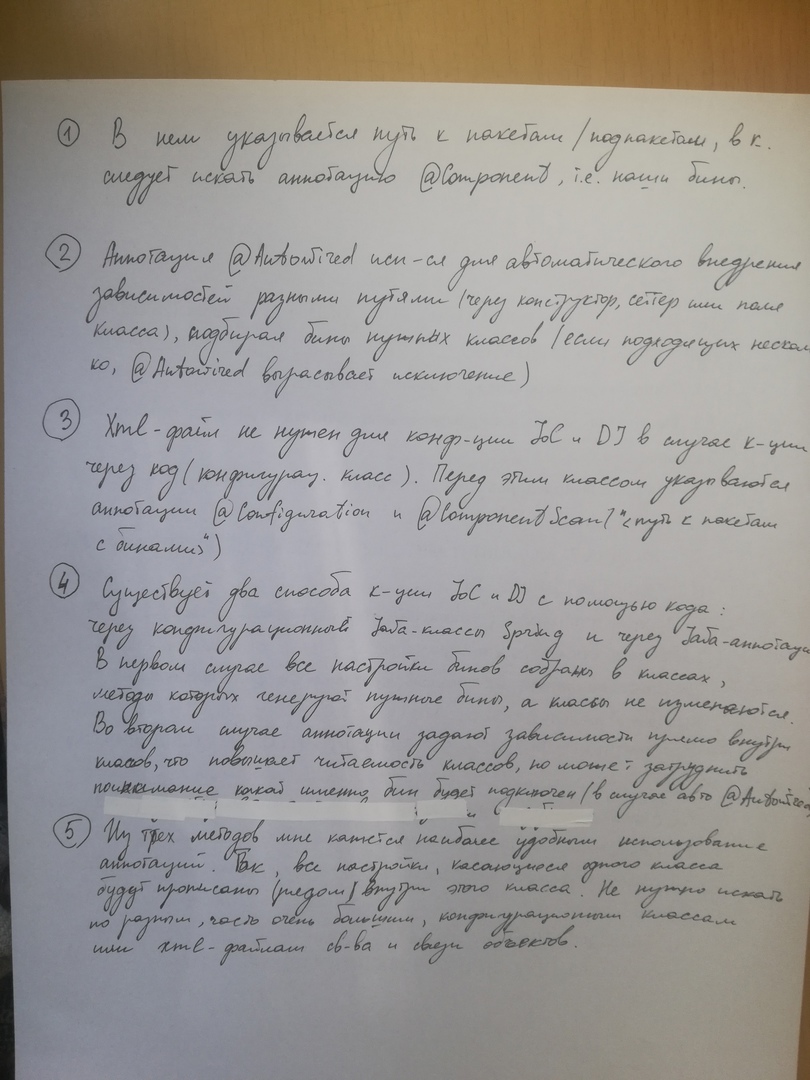
**ЛАБОРАТОРНАЯ РАБОТА №1**

**«**Inversion of Control и Dependency Injection с помощью аннотаций и кода **»**

Выполнила   
студентка группы ИВТ-1  
Гладкова К.

**Цель:** научиться применять аннотации для Inversion of Control и Dependency Injection. Создать класс, позволяющий реализовать IoC и DI с помощью кода.

**Контрольные вопросы:**

1. Опишите роль XML-файла при IoC и DI посредством аннотаций

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Какова роль аннотации @Autowired?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. В каком случае XML-файл не нужен для конфигурации IoC и DI?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Сколько способов конфигурации IoC и DI с помощью кода существует? Чем они отличаются?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Какой из трех методов конфигурации IoC и DI (XML-файл, аннотации, код) кажется вам наиболее удобным? Почему?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРИЛОЖЕНИЕ**

Листинг «MyConfig.java»

package com.example.lab2.ex3;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.PropertySource;

import org.springframework.context.annotation.Scope;

@Configuration

@PropertySource("classpath:myApp.properties")

public class MyConfig {

@Bean

@Scope("singleton")

public Pet dogBean () {

return new Dog();

}

@Bean

public Car carBean () {

return new Car();

}

@Bean

public Employee menBean () {

return new Employee(dogBean(), carBean());

}

}

Листинг «beans\_1.xml»

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd">

<bean id="myPet"

class="com.example.lab2.ex3.Dog"

init-method="init"

destroy-method="destroy">

</bean>

<bean id="oldCar"

class="com.example.lab2.ex3.Car"

init-method="init"

destroy-method="destroy">

</bean>

<bean id="lonelyMen"

class="com.example.lab2.ex3.Employee"

init-method="init"

destroy-method="destroy">

<constructor-arg name="pet" ref="myPet"/>

<constructor-arg name="car" ref="oldCar"/>

<property name="age" value="35"/>

<property name="name" value="Alee"/>

</bean>

</beans>

Листинг «beans\_2.xml»

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd">

<context:component-scan base-package="com.example.lab2.ex2"/>

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

</beans>

Листинг «Car.java»

package com.example.lab2.ex2;

import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

@Component("kiaCar")

public class Car {

public Car(){

System.out.println("Car was created");

}

@PostConstruct

public void init() {

System.out.println("Class Car: init method");

}

@PreDestroy

public void destroy() {

System.out.println("Class Car: destroy method");

}

public void drive(){

System.out.println(" ~ VROOM VROOM ~ ");

}

}

Листинг «Cat.java»

package com.example.lab2.ex2;

import org.springframework.context.annotation.Scope;

import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

@Component("lazyCat")

@Scope("prototype")

public class Cat implements Pet {

@PostConstruct

public void init() {

System.out.println("Class Cat: init method");

}

public Cat() {

System.out.println("Cat was created");

}

@PreDestroy

public void destroy() {

System.out.println("Class Cat: destroy method");

}

@Override

public void say() {

System.out.println("Meow~~meow");

}

}

Листинг «Employee.java»

package com.example.lab2.ex2;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;

import javax.annotation.PreDestroy;

@Component("newWorker")

public class Employee {

private Pet pet;

private Car car;

@Value("${employee.name}")

private String name;

@Value("${employee.age}")

private int age;

@Autowired

public Employee(@Qualifier("lazyCat") Pet pet, Car car) {

this.pet = pet;

this.car = car;

System.out.println("Employee was created and got pet and car");

}

public void setName(String name) {

this.name = name;

System.out.println("Employee get name \"" + name + "\"");

}

public void setAge(Integer age) {

this.age = age;

System.out.println("Employee get age \"" + age + "\"");

}

public int getAge() { return age; }

public String getName() { return name; }

@PostConstruct

public void init() {

System.out.println("Class Employee: init method");

}

@PreDestroy

public void destroy() {

System.out.println("Class Employee: destroy method");

}

public void restAtHome() {

System.out.println("[takes favourite toy to play with little friend]");

this.pet.say();

System.out.println("[both are filled with happiness]");

}

public void goToWork() {

System.out.println("[pours food to the pet]");

this.pet.say();

System.out.println("[close the door and start the engine]");

this.car.drive();

}

public void wentFromWork() {

this.car.drive();

System.out.println("[open the door]");

this.pet.say();

}

}