EXPLORING PRIMARY AND QUALIFIER ANNOTATIONS FOR SPRING COMPONENTS

What would happen if we put @Component at multiple games? Which game should be chosen by spring for GameRunner class?

MarioGame.java

```
package com.naveen.learnspringframework.game;

import org.springframework.stereotype.Component;

@Component
public class MarioGame implements GamingConsole {

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

    public void down() {
        System.out.println("Go into a hole");
    }

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

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

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");
    }
}
```

OUTPUT:

```
org.springframework.beans.factory.NoUniqueBeanDefinitionException: No qualif
org.springframework.beans.factory.config.DependencyDescriptor.resolveNotUniq
org.springframework.beans.factory.support.DefaultListableBeanFactory.doResol
org.springframework.beans.factory.support.DefaultListableBeanFactory.resolve
org.springframework.beans.factory.support.ConstructorResolver.resolveAutowir
org.springframework.beans.factory.support.ConstructorResolver.createArgument
 14 more
```

We would get **NoUniqueBeanDefinitionException** exception. For the GameRunner class we need the implementation of GamingConsole. Spring interface will look for the implementation of GamingConsole interface. But now we have two implementations. That is PacmanGame and MarioGame. So, the spring unable to choose the implementations among these.

SOLUTIONS:

1. **Use @Primary annotation:** Indicates that a bean should be given preference when multiple candidates are qualified to autowire a single-valued dependency.
2. **Use @Qualifier annotation:** Before the class definition, we can specify the qualifier name inside the parenthesis within double quotes. We can directly specify the bean which should be autowired in the constructor parameter using qualifier name.

SuperContraGame.java

```
package com.naveen.learnspringframework.game;

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

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

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

    public void left() {
```

```
        System.out.println("Go back");
    }

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

GameRunner.java

```
package com.naveen.learnspringframework.game;

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

@Component
public class GameRunner {
    private GamingConsole game;

    public GameRunner(@Qualifier("SuperContraGameQualifier")GamingConsole game)
{
        this.game = game;
    }

    public void run() {
        System.out.println("Running game: " + game);
        game.up();
        game.down();
        game.left();
        game.right();
    }
}
```