Exercise (Spring Container)

What is the Output :

Q1 :

```
1 usage
@SpringBootApplication
public class SpringPollApplication {

    public static void main(String[] args) { SpringApplication.run(SpringPollApplication.class, args); }


    @Bean
    public String getMessage1(){
        System.out.println("hey from message1");
        return "1";
    }
}
```

Answer :

This screenshot is a code that looks like it is in a layer that called SpringPollApplication and it looks like it is the first layer that make us be able to make another layer on top for example the Controller Layer.

Under the main method we have a method called getMessage1 , this method is a String method that return 1 to be stored in the Spring container , this method prints a String “hey from message1” which its output we will see down the screen when we run the code.

The reason why we can do it is because of the @Bean annotation which is considered as the foundation for all @ annotations as it makes the method readable and seen to the Spring framework.

There is no possible order since there is only one method in this code.

Q2 :

```
1 usage
@SpringBootApplication
public class SpringPollApplication {

    public static void main(String[] args) { SpringApplication.run(SpringPollApplication.class, args); }


    @Bean
    @Qualifier("1")
    public String getMessage1(){
        System.out.println("hey from message1");
        return "1";
    }
    @Bean
    public String getMessage2(@Qualifier("1") String data ){
        System.out.println("hey from message2");
        return data ;
    }
```

Answer :

This screenshot is a code that looks like it is in a layer that called SpringPollApplication and it looks like it is the first layer that make us be able to make another layer on top for example the Controller Layer.

Under the main method we have 2 methods :

First one called getMessage1 , this method is a String method that return 1 to be stored in the Spring container , this method prints a String “hey from message1” which its output we will see down the screen when we run the code. This method also has a @Qualifier annotation above it with a key (“1”) which will make this a priority and qualified to be printed first since it is a method without a parameter.

Seconed one called getMessage2 , this method is a String method that return data (which is equal to the value of the parameter it will get from the Spring container), this method prints a String “hey from message2” which its output we will see down the screen when we run the code after getMessage1 method output. This method also has a @Qualifier annotation next to the parameter which will make this a second priority and qualified to be printed second right after getMessage1 , and the reason behind it is the @Qualifier annotation, this annotation makes like an order of who go next based on the key and value.

Possiple order since there is only 2 meathod is always gonna be getMessage1 then getMessage2

Q3 :

```java
@Bean
@Qualifier("1")
public String getMessage1(){
    System.out.println("hey from message1");
    return "1";
}

@Bean
@Qualifier("2")
public String getMessage2(@Qualifier("3") String data ){
    System.out.println("hey from message2");
    return data;
}

@Bean
@Qualifier("3")
public String getMessage3(){
    System.out.println("hey from message3");
    return "3" ;
}
```

Answer :

This screenshot is a code that have 3 methods :

First one called getMessage1 , this method is a String method that return 1 to be stored in the Spring container , this method prints a String “hey from message1” which its output we will see down the screen when we run the code. This method also has @Qualifier annotation above it with a key (“1”) which will make this bean registered with the name (“1”) and it does not depend on any other method.

Seconed one called getMessage2 , this method is a String method that return data (which is equal to the value of the bean with @Qualifier(“3”)) from the Spring container , this method prints a String “hey from message2” which its output we will see down the screen when we run the code. This method also has a @Qualifier annotation next to the parameter which will make this method depend on getMessage3 , because it needs its value first.

Third one called getMessage3 , this method is a String method that return 3 to be stored in the Spring container , this method prints a String “hey from message3” which its output we will see down the screen when we run the code. This method also has a @Qualifier annotation above it with a key (“3”) which will make this a priority and qualified to be printed first since it is a method without a parameter.

Possiple methods order since there is three :

1- getMessage3 ------ > then getMessage2 ------ > then getMessage1
2- getMessage1 ------ > then getMessage3 ------ > then getMessage2
3- getMessage3 ------ > then getMessage1 ------ > then getMessage2

Q4 :

```java
@Bean
@Qualifier("1")
public String getMessage1(){
    System.out.println("hey from message1");
    return "1";
}

@Bean
@Qualifier("2")
public String getMessage2(@Qualifier("3") String data ){
    System.out.println("hey from message2");
    return data;
}

@Bean
@Qualifier("3")
public String getMessage3(){
    System.out.println("hey from message3");
    return "3" ;
}
```

Answer :

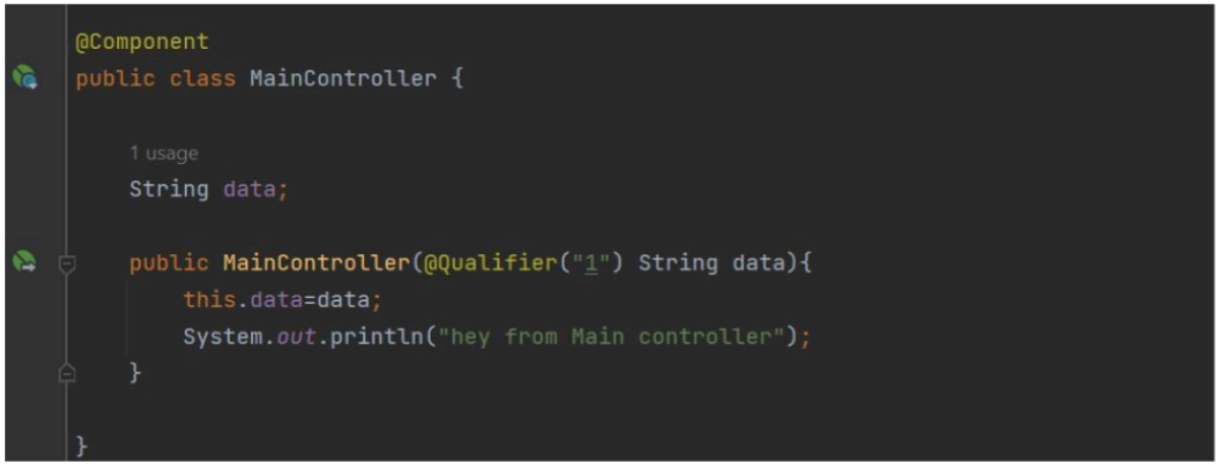
This screenshot is a code that have 3 methods :

First one called getMessage1 , this method is a String method that return 1 to be stored in the Spring container , this method prints a String “hey from message1” which its output we will see down the screen when we run the code . this method also has @Qualifier annotation above it with a key (“1”) which will make this bean registered with the name (“1”) and it does not depend on any other bean .

Seconed one called getMessage2 , this method is a String method that return data (which is equal to the value of @Qualifier(“3”)) from the Spring container , this method prints a String “hey from message2” which its output we will see down the screen when we run the code . this method also has a @Qualifier annotation next to the parameter which will make this method depend on getMessage3 , because it needs its value first .

Third one called getMessage3 , this method is a String method that return 3 to be stored in the Spring container , this method prints a String “hey from message3” which its output we will see down the screen when we run the code . this method also has @Qualifier annotation above it

with a key (“3”) which will make this a priority and qualified to be printed first since it is a method without a parameter .



```java
@Component
public class MainController {

    1 usage
    String data;

    public MainController(@Qualifier("1") String data){
        this.data=data;
        System.out.println("hey from Main controller");
    }

}
```

This screenshot also has another class called MainController , this class have a @Component annotation which make it a bean in the Spring container . inside it we have a variable String data and a constructor that have @Qualifier(“1”) next to its parameter , which mean it will take the value from getMessage1 bean . this constructor also prints “hey from Main controller” when it run .

Only Possiple order :

1- getMessage3 ------ > then getMessage2------ > then getMessage1 ------ > mainController
2- getMessage3 ------ > then getMessage2 ------ > mainController ------ > then getMessage1
3- getMessage1 ------ > mainController ------ > then getMessage3 ------ > then getMessage2
4- getMessage2 ------ > then getMessage3------ > then getMessage1 ------ > mainController

Q5 :

```java
@Bean
@Qualifier("1")
public String getMessage1(MainController mainController){
    System.out.println("hey from message1");
    return "1";
}

@Bean
@Qualifier("2")
public String getMessage2(@Qualifier("3") String data ){
    System.out.println("hey from message2");
    return data;
}

@Bean
@Qualifier("3")
public String getMessage3(){
    System.out.println("hey from message3");
    return "3" ;
}
```

Answer :

First one called getMessage1 , this method is a String method that returns 1 to be stored in the Spring container , this method prints a String “hey from message1” which its output we will see down the screen when we run the code. This method also has a @Qualifier annotation above it with a key (“1”) and it has a parameter called MainController mainController , which means that before message1 runs, Spring will first create the MainController object because it is needed inside this method.

Seconed one called getMessage2 , this method is a String method that returns data (which is equal to the value of @Qualifier(“3”)) from the Spring container , this method prints a String “hey from message2” which its output we will see down the screen when we run the code. This method depends on getMessage3 because of the @Qualifier(“3”) next to the parameter.

Third one called getMessage3 , this method is a String method that returns 3 to be stored in the Spring container , this method prints a String “hey from message3” which its output we will see down the screen when we run the code. This method has a @Qualifier annotation above it with a key (“3”) and since no other bean depends on it, it will be created first before getMessage2.

```java
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Component;

1 usage
@Component
public class MainController {

    1 usage
    String data;

    public MainController(@Qualifier("2") String data){
        this.data=data;
        System.out.println("hey from Main controller");
    }

}
```

This screenshot also has another class called MainController , this class has a @Component annotation which makes it a bean in the Spring container . inside it we have a variable String data and a constructor that has @Qualifier(“2”) next to its parameter , which means it will take the value from getMessage2 bean . this constructor also prints “hey from Main controller” when it run .

Only Possiple order :

1- getMessage3 ------ > then getMessage2 ------ > mainController------ > then getMessage1

Refrence :

https://medium.com/%40AlexanderObregon/controlling-bean-initialization-order-in-spring-boot-applications-ceaa977bf7fe
https://stackoverflow.com/questions/56642356/when-to-use-qualifier-and-primary-in-spring?utm_source=chatgpt.com
https://docs.spring.io/spring-framework/reference/core/beans/java/bean-annotation.html?utm_source=chatgpt.com