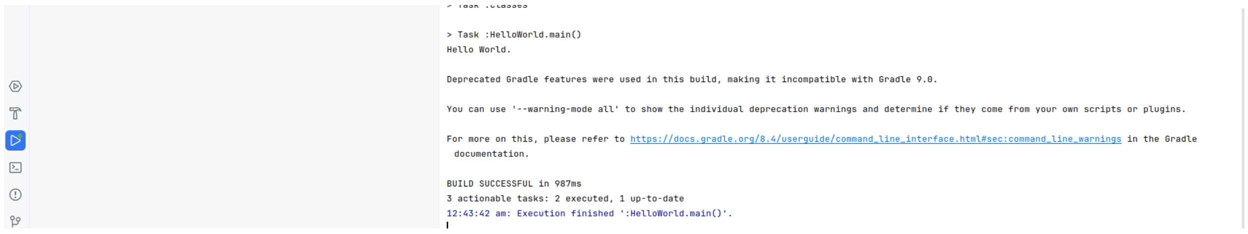
Name of student: Abhay Omprakash Prajapati		
Roll no: 41	**Tutorial No: 5**	
Title of LAB Assignment: Implementation of java application using spring framework.		
DOP: 25-09-2023	**DOS:02-10-2023**	
CO Mapped: Co1,Co2	**PO Mapped:** PO3 ,PO6	**Signature:**

1.Write a program to print "Hello World" using the Spring framework:

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class HelloWorldApp {
   public static void main(String[] args) {
       ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

       HelloWorld helloWorld = context.getBean("helloWorld",
HelloWorld.class);

       helloWorld.printMessage();
   }
}
```

//.applicationContext.xml
```xml
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

   <bean id="helloWorld" class="HelloWorld">
       <property name="message" value="Hello World" />
   </bean>
</beans>
```

Output:



```
> Task :HelloWorld.main()
Hello World.

Deprecated Gradle features were used in this build, making it incompatible with Gradle 9.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own scripts or plugins.

For more on this, please refer to https://docs.gradle.org/8.4/userguide/command_line_interface.html#sec:command_line_warnings in the Gradle
 documentation.

BUILD SUCCESSFUL in 987ms
3 actionable tasks: 2 executed, 1 up-to-date
12:43:42 am: Execution finished ':HelloWorld.main()'.
```

2. Create a Shape class and Rectangle class with constructor injection:

```java
//Shape.java

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

@Component
public class Shape {
   private String type;

   @Autowired
   public Shape(String type) {
       this.type = type;
   }

   public String getType() {
       return type;
   }
}


//Rectangle.java
@Component
public class Rectangle {
   private Shape shape;

   @Autowired
   public Rectangle(Shape shape) {
       this.shape = shape;
   }

   public void printDetails() {
       System.out.println("Rectangle has a " + shape.getType() + " shape.");
   }
}
```

3. Create an Address class and Student class with setter injection:

```java
//Address.java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class Address {
    private String city;
    private String state;

    public String getCity() {
        return city;
    }

    public String getState() {
        return state;
    }

    @Autowired
    public void setCity(String city) {
        this.city = city;
    }

    @Autowired
    public void setState(String state) {
        this.state = state;
    }
}
```

```java
//Student.java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
```

```java
@Component
public class Student {
    private Address address;

    public Address getAddress() {
        return address;
    }

    @Autowired
    public void setAddress(Address address) {
        this.address = address;
    }

    public void printAddress() {
        System.out.println("Student lives in " + address.getCity() + ", " +
address.getState());
    }
}
```

Conclusion:

we've used annotations like @Component for automatic bean detection and setter or constructor injection for the dependencies.