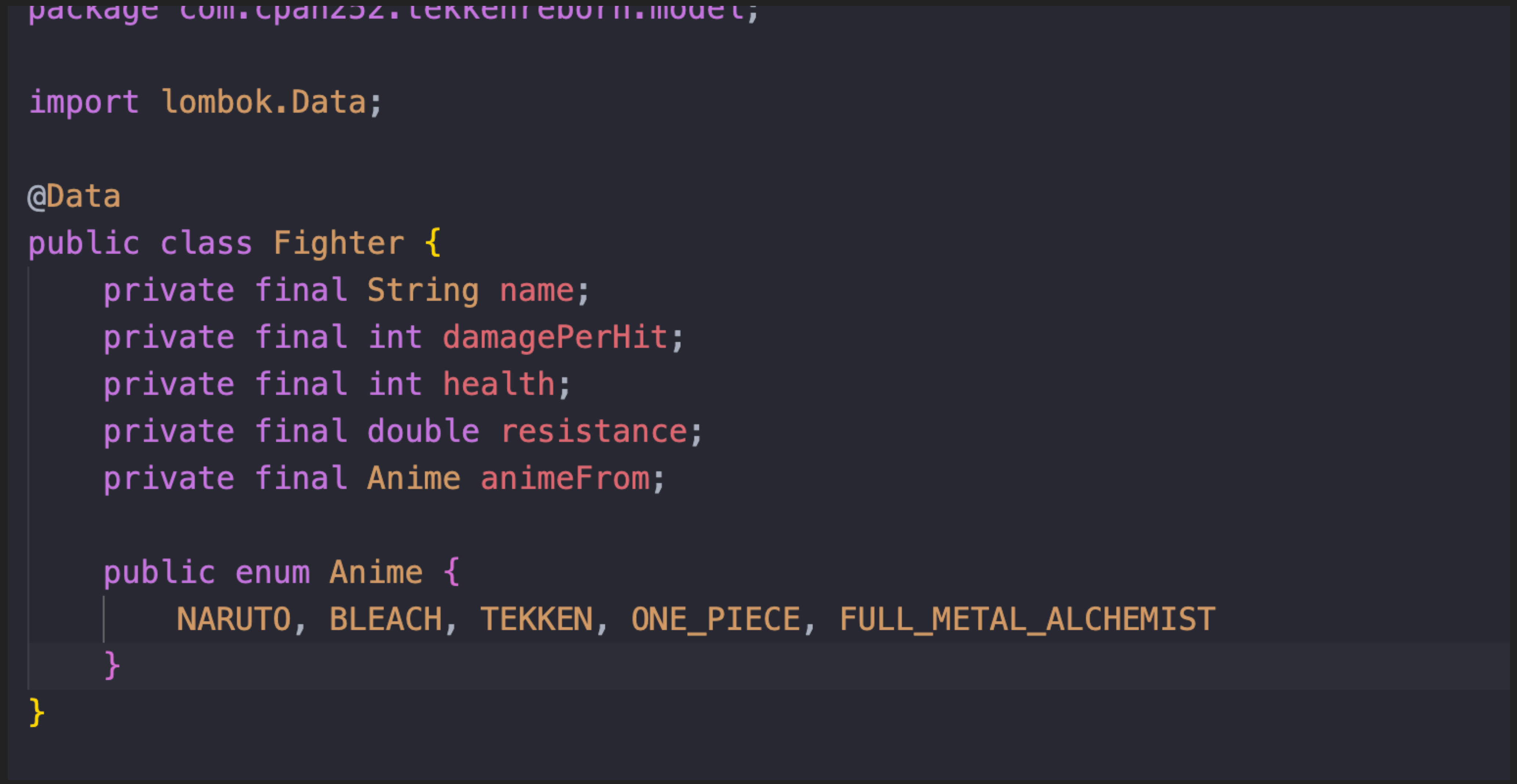
CPAN-252 WEB APPLICATION DEVELOPMENT FOR CPAN DIPLOMA. PROF. ANTON KOVUNOV

DIVING DEEPER INTO SPRING WEB, PRESENTING MODEL DATA, PROCESSING INPUT

DOMAIN MODEL FIGHTER

- Now we will create domain model of Fighter class to be able to create characters
- We will use Lombok library to reduce some boilerplate code, like getters, setters, toString, equals, Hashcode and provides arg constructor
- 

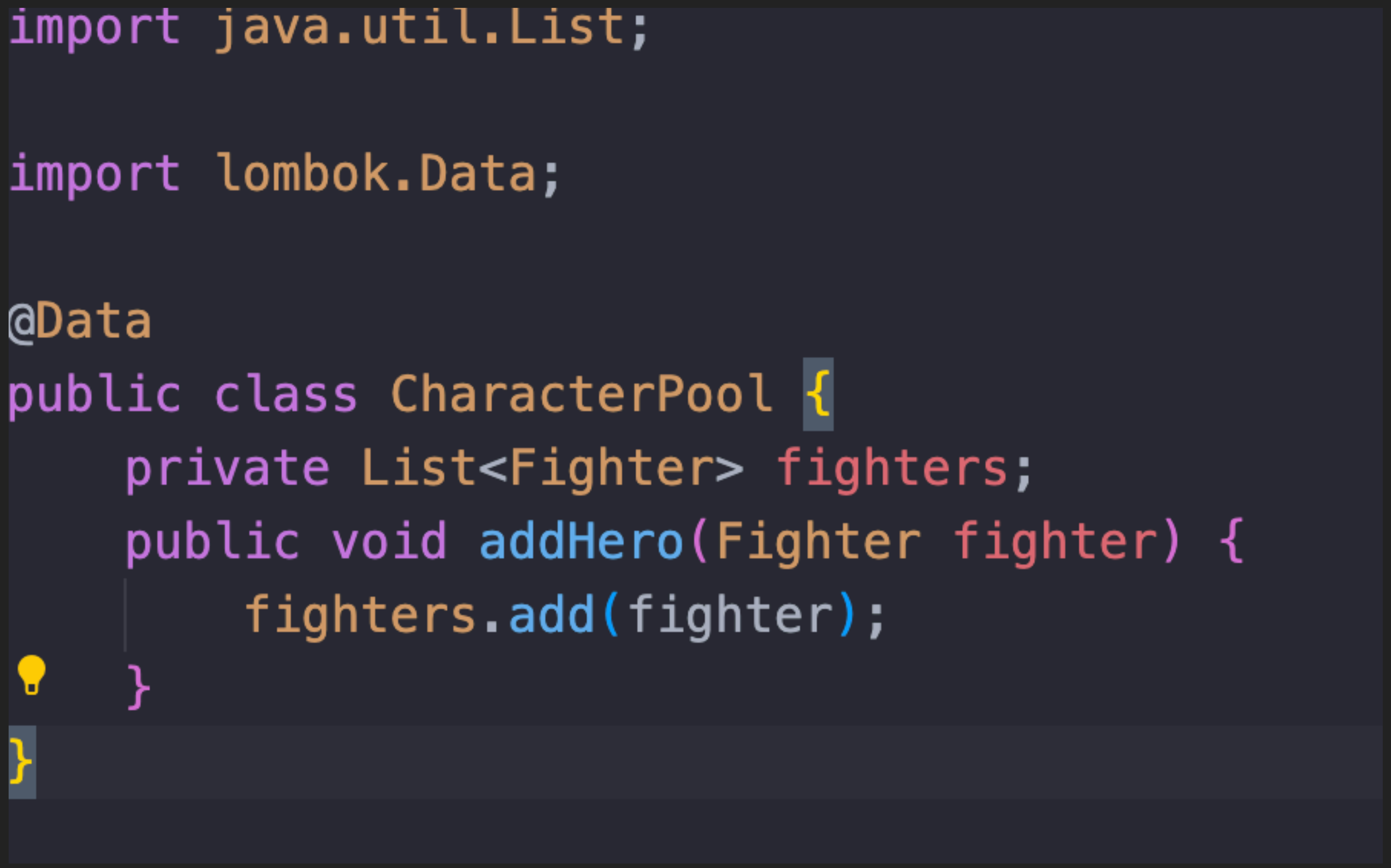
```java
package com.cpan252.tekkenreborn.model;

import lombok.Data;

@Data
public class Fighter {
    private final String name;
    private final int damagePerHit;
    private final int health;
    private final double resistance;
    private final Anime animeFrom;

    public enum Anime {
        NARUTO, BLEACH, TEKKEN, ONE_PIECE, FULL_METAL_ALCHEMIST
    }
}
```

LOMBOK PROJECT

- We can also find it in maven code of pom.xml file, we need to exclude it from build, because essentially it's not used in runtime, but acts more as syntactic sugar.
- If you want to see more features of it, you can visit this website https://projectlombok.org/

CREATING CHARACTER POOL

- We will create character pool which will act as a list of characters, it will be a simple class that contains collection of fighters



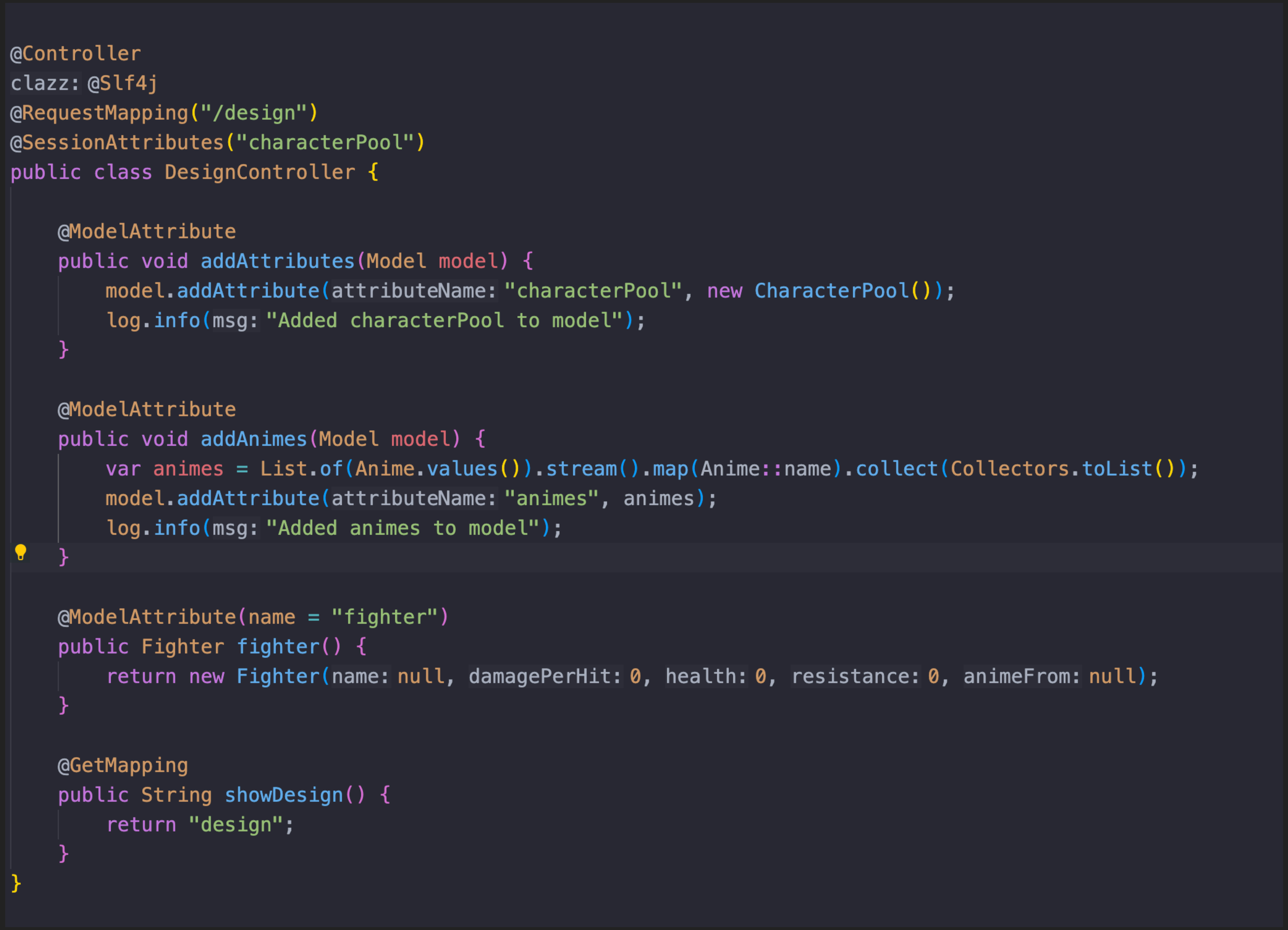
```java
import java.util.List;

import lombok.Data;

@Data
public class CharacterPool {
    private List<Fighter> fighters;
    public void addHero(Fighter fighter) {
        fighters.add(fighter);
    }
}
```

-

FIGHTER DESIGN CONTROLLER

- Annotate class as controller
- Create some model attributes to work with them when designing your hero
- Define GET method to return design page
- Define submit button

DESIGN CONTROLLER



```
@Controller
clazz: @Slf4j
@RequestMapping("/design")
@SessionAttributes("characterPool")
public class DesignController {

    @ModelAttribute
    public void addAttributes(Model model) {
        model.addAttribute(attributeName: "characterPool", new CharacterPool());
        log.info(msg: "Added characterPool to model");
    }

    @ModelAttribute
    public void addAnimes(Model model) {
        var animes = List.of(Anime.values()).stream().map(Anime::name).collect(Collectors.toList());
        model.addAttribute(attributeName: "animes", animes);
        log.info(msg: "Added animes to model");
    }

    @ModelAttribute(name = "fighter")
    public Fighter fighter() {
        return new Fighter(name: null, damagePerHit: 0, health: 0, resistance: 0, animeFrom: null);
    }

    @GetMapping
    public String showDesign() {
        return "design";
    }
}
```

DESIGN TEMPLATE

- Overall we need to define the hero form to create the new fighter, by using model anime attributes specified and binding each form input to character attribute

- Define some basic styling to make app look a little bit aligned
- Template is too big to paste it, so let’s open IDE and take a look there

QUESTIONS?

On the lab we going to add some validations to our design page and add post method to add character to hero pool.