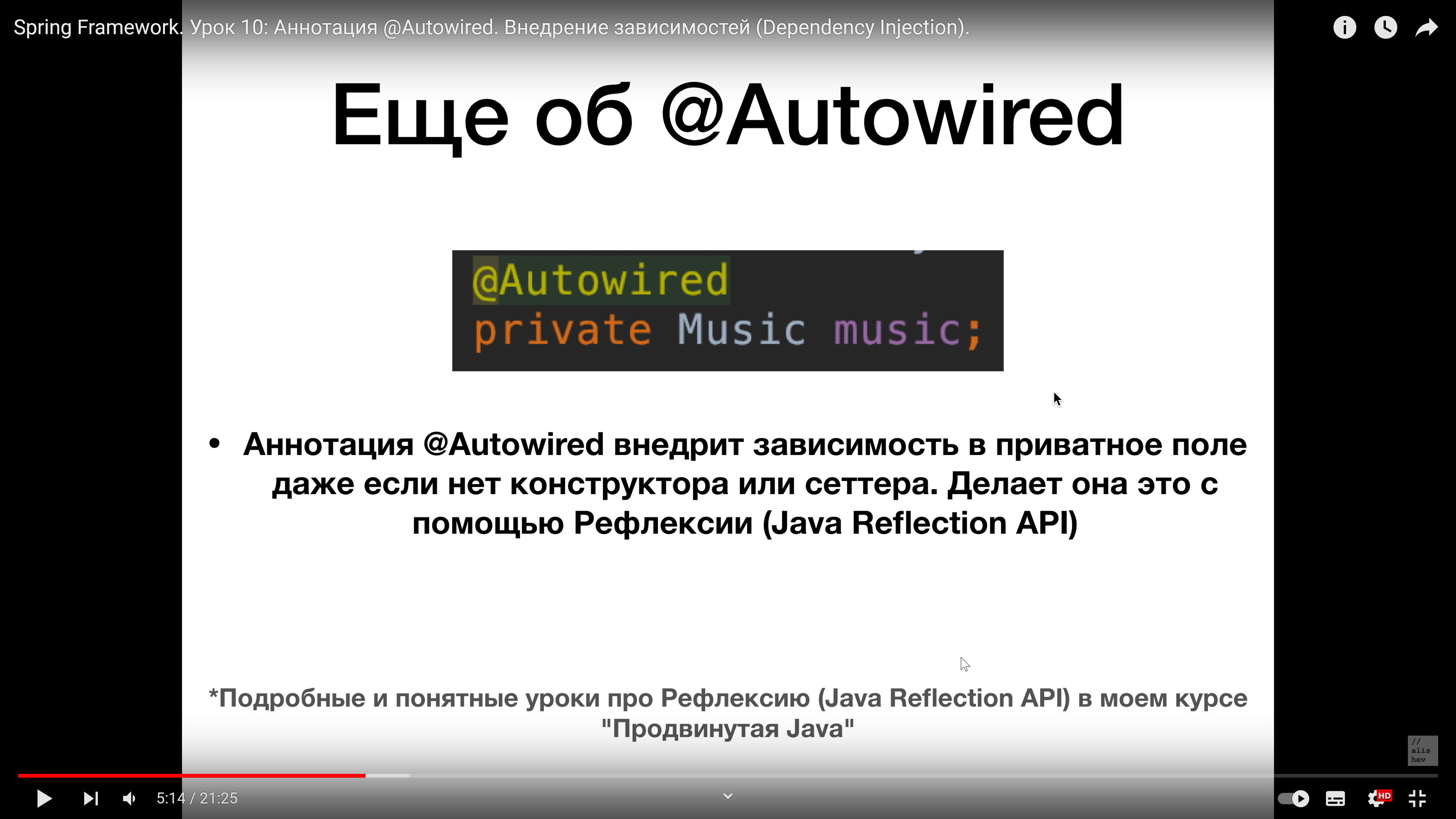
## Teil 1

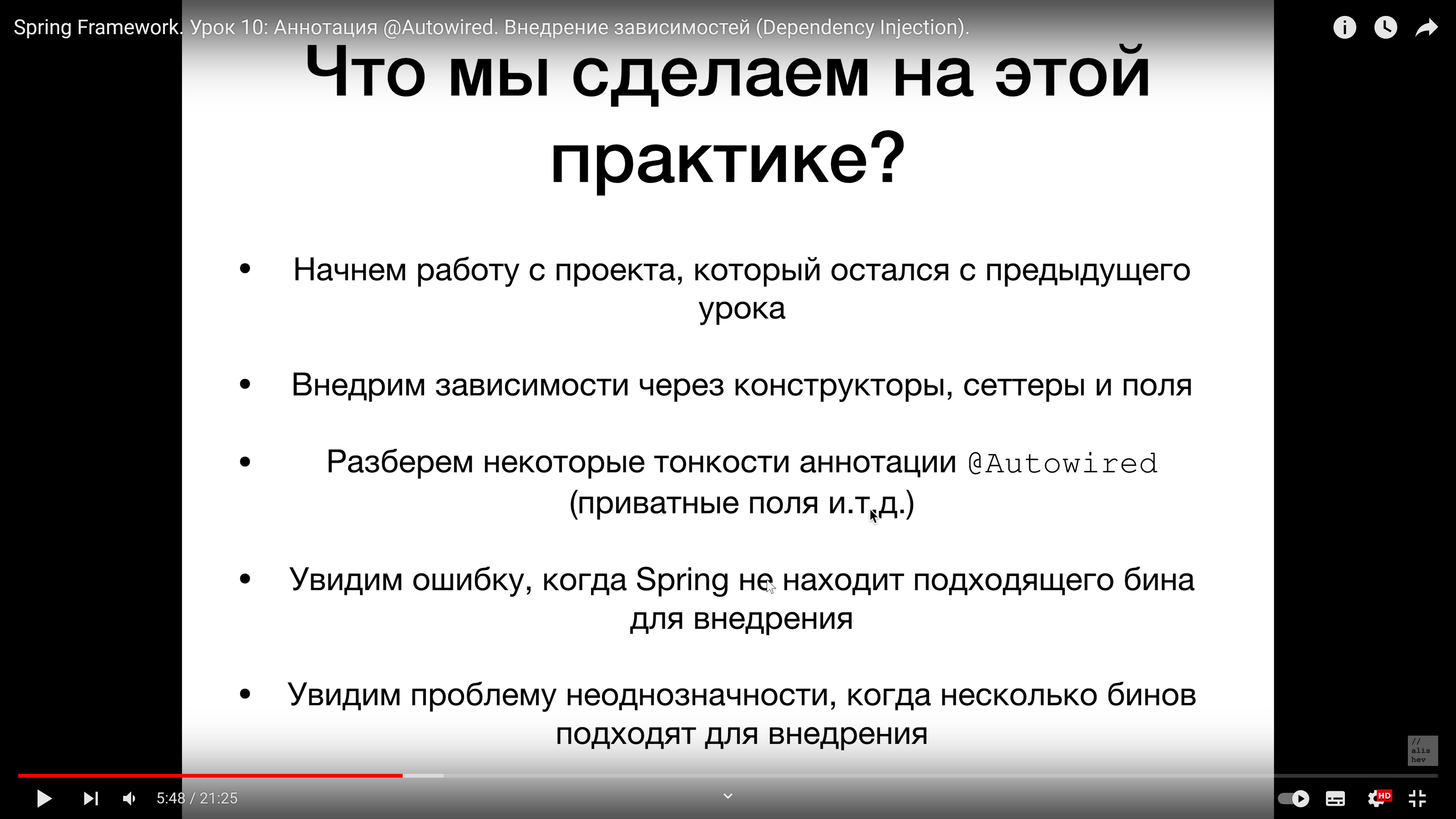
Verwirklichung der Dependency Injection mit Hilfe der Annotation

In dem vorherigen Unterricht erstellten wir das Objekt MusicPlayer manuel

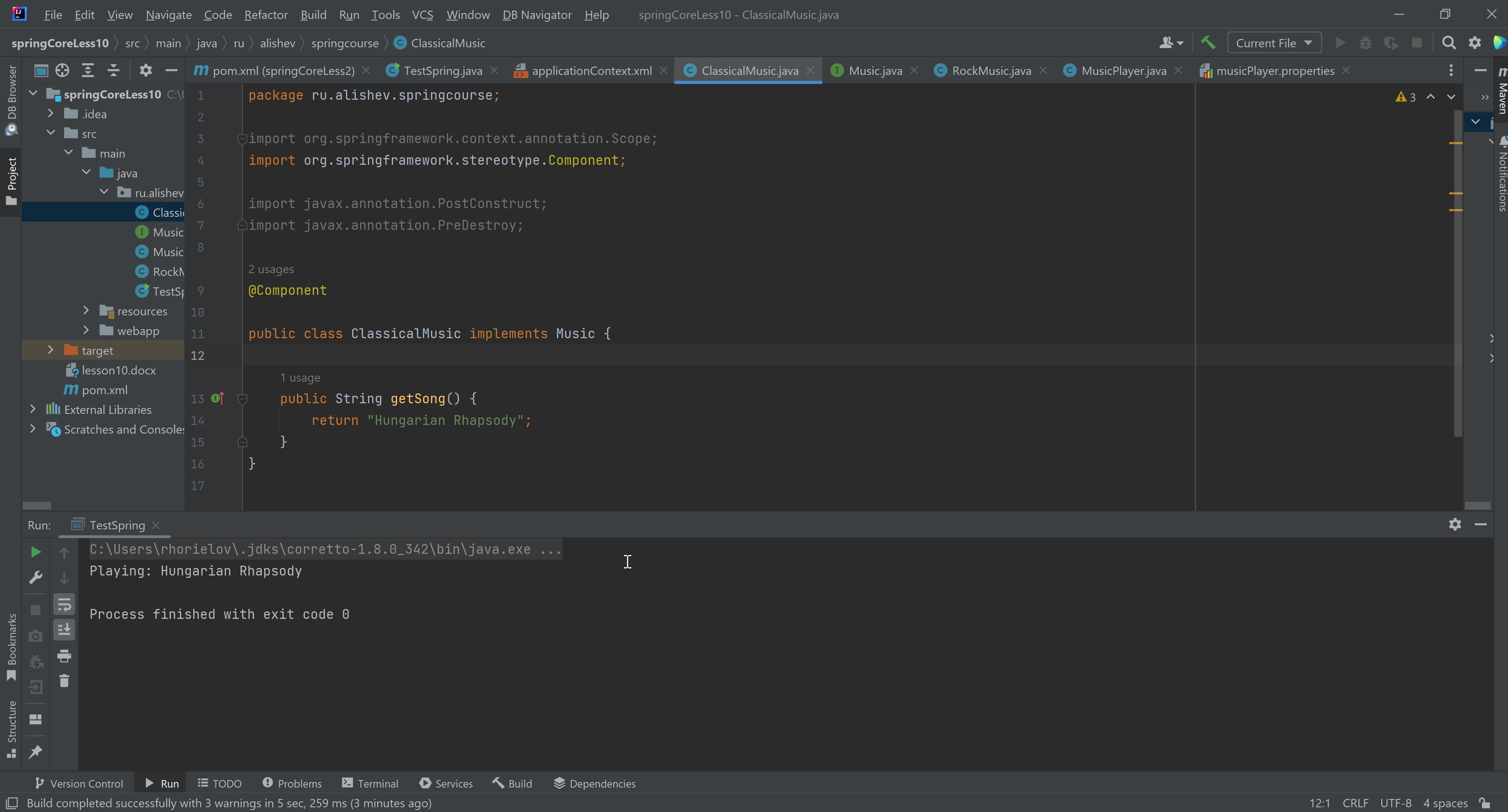
MusicPlayer musicPlayer = new MusicPlayer(music);

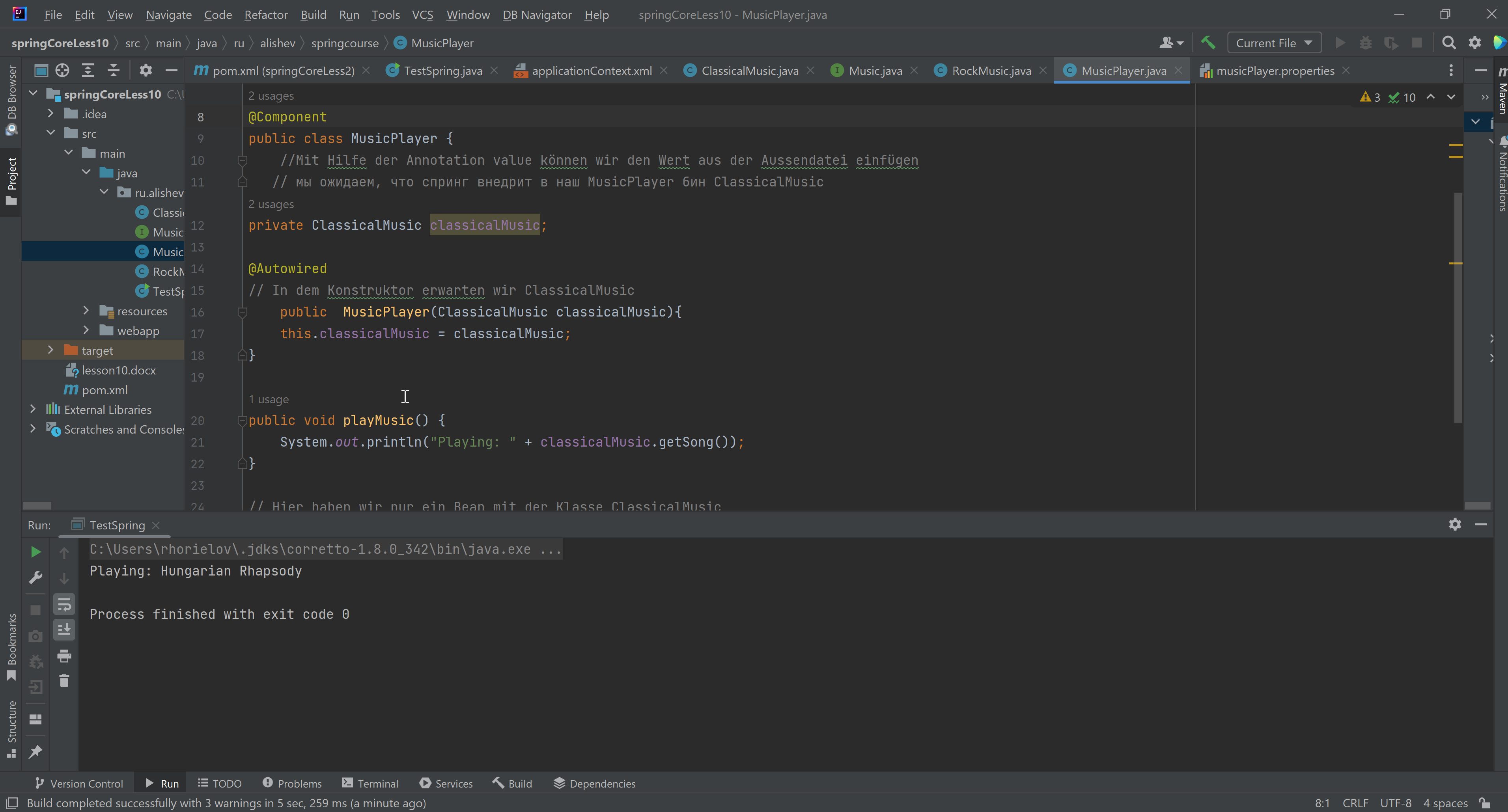
In diesem Unterricht bekommen wir es mit Hilfe von Spring und es wird von dem Context erhalten.





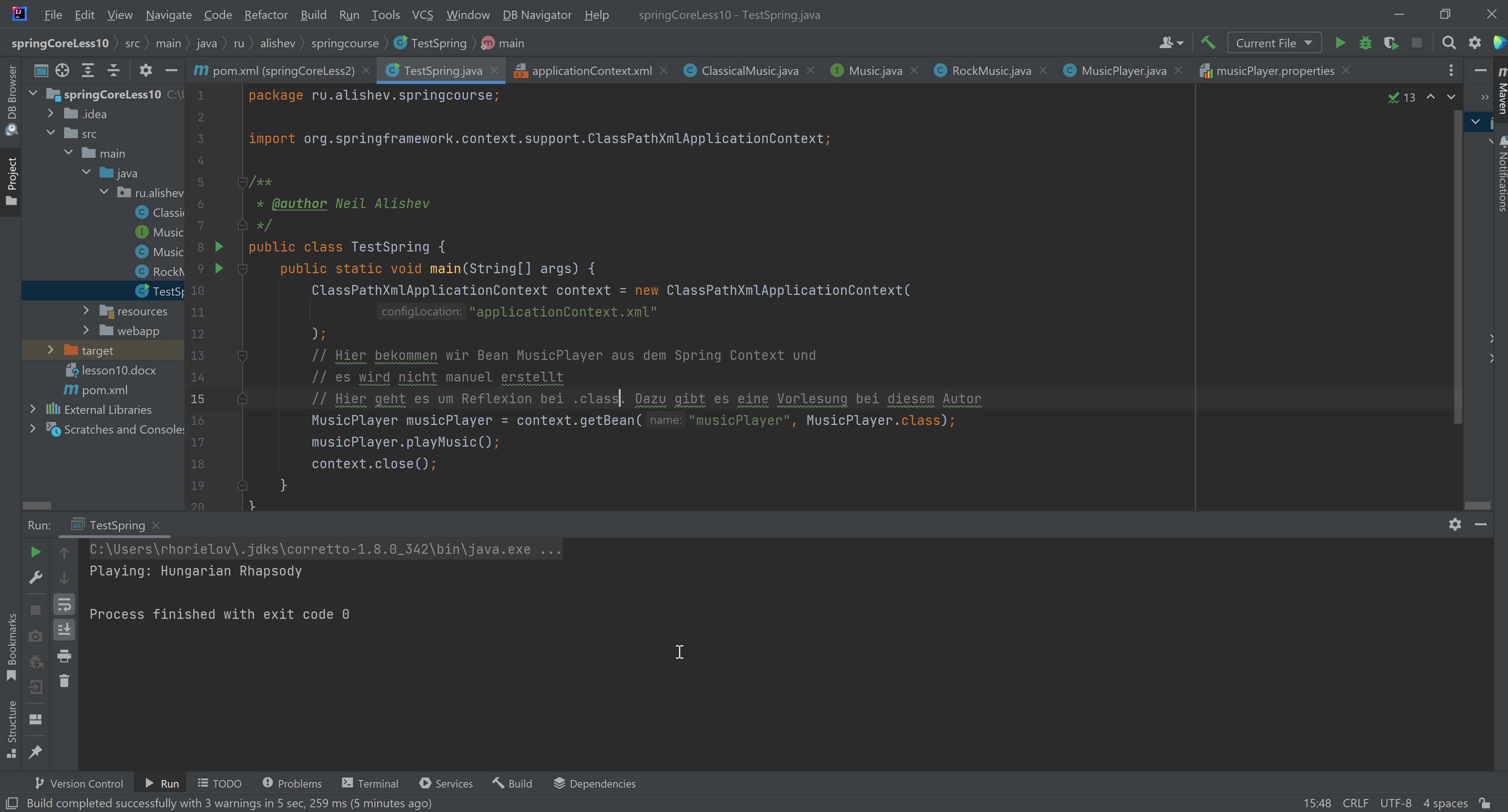
Die Component ClassicalMusic





Wurde automatisch in unser MusicPlayer eingefügt

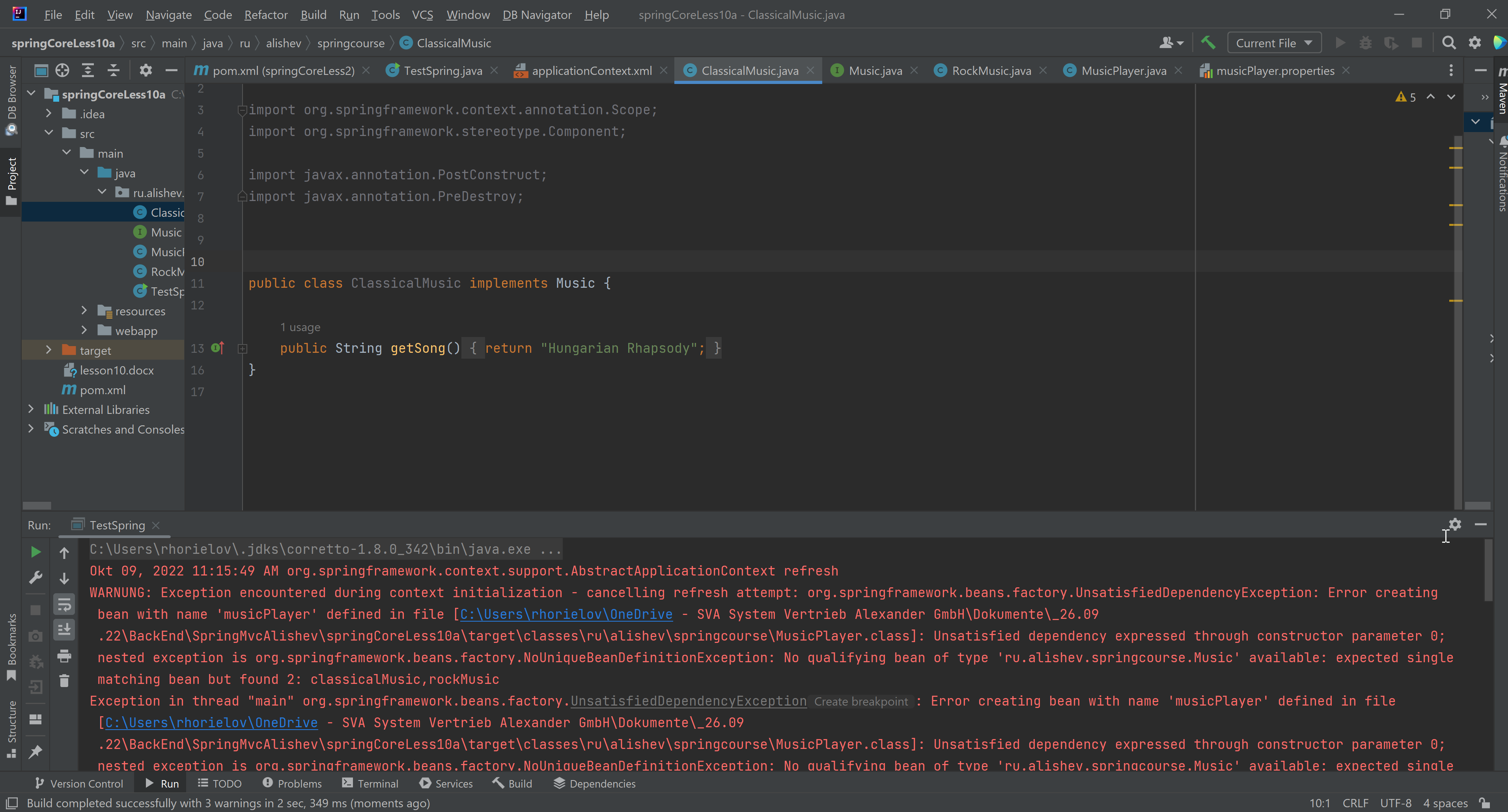
@Autowired



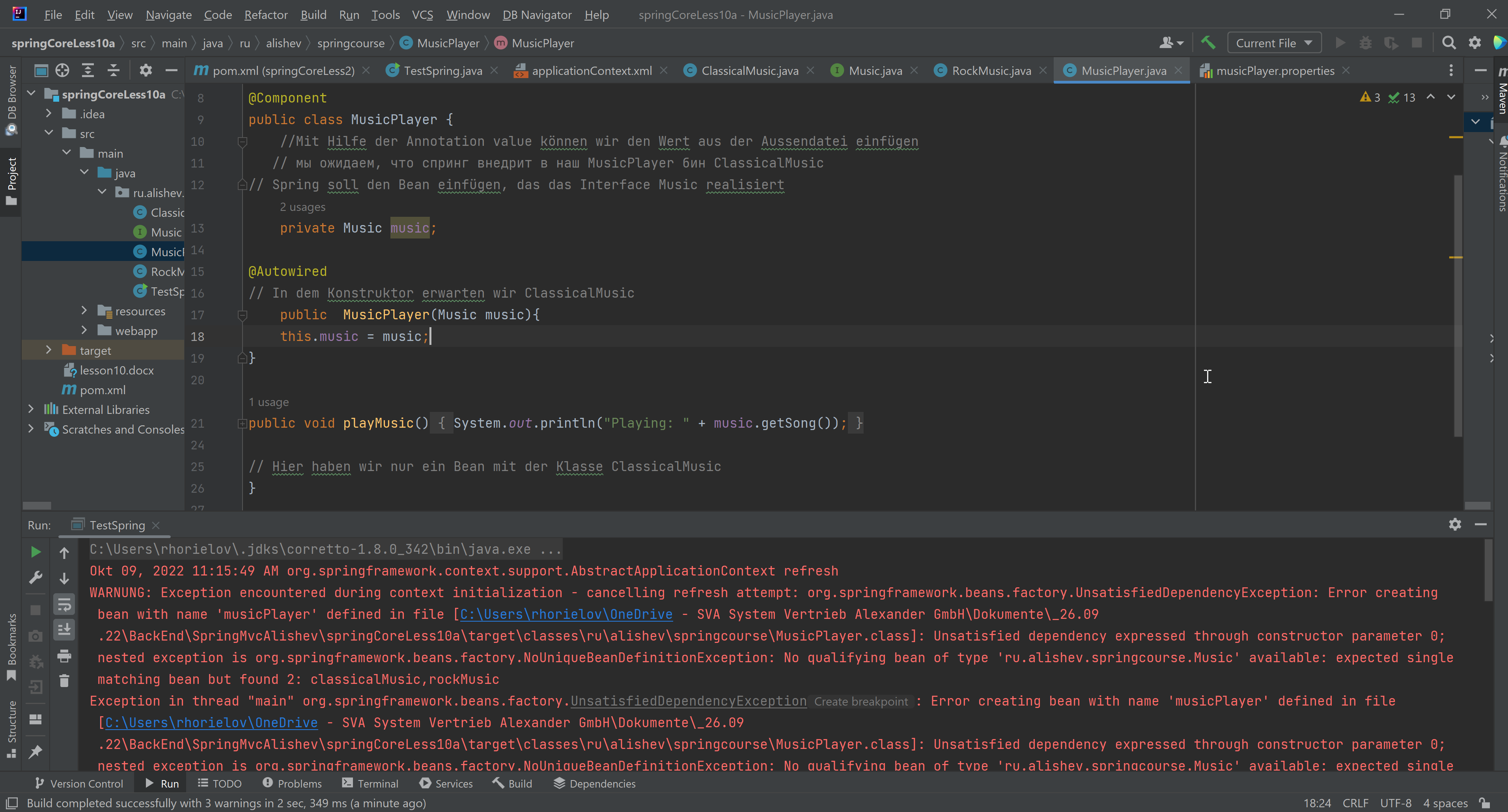
## Teil 2

Was passiert, wenn Spring den nötigen Bean nicht findet, um in den MusicPlayer einzufügen

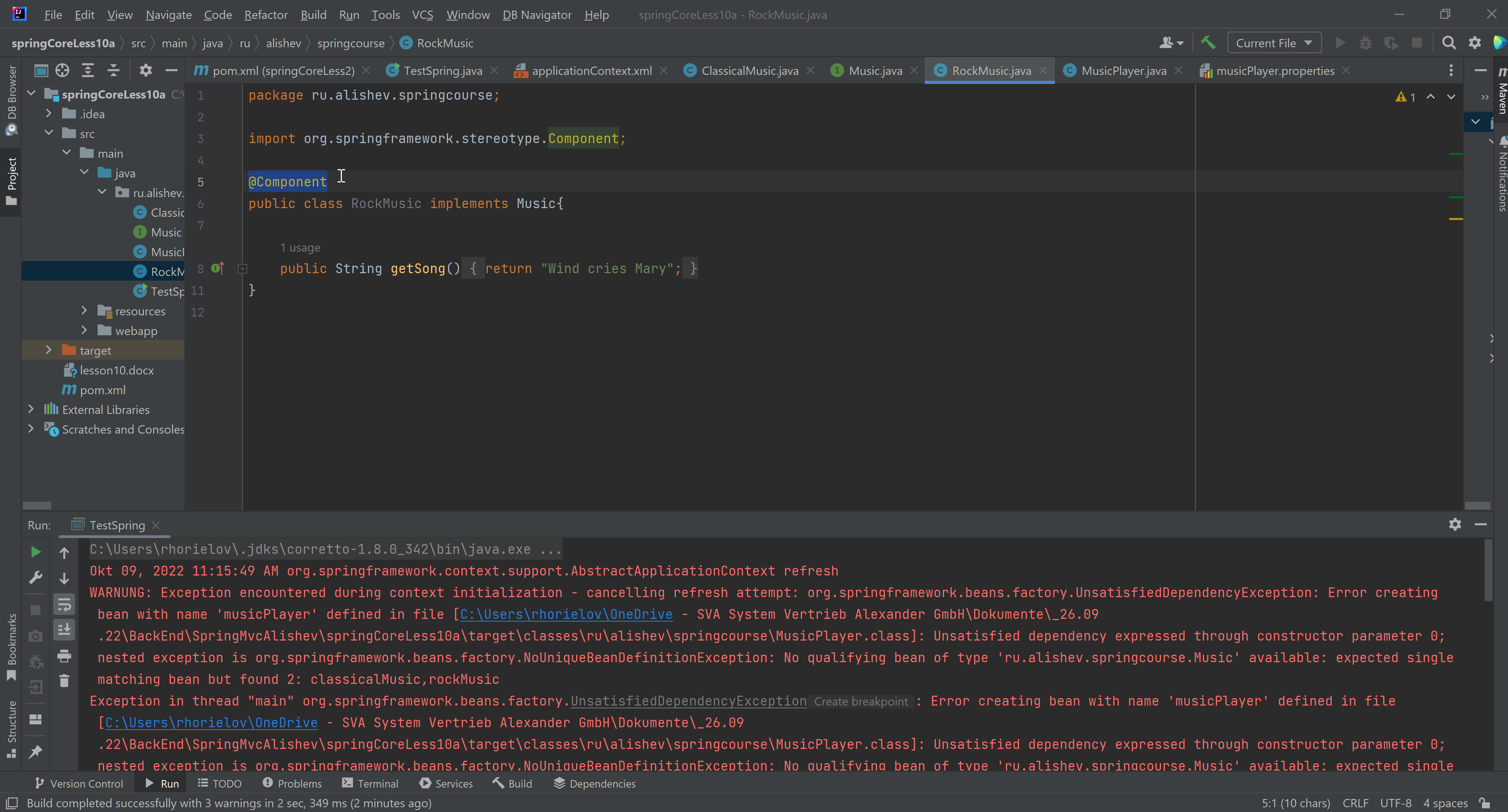
In der Datei wurde @Component entfernt

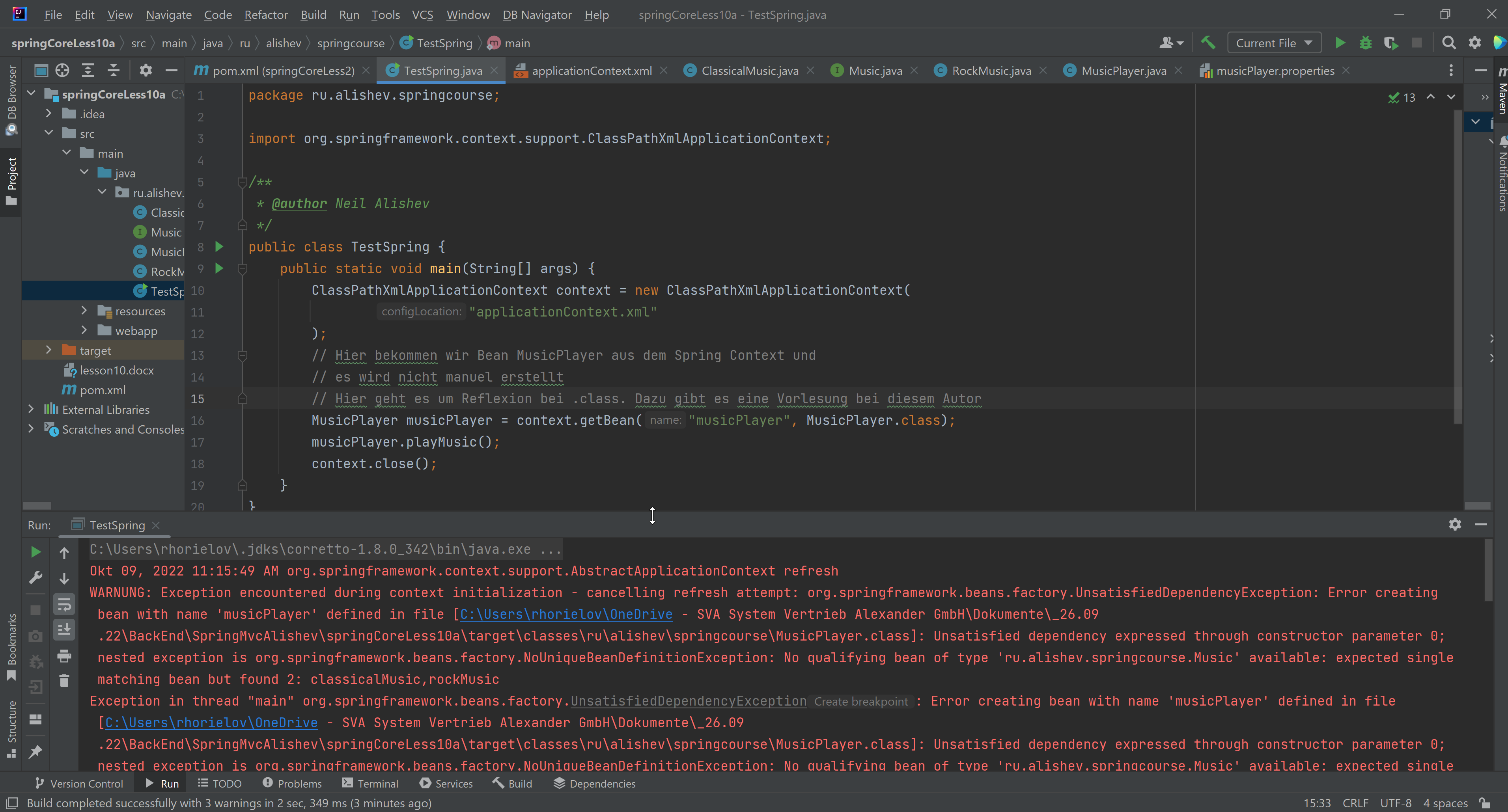


In den Konstruktor MusicPlayer ist ein Interface hinzugefügt



Nur in der Klasse RockMusic bleint @Component

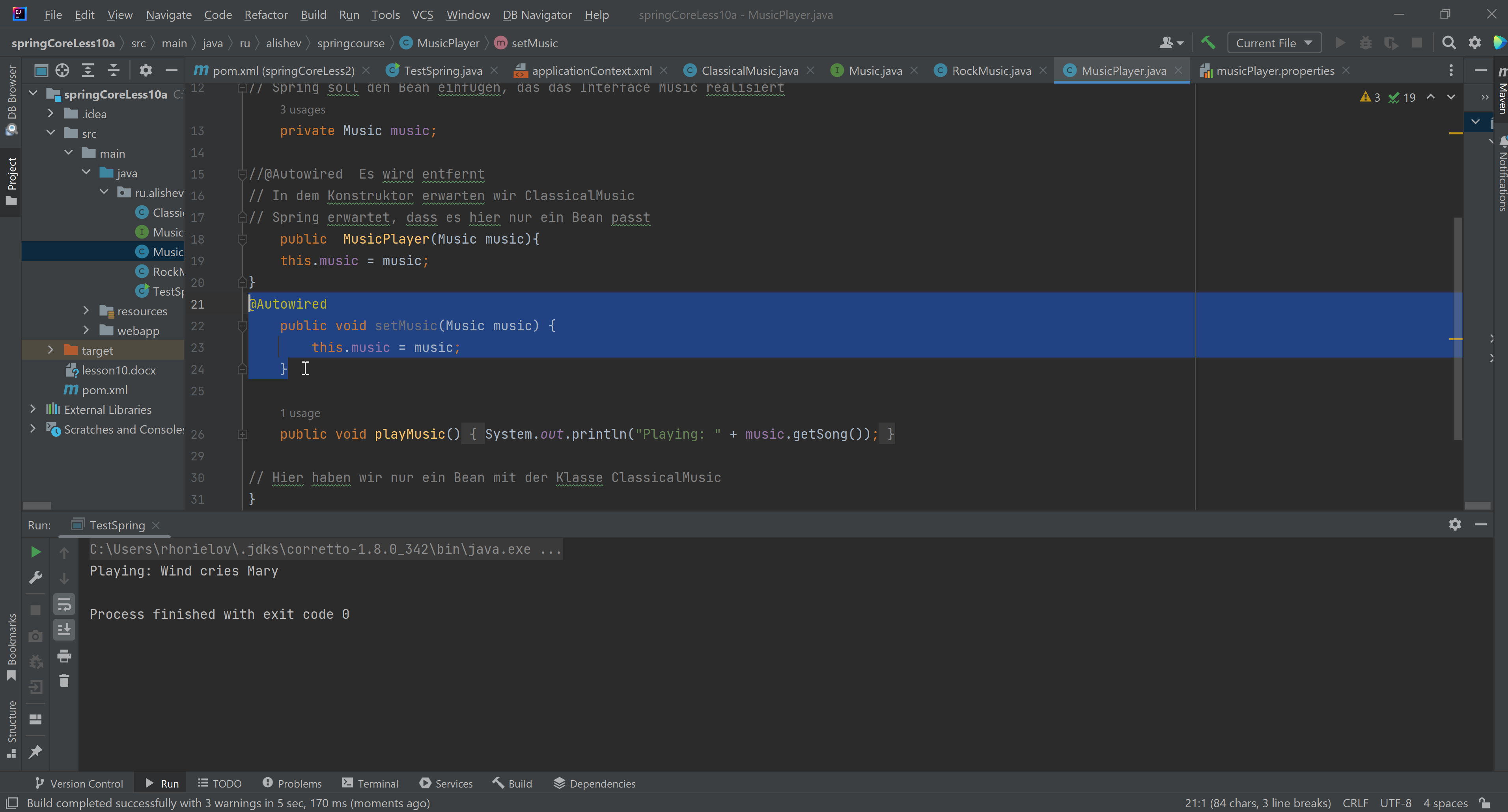




## Teil 3

Hier wird die Dependency mit Setter hinzugefügt

In der Datei wurde nur das Unterstrichene hinzugefügt



Für den Sprint ist nur wichtig die Annotaion Autowired

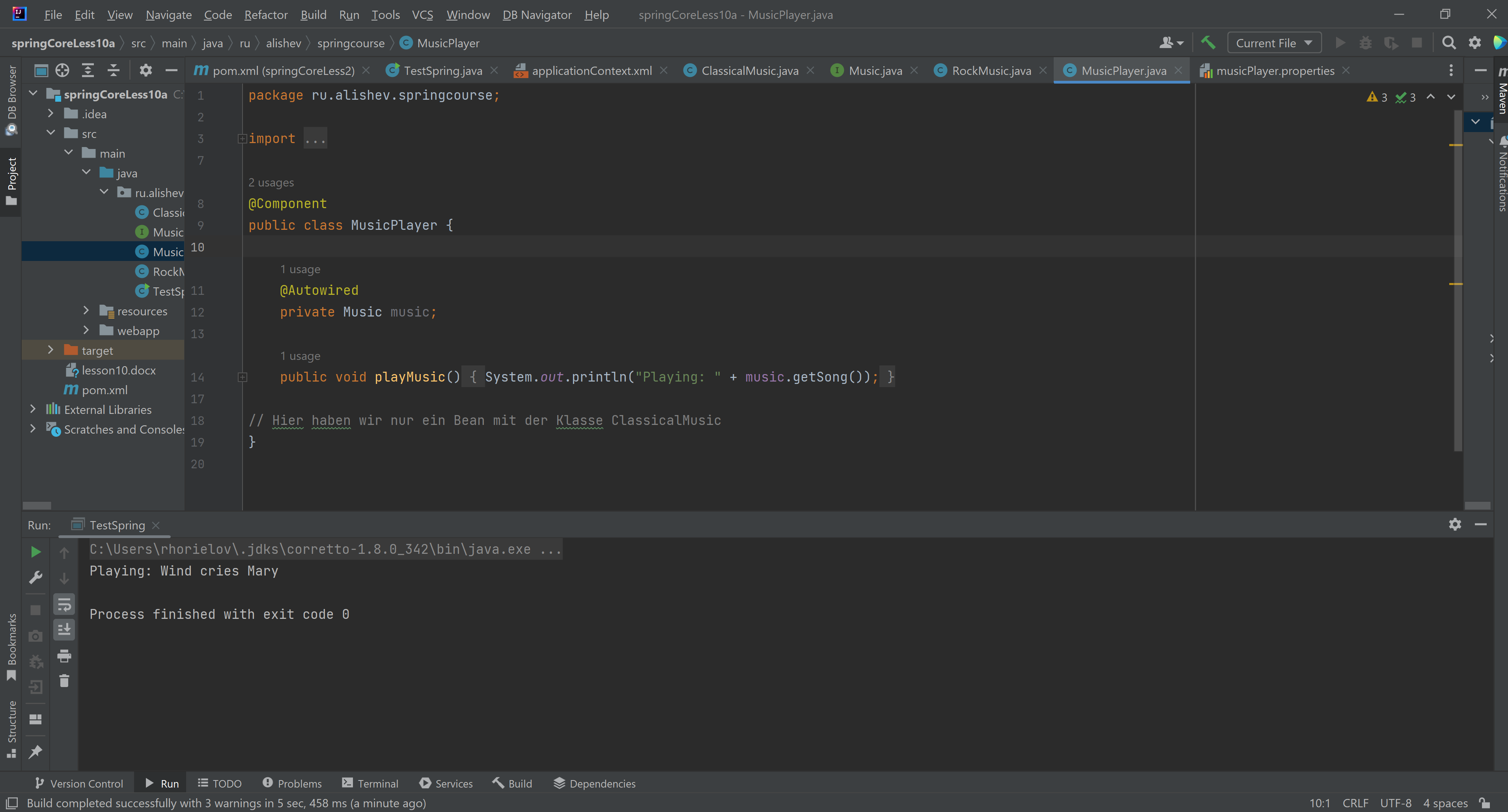
## Teil 4

Einfügen der Dependency mit dem Feld

Das brauchen wir jetzt nicht mehr

// public MusicPlayer(Music music){  
// this.music = music;  
//}  
//@Autowired  
// public void setMusic(Music music) {  
// this.music = music;  
// }

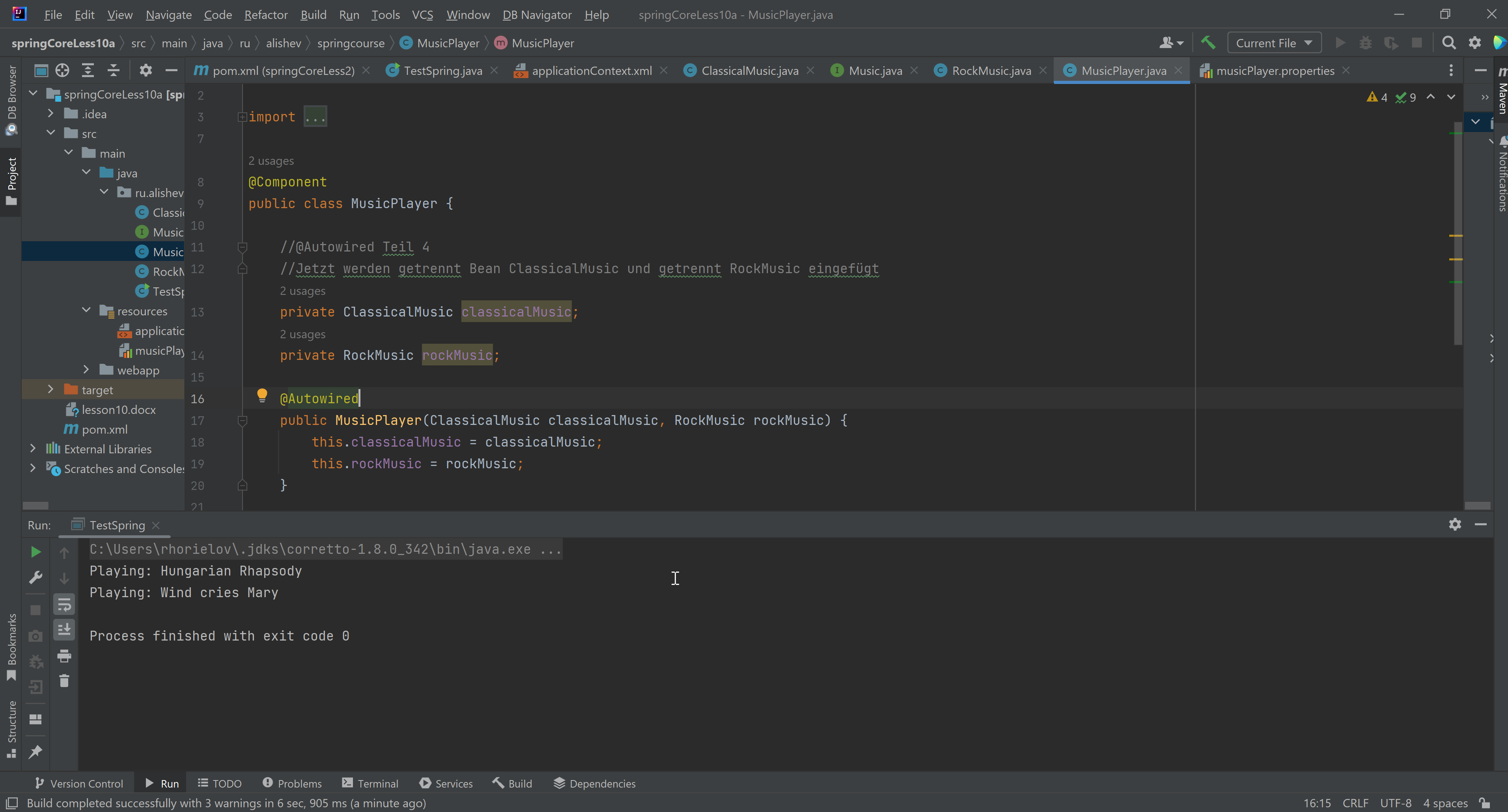
So wird Dependency für den Field gebraucht



## Teil 5

Im Konstruktor werden zwei Dependency hinzugefügt

@Autowired  
public MusicPlayer(ClassicalMusic classicalMusic, RockMusic rockMusic) {  
 this.classicalMusic = classicalMusic;  
 this.rockMusic = rockMusic;  
}



## Teil 6