Laborator 2: Annotation Configuration

@Component, @Qualifier, @Primary

@Component este utilizată împreună cu class-path scanning pentru înregistrarea bean-urilor în context.
Elementul Value se completează cu Id-ul bean-ului.
Id-ul default este numele clasei cu prima literă mică.

Exemple de subtipuri ale **@Component**

- ❑ @Service
- ❑ @Repository
- ❑ @Controller

1. Importați în IntelliJ proiectul lab2_start. Din meniu selectați File – New project from existing sources. Rulați mvn clean install, adăugați o configurație pentru a rula mvn clean install.

2. Adnotați clasa SportSubscription @Component(“mySportSubscription”).

```
@Component("mySportSubscription")
public class SportSubscription implements Subscription
```

3. Adnotați clasa SportSubscription @Component. Ce ar trebui modificat în testul testXmlContext?

4. Adnotați clasele BooksSubscription și DicountCalculator @Component. Adnotați constructorul clasei BooksSubscription cu @Autowired.

```
@Autowired
public BooksSubscription(DiscountCalculator  discountCalculator)
{
    this.discountCalculator = discountCalculator;
}
```

5. Adăugați o clasa care va conține testul `constructorDITest`.

```
public class DITest {

    @Test
    public void constructorDITest(){
        ClassPathXmlApplicationContext context =
                new
ClassPathXmlApplicationContext("applicationContext.xml");
        Subscription theSubscription =
context.getBean("booksSubscription", Subscription.class);
        System.out.println(theSubscription.getPrice() + " "
                + theSubscription.getDescription());
        context.close();
    }
}
```

6. Adnotați clasa MoviesSubscriptions @Component și metoda setDiscountCalculator @Autowired. Adăugați testul setterDITest. Se poate modifica denumirea metodei setDiscountCalculator. Se vor injecta argumentele pentru **orice metode** ale clasei adnotate @Autowired.

```
@Autowired
    public void setDiscountCalculator(DiscountCalculator
discountCalculator) {
        this.discountCalculator = discountCalculator;
    }
```

```
@Test
public void setterDITest(){
        ClassPathXmlApplicationContext context =
                new
ClassPathXmlApplicationContext("applicationContext.xml");
        Subscription theSubscription =
context.getBean("moviesSubscription", Subscription.class);
        System.out.println(theSubscription.getPrice() + " "
                + theSubscription.getDescription());
        context.close();
}
```

7. Adăugați în clasa SportSubscription atributul discountCalculator. Testați.

```
@Autowired
DiscountCalculator discountCalculator;
```

```
@Test
public void fieldDITest(){
        ClassPathXmlApplicationContext context =
                new
ClassPathXmlApplicationContext("applicationContext.xml");
        Subscription theSubscription =
context.getBean("sportSubscription", Subscription.class);
        System.out.println(theSubscription.getPrice() + " "
                + theSubscription.getDescription());
        context.close();
}
```

@Qualifier, @Primary sunt utilizate atunci când pentru o dependență sunt găsite mai multe implementări posibile.
@Primary va specifica implementarea care va fi folosită default.
@Qualifier permite specificarea id-ului dependenței care va fi injectată

8. Adăugați o nouă implementare a interfeței DiscountCalculator. Testați.

```
@Component
public class FixDiscount implements DiscountCalculator{

    public double calculate(int price) {
        return 0.85 * price;
    }
}
```

9. Adnotați @Primary una dintre implementările intervеței DiscountCalculator

```
@Component
@Primary
public class FixDiscount implements DiscountCalculator{

    public double calculate(int price) {
        return 0.85 * price;
    }
}
```

10. Adnotați metoda setDiscountCalculator a clasei MoviesSubscription @Qualifier(“discountCalculatorImp”).

```
@Autowired
@Qualifier("discountCalculatorImpl")
public void setDiscountCalculator(DiscountCalculator
discountCalculator) {
        this.discountCalculator = discountCalculator;
}
```

11. Adnotați argumentul constructorului clasei BooksSubscription:

```
@Autowired
public BooksSubscription(@Qualifier("discountCalculatorImpl")
DiscountCalculator discountCalculator)
```

12. Configurați în fișierul **applicationContext.xml** fișierul de proprități application.properties

```
<context:property-placeholder
location="classpath:application.properties" />
```

13. Adnotați atributul @Value percent al clasei DiscountCalculatorImp.

```
@Value("0.2")
private double percent;

@Value("${discount.percent}")
private double percent;
```

14. Adăugați adnotări @Scope în clasele BooksSubscription și SportSubscription. Adăugați adnotari @Test metodelor din clasa LifeCycleTest

```
@Component
@Scope("singleton")
public class SportSubscription implements Subscription{

@Component
@Scope("prototype")
public class BooksSubscription implements Subscription{
```

@PostConstruct, @PreDestroy sunt utilizate pentru configurarea metodelor care se vor apela după inițializarea bean-ului respectiv înainte de distrugerea bean-ului.

15. Adnotați metodele initSportSubscription și destroySportSubscription:

```
@PostConstruct
 void initSportSubscription(){
       System.out.println(">>> init sport subscription");
   }

@PreDestroy
 void destroySportSubscription(){
       System.out.println(">>> destroy sport subscription");
   }
```

16. Adăugați clasa de configurare SubscriptionConfig.

```
@Configuration
@ComponentScan("com.apbdoo.lab2")
public class SubscriptionConfig {

}
```