as **Logging**, **Caching**, **Transaction**, **Exception Handling.**
These concerns can be in every layer and AOP aims to separate these from the application's business logic.





Aspect:

This is a module which has a set of APIs providing cross-cutting requirements. For example, a logging module would be called AOP aspect for logging.

Join Point:

A *Joinpoint* is a point during the execution of a program, such as the execution of a method or the handling of an exception. (**public String getName()**)

Advice:

This is the actual action to be taken either before or after the method execution. (**getNameAdvice()**)

Pointcut:

This is a set of one or more join points where an advice should be executed. **execution(public String getName())**

5 – What is Singleton and where to use it ?

Singleton is a creational design pattern that lets you ensure that **a class has only one instance**, while providing a **global access point to this instance**.

All implementations of the Singleton have these two steps in common;

- Make the default constructor private, to prevent other objects from using the `new` operator with the Singleton class.
- Create a static creation method that acts as a constructor. Under the hood, this method calls the private constructor to create an object and saves it in a static field. All following calls to this method return the cached object.

If your code has access to the Singleton class, then it's able to call the Singleton's static method. So whenever that method is called, the same object is always returned.

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

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.

Actuator is mainly used to **expose operational information about the running application.**

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

The most important feature of the Spring Framework is **dependency injection**.

It helps to create feature of **loosely coupled application**.

To run the Spring application, we need to **set the server** explicitly.

The most important feature of the Spring Boot is **Autoconfiguration.**

It helps to create a **stand-alone application**.

Spring Boot provides **embedded servers** such as Tomcat and Jetty etc.

8 – Why use VCS ?

Version control is important to **keep track of changes** and keep every team member working on the right version. You should use version control software for all code, files, and assets that multiple team **members** will **collaborate** on.

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.

When you need to **troubleshoot an issue**, you can easily access and **compare the last working file with the faulty file**, and decrease the time spent identifying the cause of an issue.

You can **restore older versions** of a file (or even the whole project) effectively through the use of version control systems. You can simply undo every commit you have done in just a few clicks.

A distributed VCS acts as a **backup**. If a central server goes down, a developer can retrieve one of his/her teammates' local VCS repository.

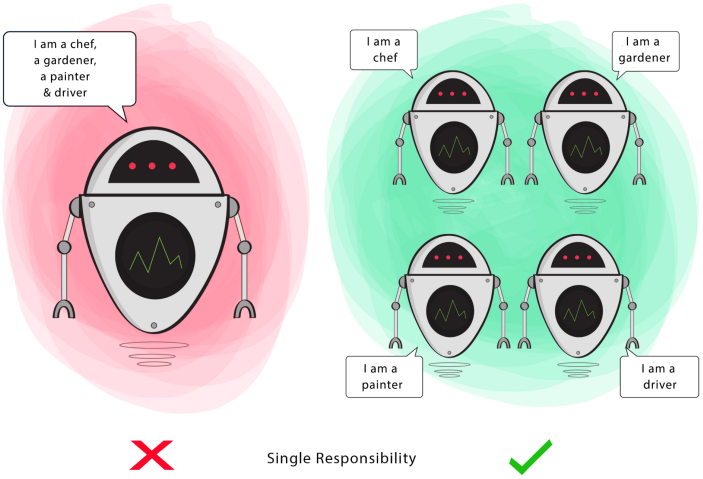
9 – What are SOLID Principles ? Give sample usages in Java ?

Single Responsibility
Open Closed
Liskov Substitution
Interface Segregation

Dependency Inversion

“

as



Single Responsibility

Single Responsibility

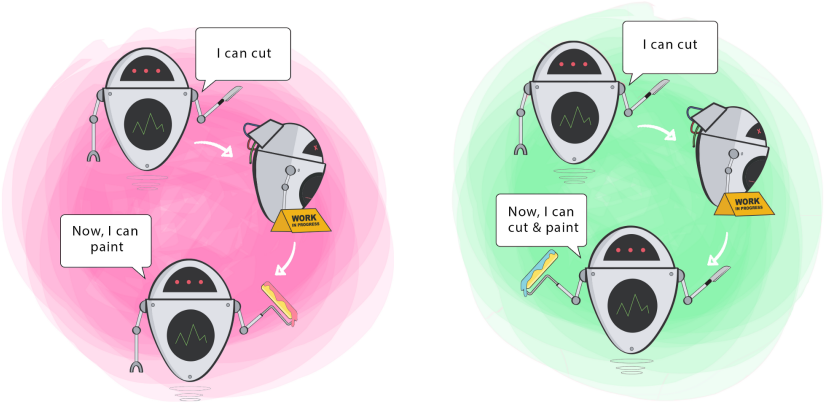
A class should have a single responsibility. ”

This principle aims to **separate behaviors** so that if bugs arise a result of your change, it won’t affect other unrelated behaviors.

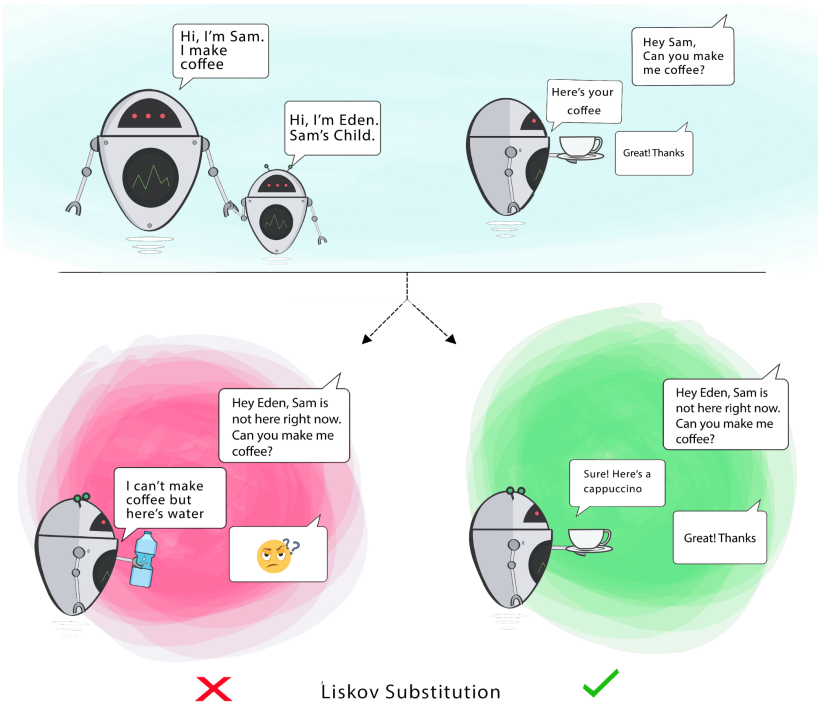
Open-Closed

“ Classes should be open for extension, but closed for modification. ”

This principle aims to extend Class’s behavior without changing the existing behavior of that Class.



Open-Closed

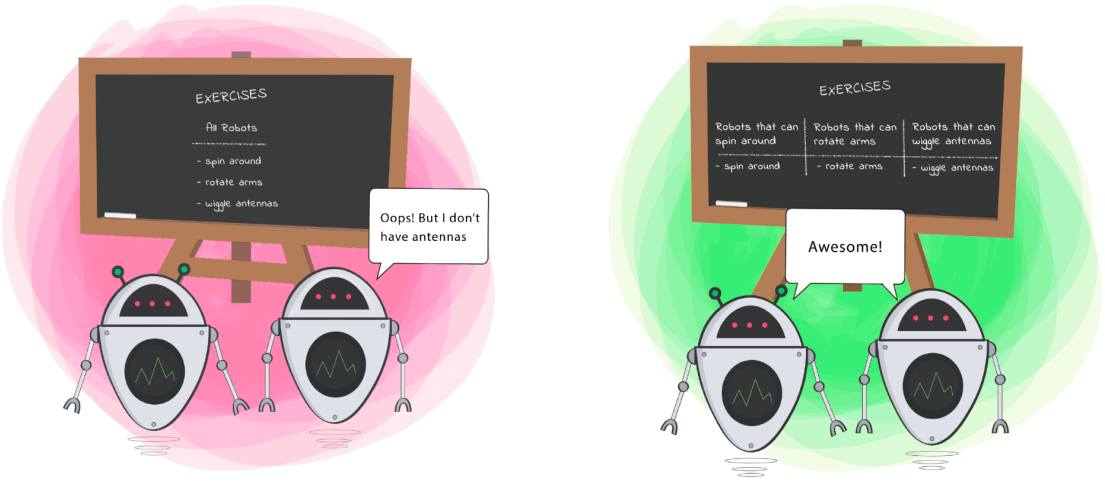


Liskov Substitution

Liskov Substitution

“ If S is a subtype of T, then objects of type T in a program may be replaced with objects of type S without altering any of the desirable properties of that program.”

This principle aims to enforce consistency so that the parent **Class or its child Class can be used in the same way** without any errors.

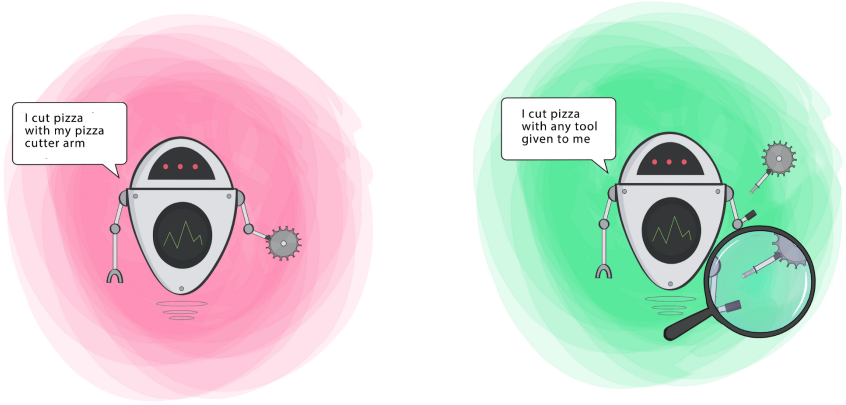


Interface Segregation

Interface Segregation

" Clients should not be forced to depend on methods that they do not use."

This principle aims at splitting a set of actions into smaller sets so that a Class executes ONLY the set of actions it requires.



Dependency Inversion

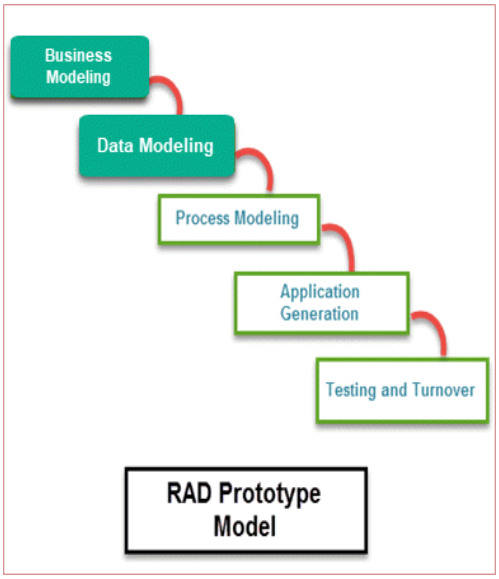
Dependency Inversion

" - High-level modules should not depend on low-level modules. Both should depend on the abstraction.
- Abstractions should not depend on details. Details should depend on abstractions."

This principle aims at reducing the dependency of a high-level Class on the low-level Class by introducing an interface.

10 - What is the RAD model ?

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



Business Modeling:
On basis of the flow of information and distribution between various business channels, the product is designed

Data Modeling:
The information collected from business modeling is refined into a set of data objects that are significant for the business

Process Modeling:
The data object that is declared in the data modeling phase is transformed to achieve the information flow necessary to implement a business function

Application Generation:
Automated tools are used for the construction of the software, to convert process and data models into prototypes

Testing and Turnover:
As prototypes are individually tested during every iteration, the overall testing time is reduced in RAD.

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**.
These starters give **all the dependencies** under **a single name**. For example, if you want to use Spring Data JPA for database access, you can include **spring-boot-starter-data-jpa** dependency.
The advantages of using Starters are as follows:

- Increase productivity by decreasing the Configuration time for developers.
- Managing the POM is easier since the number of dependencies to be added is decreased.
- 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 achieve caching in Spring Boot ?

A **cache** is a software or hardware component aimed at **storing data** so that future requests for the **same data can be served faster**.
Caching is a mechanism to improve the performance of any type of application.Technically, caching is the **process of storing** and **accessing data from a cache**.
Caching is important because it allows developers to achieve performance improvements, sometimes considerably.
Another important aspect of caching is that it allows us to **avoid** making **new requests or reprocessing** data **every time**. So that we can avoid overhead, like network overhead, and reduce CPU usage, especially if requests involve complex elaborations.
We can enable caching **in the Spring Boot** application by using the annotation **@EnableCaching.**

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

Logging is keeping a **record of all data** input, processes, data output, and final results in a program. Useful logs can make it **easier** for developers to **understand**.

Where logging takes care of **recording the events** that happen during runtime (method calls, input/outputs, HTTP calls, SQL executions), the audit logging is responsible for recording more abstract, business logic events.

There are multiple reasons why we may need to capture the application activity.

- **Recording unusual** circumstances or **errors** that may be happening in the program
- **Getting the info** about **what's going** in the application

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 Swagger framework allows developers to create interactive, machine and **human-readable API** documentation.

API specifications typically include information such as supported operations, parameters and outputs, authorization requirements, available endpoints and licenses needed.

Swagger can generate this **information** automatically from the source code by asking the API to return a documentation file from its annotations.

Yes, we have implemented it.

References:

https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring
https://www.geeksforgeeks.org/singleton-and-prototype-bean-scopes-in-java-spring/
https://kubilaycicek.medium.com/spring-bean-scope-kavramlar%C4%B1-38f79a042f66
https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm
https://medium.com/@eomercelik/spring-aop-31c57c163305#:~:text=AOP'nin%20temel%20oda%C4%9F%C4%B1%2C%20birbirleriyle%20ke%C5%9Fisen%20davran%C4%B1%C5%9Flar%C4%B1n%20ayr%C4%B1lmas%C4%B1%20%C3%BCzerinedir.,onu%20desteklemek%2C%20tamamlamak%20i%C3%A7in%20geli%C5%9Ftirilmi%C5%9Ftir.&text=her%20s%C4%B1n%C4%B1f%C4%B1n%20sadece%20kendi%20sorumlulu%C4%9Funu%20yerini%20getirmesini%20s%C3%B6yler.
https://www.tutorialspoint.com/spring/aop_with_spring.htm
https://refactoring.guru/design-patterns/singleton
https://medium.com/@lanceharvieruntime/version-control-why-do-we-need-it-1681f4888cec
https://www.guru99.com/what-is-rad-rapid-software-development-model-advantages-disadvantages.html
https://auth0.com/blog/what-is-caching-and-how-it-works/
https://searchapparchitecture.techtarget.com/definition/Swagger