Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий
институт
Кафедра «Информатика»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №2

Внедрение зависимостей в Spring
тема

Преподаватель _____ А.С. Черниговский
подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031830504 _____ Е.В. Железкин
номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2020

1 Цель работы

Ознакомиться с механизмом внедрения зависимостей в Spring.

2 Задача работы

В каждом варианте есть сущность (класс), необходимо создать интерфейс (самостоятельно на усмотрение студента) и классы его имплементирующие. Объекты классов имплементирующих данный интерфейс будут передаваться в качестве зависимостей. Выполнить связывание иполучить объекты из контекста. В случае затруднений обратитесь к конспекту второй лекции курса.

Необходимо:

1) Реализовать внедрение простых значений через конструктор;
2) Реализовать внедрение зависимости по ссылке через конструктор;
3) Интерфейс должен содержать как минимум один метод;
4) Классы, имплементирующие интерфейс должны содержать как минимум одно поле (у разных классов разные);
5) Зависимый класс должен содержать метод, который бы на основе вызова метода у зависимости выводил бы некоторое сообщение в консоль (Например для класса Автомобиля, в который внедряются Двигатели. Двигатели могут выдавать свою мощность, а автомобиль может выводить сообщение с какой скоростью он может двигаться.)
6) Реализовать внедрение простых значений из внешнего файла через Setter

Вариант 9 - механизт.

3 Ход работы

1) Реализован интерфейс механизма (рисунок 1):

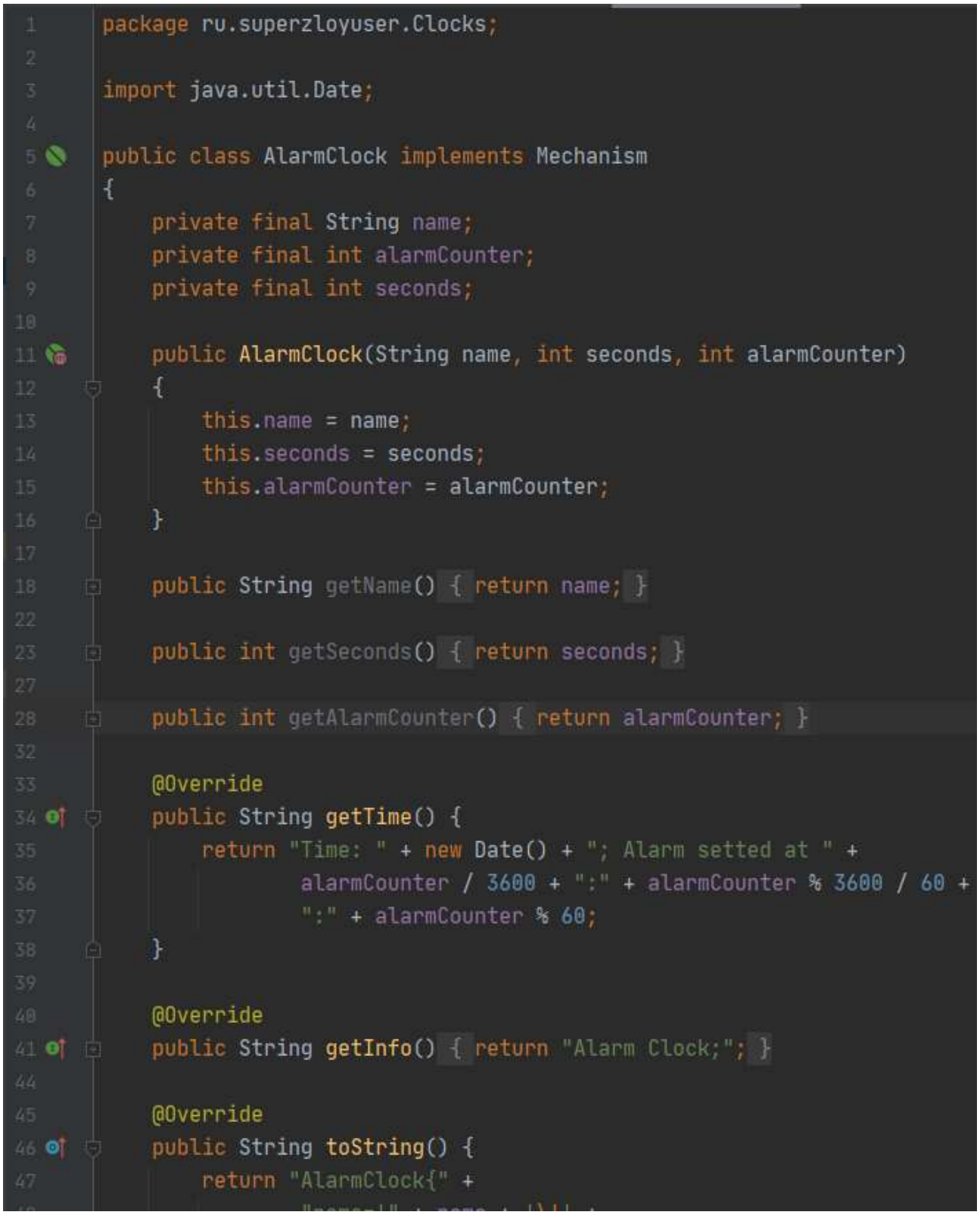
```
package ru.superzloyuser.Clocks;

public interface Mechanism
{
    String getTime();
    String getInfo();
}
```

Рисунок 1 – Интерфейс механизма, Mechanism.java

2) Реализованы классы, реализующие интерфейс (рисунки 2, 3):

Рисунок 2 – Структура классов



```java
package ru.superzloyuser.Clocks;

import java.util.Date;

public class AlarmClock implements Mechanism
{
    private final String name;
    private final int alarmCounter;
    private final int seconds;

    public AlarmClock(String name, int seconds, int alarmCounter)
    {
        this.name = name;
        this.seconds = seconds;
        this.alarmCounter = alarmCounter;
    }

    public String getName() { return name; }

    public int getSeconds() { return seconds; }

    public int getAlarmCounter() { return alarmCounter; }

    @Override
    public String getTime() {
        return "Time: " + new Date() + "; Alarm setted at " +
                alarmCounter / 3600 + ":" + alarmCounter % 3600 / 60 +
                ":" + alarmCounter % 60;
    }

    @Override
    public String getInfo() { return "Alarm Clock;"; }

    @Override
    public String toString() {
        return "AlarmClock{" +
```

Рисунок 3 – Один из классов – AlarmClock.java

3) Заполнен файл applicationContext.xml (внедрены зависимости бинов, рисунок 4):

```xml
<bean id="alarmClock" class="ru.superzloyuser.Clocks.AlarmClock">
    <constructor-arg name="name" value="AlarmClock"/>
    <constructor-arg name="seconds" value="10000"/>
    <constructor-arg name="alarmCounter" value="62789"/>
</bean>

<bean id="clock" class="ru.superzloyuser.Clocks.Clock">
    <constructor-arg name="name" value="AlarmClock"/>
    <constructor-arg name="seconds" value="48020"/>
</bean>

<context:property-placeholder location="classpath:someProperties.properties"/>
<bean id="stopwatch" class="ru.superzloyuser.Clocks.Stopwatch">
    <constructor-arg name="name" value="AlarmClock"/>
    <constructor-arg name="seconds" value="${savedTime}"/>
</bean>


<bean id="user1" class="ru.superzloyuser.MechanismUser">
    <constructor-arg name="name" value="User1"/>
    <constructor-arg name="mechanism" ref="alarmClock"/>
</bean>

<bean id="user2" class="ru.superzloyuser.MechanismUser">
    <constructor-arg name="name" value="User2"/>
    <constructor-arg name="mechanism" ref="clock"/>
</bean>

<bean id="user3" class="ru.superzloyuser.MechanismUser">
    <constructor-arg name="name" value="User3"/>
    <constructor-arg name="mechanism" ref="stopwatch"/>
</bean>
```

Рисунок 4 – Внедрение зависимостей

4) Запуск Spring приложения (Рисунок 5):

```
Person: User1; Clock - Alarm Clock;; Time: Thu Dec 03 15:09:43 KRAT 2020; Alarm setted at 17:26:29
Person: User2; Clock - Wrist Watch; Time: Thu Dec 03 20:20:20 KRAT 2020
Person: User3; Clock - Stopwatch; Seconds right now: 43

Total defined beans count: 7
alarmClock
clock
org.springframework.context.support.PropertySourcesPlaceholderConfigurer#0
stopwatch
user1
user2
user3

Process finished with exit code 0
```

Рисунок 5 – Результат выполнения

4 Вывод

В ходе данной лабораторной работы был изучен механизм внедрения зависимостей в Spring.