

A grayscale photograph of a person with dark hair wearing large headphones, sitting at a desk and looking intently at a laptop screen. Their hands are clasped together near their chin. The background is blurred, showing what appears to be a modern office or studio environment with other people and equipment. A white rectangular border is superimposed over the image, framing the text.

# CLASSES

Java Clean code

# Outline

## **Lesson 1.**

Clean code

## **Lesson 2.**

Names

## **Lesson 3.**

Methods

## **Lesson 4.**

Classes

## **Lesson 5.**

Comments

*Any fool can write code that  
a computer can understand.  
Good programmers write code that  
humans can understand.*

**Martin Fowler**



# What is a class

## **Class is like a container**

- Class should do one thing only.
- Methods are important
- Use generalization for abstract classes and interfaces
- Opposite to data structures



# Class naming

## Camel Case

- Nouns
- Simple
- Use whole word for the name







# Classes name length

- Private and inner classes: long and descriptive names.
- Public classes should have small names

# When to create a class

- Modeling objects
- Code reusability
- Single Responsibility



# Cohesion



Strong responsibility



Reusable and easier to debug



Dead methods





# Bad Example

## Bus

- Edit bus
- Update ticket price
- Schedule maintenance
- Monthly payroll drivers
- Select payroll types



# Good Example

## **Bus**

- Edit bus
- Update ticket price

## **BusRepairment**

- Schedule maintenance

## **DriverPayroll**

- Moonthly payroll drivers
- Select payroll types

\*PasswordFormCommand.java

```
1 package com.kirilanastasov.reservationservices.controller.command;
2
3 import lombok.Data;
4 import lombok.experimental.Accessors;
5
6 import javax.validation.constraints.NotBlank;
7 import javax.validation.constraints.Size;
8
9 @Data
10 @Accessors(chain = true)
11 public class PasswordFormCommand {
12     @NotBlank
13     @Size(min = 5, max = 12)
14     private String password;
15
16     private String email;
17 }
18
```

UserSignupRequest.java

```
1 package com.kirilanastsov.reservationservices.controller.request;
2
3+ import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
10
11 @Getter
12 @Setter
13 @Accessors(chain = true)
14 @NoArgsConstructor
15 @JsonIgnoreProperties(ignoreUnknown = true)
16 public class UserSignupRequest {
17-     @NotEmpty(message = "{constraints.NotEmpty.message}")
18     private String email;
19
20-     @NotEmpty(message = "{constraints.NotEmpty.message}")
21     private String password;
22
23-     @NotEmpty(message = "{constraints.NotEmpty.message}")
24     private String firstName;
25
26-     @NotEmpty(message = "{constraints.NotEmpty.message}")
27     private String lastName;
28
29     private String mobileNumber;
30 }
31
```



DashboardController.java

```
1 package com.kirilanastasov.reservationservices.controller.ui;
2
3 import com.kirilanastasov.reservationservices.controller.command.*;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26 @Controller
27 public class DashboardController {
28
29     @Autowired
30     private UserService userService;
31
32     @Autowired
33     private BusReservationService busReservationService;
34
35     @GetMapping(value = "/dashboard")
36     public ModelAndView dashboard() {
37         ModelAndView modelAndView = new ModelAndView("dashboard");
38         Authentication auth = SecurityContextHolder.getContext().getAuthentication();
39         UserDto userDto = userService.findUserByEmail(auth.getName());
40         modelAndView.addObject("currentUser", userDto);
41         modelAndView.addObject("userName", userDto.getFullName());
42         return modelAndView;
43     }
44
45     @GetMapping(value = "/agency")
46     public ModelAndView agencyDetails() {
47         ModelAndView modelAndView = new ModelAndView("agency");
48         Authentication auth = SecurityContextHolder.getContext().getAuthentication();
49         UserDto userDto = userService.findUserByEmail(auth.getName());
50         AgencyDto agencyDto = busReservationService.getAgency(userDto);
51         AgencyFormCommand agencyFormCommand = new AgencyFormCommand()
52             .setAgencyName(agencyDto.getName())
53             .setAgencyDetails(agencyDto.getDetails());
54         modelAndView.addObject("agencyFormData", agencyFormCommand);
55         modelAndView.addObject("agency", agencyDto);
56         modelAndView.addObject("userName", userDto.getFullName());
57         return modelAndView;
58     }
59 }
```

```
TripScheduleDto.java ⌵
1 package com.kirilanastsov.reservationservices.dto.model.bus;
2
3 import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
4 import com.fasterxml.jackson.annotation.JsonInclude;
5 import lombok.Getter;
6 import lombok.NoArgsConstructor;
7 import lombok.Setter;
8 import lombok.ToString;
9 import lombok.experimental.Accessors;
10
11 @Getter
12 @Setter
13 @Accessors(chain = true)
14 @NoArgsConstructor
15 @ToString
16 @JsonInclude(value = JsonInclude.Include.NON_NULL)
17 @JsonIgnoreProperties(ignoreUnknown = true)
18 public class TripScheduleDto {
19
20     private String id;
21
22     private String tripId;
23
24     private String tripDate;
25
26     private int availableSeats;
27
28     private int fare;
29
30     private int journeyTime;
31
32     private String busCode;
33
34     private String sourceStop;
35
36     private String destinationStop;
37 }
--
```

TripRepository.java

```
1 package com.kirilanastsov.reservationservices.repository.bus;
2
3 import org.springframework.data.mongodb.repository.MongoRepository;
4
5 import com.kirilanastsov.reservationservices.model.bus.Agency;
6 import com.kirilanastsov.reservationservices.model.bus.Bus;
7 import com.kirilanastsov.reservationservices.model.bus.Stop;
8 import com.kirilanastsov.reservationservices.model.bus.Trip;
9
10 import java.util.List;
11
12 public interface TripRepository extends MongoRepository<Trip, String> {
13     Trip findBySourceStopAndDestStopAndBus(Stop source, Stop destination, Bus bus);
14
15     List<Trip> findByName(String name);
16
17     List<Trip> findAllBySourceStopAndDestStop(Stop source, Stop destination);
18
19     List<Trip> findByAgency(Agency agency);
20 }
21
```

A black and white photograph of three students in a classroom. Two young men are on the left, one sitting on a stool and leaning over a desk, the other standing behind him. A young woman is standing on the right, looking at a laptop on the desk. The background is a wall with a grid of small holes.

# Classes Lesson Summary

- Class should do one thing only
- Use generalization for abstract classes and interfaces
- Public classes should have small but descriptive names
- Cohesion



# Course Progress

Lesson 1

Clean code

Lesson 2

Names

Lesson 3

Methods

Lesson 4

Classes

Lesson 5

Comments



# Oracle naming convention

- <https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html>

A sepia-toned photograph of a person clapping their hands. In the foreground, a wooden desk holds an open notebook with a smartphone resting on it. A laptop is partially visible in the background. A dark grey rectangular box with a thin white border is overlaid on the left side of the image, containing the text 'THANK YOU!' in a white serif font.

THANK YOU!