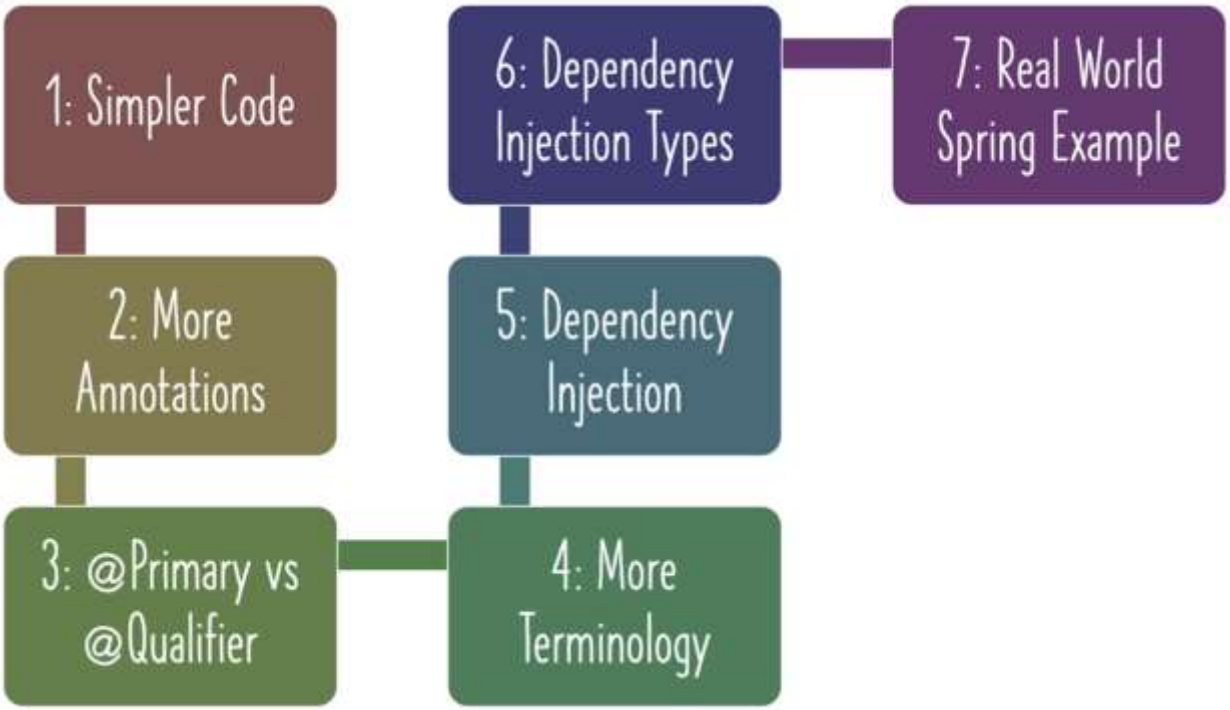
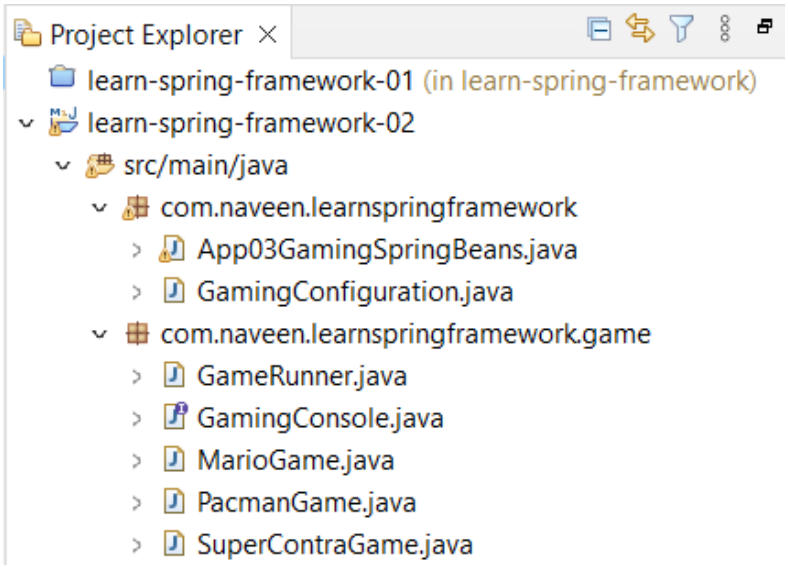
GETTING SPRING FRAMEWORK TO CREATE AND MANAGE YOUR JAVA OBJECTS

In this section we will learn,



In order to **comprehend** the distinctions between the **existence approach** and the **updatable approach**, it would be beneficial to **replicate the existing project** and subsequently **update the copied version.** This would **facilitate** the process of making comparisons between the two approaches.

PROJECT STRUCTURE:



Renamed our existence project as **learn-spring-framework-01.** And the copied project is named as **learn-spring-framework-02.**

Let’s get start with this question.

1. **Spring is managing objects and performing auto-wiring.**
    a. But aren't we writing the code to create objects?
    b. How do we get Spring to create objects for us?

What if we utilize the Spring framework to generate the beans for us, eliminating the need for manual bean creation?

SOLUTION

STEP 1:

1. Combine the configuration file with application file.
2. Copy the Configuration file and paste it in application file and delete the configuration file.
3. In same java file, there should not be have two public classes. So, remove public from configuration file.

App03GamingSpringBeans.java

```
package com.naveen.learnspringframework;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import com.naveen.learnspringframework.game.GameRunner;
import com.naveen.learnspringframework.game.GamingConsole;
import com.naveen.learnspringframework.game.PacmanGame;

@Configuration
class GamingConfiguration {

    @Bean
    public GamingConsole game() {
        var game = new PacmanGame();
        return game;
    }

    @Bean
    public GameRunner gameRunner(GamingConsole game) {
        var gameRunner = new GameRunner(game);
        return gameRunner;
    }
}

public class App03GamingSpringBeans {

    public static void main(String[] args) {
```

```
        try(var context =
                new AnnotationConfigApplicationContext
                    (GamingConfiguration.class)){
            context.getBean(GamingConsole.class).up();

            context.getBean(GameRunner.class).run();
        }
    }
}
```

OUTPUT:

```
12:04:27.119 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
12:04:27.121 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
12:04:27.123 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
12:04:27.135 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
12:04:27.140 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
12:04:27.146 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
12:04:27.155 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
up
Running game: com.naveen.learnspringframework.game.PacmanGame@4f49f6af
up
down
left
right
12:04:27.217 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicatic
```

STEP 2:

To make this even simpler, instead of creating separate class as a Configuration file, we can reuse the class (**App03GamingSpringBeans**) that we have.

1. Remove the beans from the Configuration class and put it in the Launcher class itself.
2. Remove the configuration file and make **App03GamingSpringBeans** as Configuration file.

App03GamingSpringBeans.java

```
package com.naveen.learnspringframework;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import com.naveen.learnspringframework.game.GameRunner;
import com.naveen.learnspringframework.game.GamingConsole;
import com.naveen.learnspringframework.game.PacmanGame;
```

```java
@Configuration
public class App03GamingSpringBeans {

    @Bean
    public GamingConsole game() {
        var game = new PacmanGame();
        return game;
    }

    @Bean
    public GameRunner gameRunner(GamingConsole game) {
        var gameRunner = new GameRunner(game);
        return gameRunner;
    }

    public static void main(String[] args) {

        try(var context =
                new AnnotationConfigApplicationContext
                    (App03GamingSpringBeans.class)){
            context.getBean(GamingConsole.class).up();

            context.getBean(GameRunner.class).run();
        }
    }
}
```

OUTPUT:

```
12:12:32.666 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
12:12:32.668 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
12:12:32.671 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
12:12:32.685 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
12:12:32.695 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
12:12:32.704 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
12:12:32.716 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
up
Running game: com.naveen.learnspringframework.game.PacmanGame@21be3395
up
down
left
right
12:12:32.814 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicatic
```

STEP 3:

1. To make it even more simpler.
2. We are manually creating beans using @Bean annotation.
3. We can ask spring to create the beans for us.
    a. Add **@Component** annotation to a pacmanGame class.
    b. Remove the code that we have manually created bean.
    c. And tell the spring framework to search the component in a particular package using @ComponentScan(package)

For example, we can ask spring to create a bean for PacmanGame for us. Instead of writing this code,

```
@Bean
    public GamingConsole game() {
        var game = new PacmanGame();
        return game;
}
```

How can we ask spring to create pacmanGame of us?

In the Java Spring Framework, "**@Component**" is an annotation used to denote that a **particular class is a component or a bean** that should be **managed by the Spring container**. By adding this annotation to a class, **Spring framework automatically creates an instance of that class and manages its lifecycle**. This allows for easier dependency injection and inversion of control, as well as facilitates the development of modular and reusable code.

"**@ComponentScan**" is an annotation in the java Spring framework that enables automatic discovery and registration of Spring-managed components or beans, by scanning the specified base packages for classes annotated with "**@Component**".

When "**@ComponentScan**" is added to a **Spring configuration class**, it **instructs** the Spring container to **scan the specified base packages** for **components or beans** and **register** them in the **container**, thereby making them **available for dependency injection** and other **Spring-managed services**.

GameRunner.java

```
package com.naveen.learnspringframework.game;
import org.springframework.stereotype.Component;

@Component
public class GameRunner {
    private GamingConsole game;

    public GameRunner(GamingConsole game) {
        this.game = game;
    }

    public void run() {
        System.out.println("Running game: " + game);
```

```
        game.up();
        game.down();
        game.left();
        game.right();
    }
}
```

PacmanGame.java

```
package com.naveen.learnspringframework.game;
import org.springframework.stereotype.Component;

@Component
public class PacmanGame implements GamingConsole {
    public void up() {
        System.out.println("up");
    }

    public void down() {
        System.out.println("down");
    }

    public void left() {
        System.out.println("left");
    }

    public void right() {
        System.out.println("right");
    }
}
```

GamingAppLauncherApplication.java (renamed)

```
package com.naveen.learnspringframework;

import
org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import com.naveen.learnspringframework.game.GameRunner;
import com.naveen.learnspringframework.game.GamingConsole;

@Configuration
@ComponentScan("com.naveen.learnspringframework.game")
public class GamingAppLauncherApplication {

    public static void main(String[] args) {

        try(var context =
                new AnnotationConfigApplicationContext
```

```
                    (GamingAppLauncherApplication.class)){
            context.getBean(GamingConsole.class).up();

            context.getBean(GameRunner.class).run();
        }
    }
}
```

1. In the "**main()**" method of the "**GamingAppLauncherApplication**" class, an instance of "**AnnotationConfigApplicationContext**" is created, which is a Spring container that loads the Spring configuration defined in the "**GamingAppLauncherApplication**" class.
2. The "**context.getBean(GamingConsole.class).up()**" statement retrieves the "**GamingConsole**" bean from the Spring container and calls its "**up()**" method, which prints the message "**up**".
3. The "**context.getBean(GameRunner.class).run()**" statement retrieves the "**GameRunner**" bean from the Spring container and calls its "**run()**" method. This in turn retrieves the "**GamingConsole**" bean from the Spring container, passes it to the "**GameRunner**" constructor, and then calls the "**run**()" method of the "**GameRunner**" instance. The "**run**()" method prints the message "**Running game:**" followed by the instance of the game being run, and then calls the "up()", "down()", "left()", and "right()" methods of the "**GamingConsole**" instance, which in this case is an instance of "**PacmanGame**".

OUTPUT:

```
13:57:24.897 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:57:24.900 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:57:24.902 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:57:24.918 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:57:24.929 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:57:24.947 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
13:57:24.957 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFact
up
Running game: com.naveen.learnspringframework.game.PacmanGame@81d9a72
up
down
left
right
13:57:25.051 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicatic
```