NAMES

Java Clean code

Outline

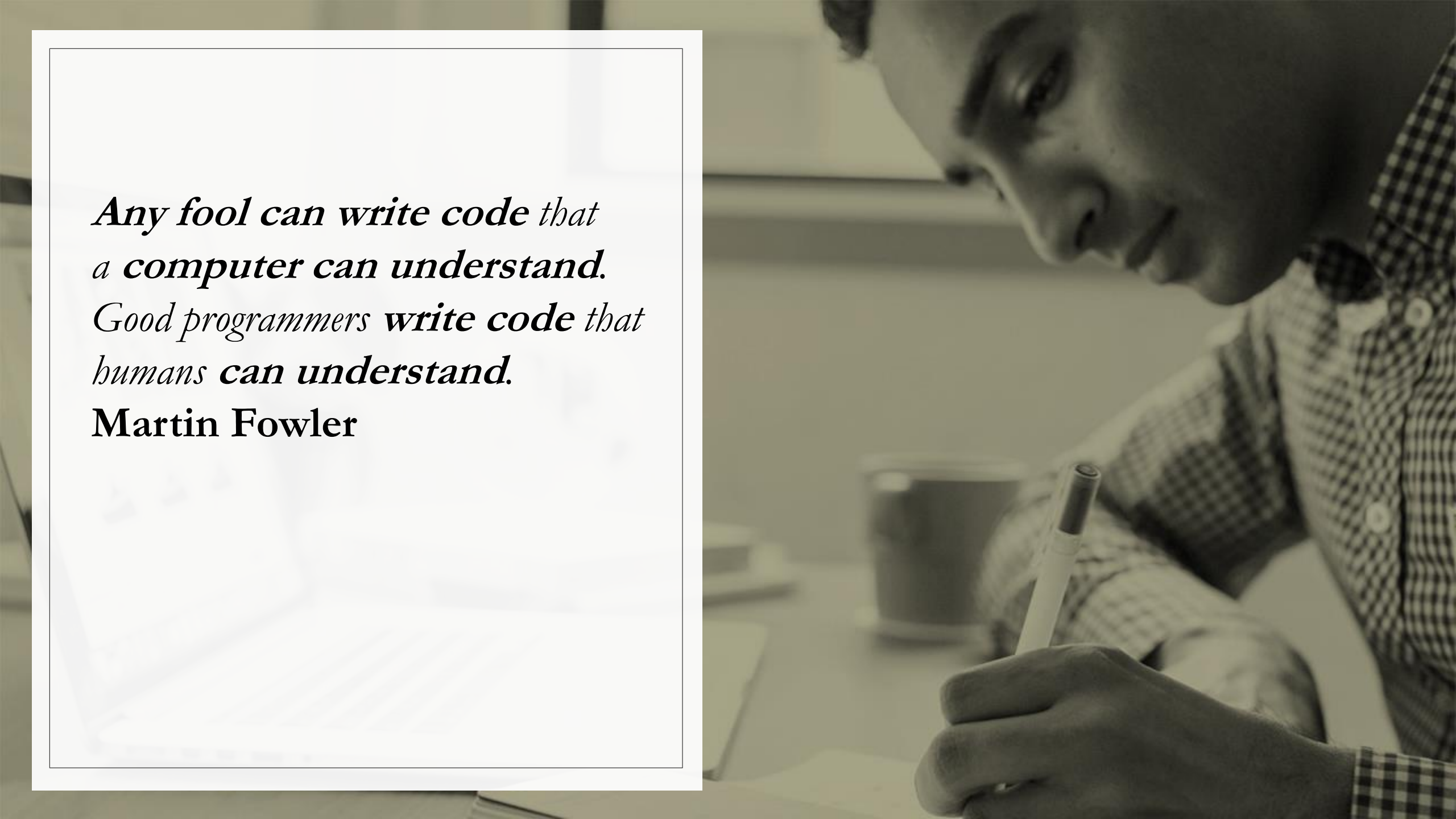
Lesson 1.
Clean code

Lesson 2.
Names

Lesson 3.
Methods

Lesson 4.
Classes

Lesson 5.
Comments

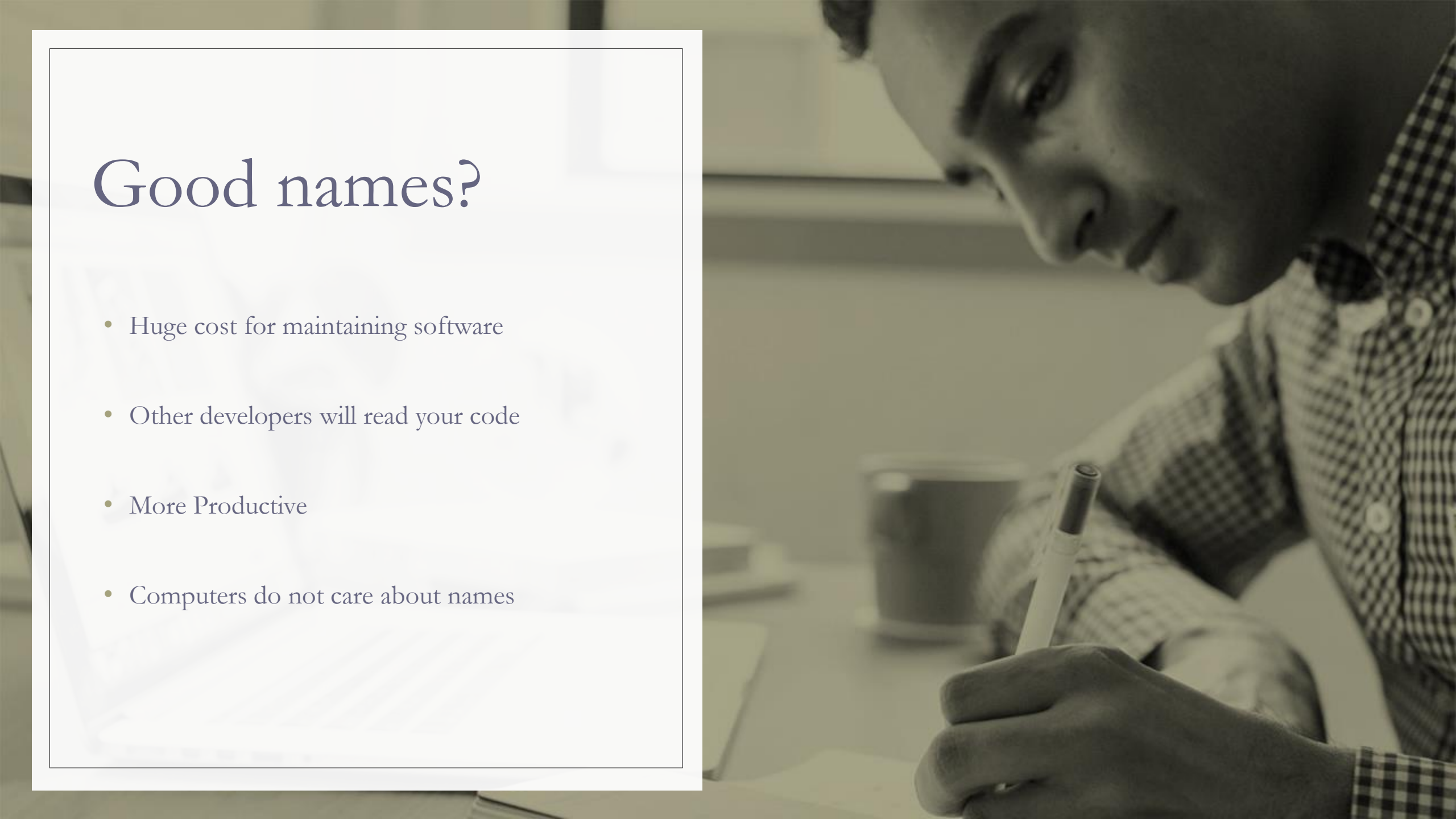


Any fool can write code *that a* ***computer can understand.*** *Good programmers* ***write code*** *that humans* ***can understand.***

Martin Fowler

Good names?

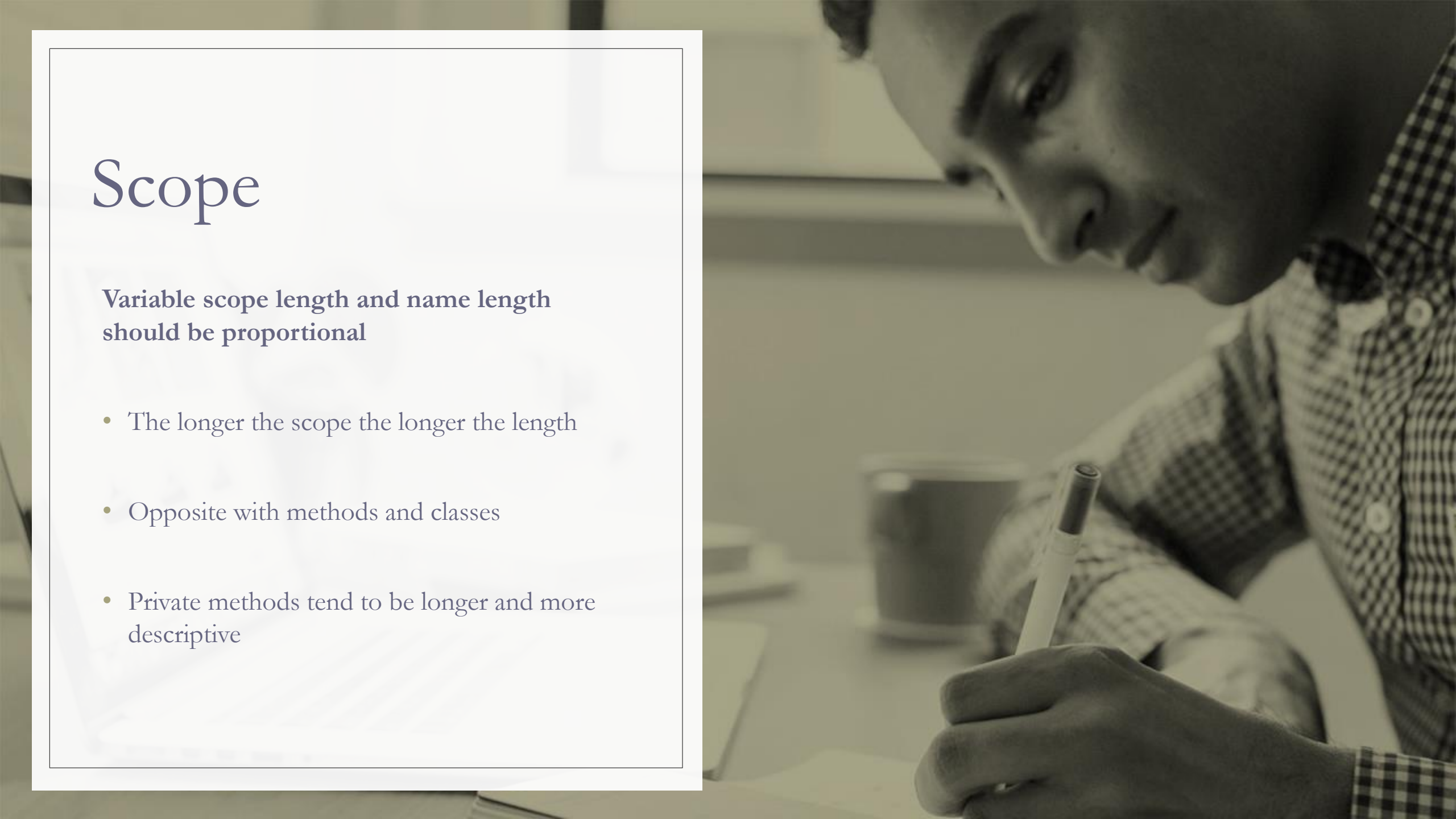
- Huge cost for maintaining software
- Other developers will read your code
- More Productive
- Computers do not care about names



Scope

Variable scope length and name length should be proportional

- The longer the scope the longer the length
- Opposite with methods and classes
- Private methods tend to be longer and more descriptive



Meaningful names

- It takes time to choose meaningful names
- Names should be descriptive
- Other people will read your code
- Searchable names unless in small loops

Choose good names

Names are not just for you but to communicate with others

- If you have to put comment for the name then it is not a good name.
- If you have to read the code than it is not a good name.
- Use names that are easy to pronounce.
- int d ➔ numberOfDays.

Don’t mislead

- employeeList when u don’t use lists ➔ employeeGroups
- calculateWorkHoursWeekly ➔ return payroll
- studentsGroup4, strDays

Variables

- Short and meaningful
- Should not start with _ $
- int speed = 0;
  int gear = 1;

Constants

final

gearRation
➔GEAR_RATIO

private static int
GEAR_RATION = 3;

Rubber duck

- As Einstein said **If you can't explain** it simply, **you** don't understand it well enough.
- Use colleague/rubber duck
- Name as if you are naming your first born child

Antonyms/ opposites

on/dead, lock/open,
min/fast freezing/hot

on/off, lock/unlock,
min/max, cold/hot

Watch out for If, And, Or

readOrWrite() ➔ read(), write()

getPriceAndDate ➔ getPrice(), getDate()

getValueIf() ➔ getValue()

Booleans

- Should sound like a question
- Not a statement
- active, open, flag
- isActive, isOpened, isFlagged

Searchable names

- Avoid names like j,i,x,y (unless used in small loops)
- Easier to search and debug
- Pick names that are also to pronounce

Don’t be funny

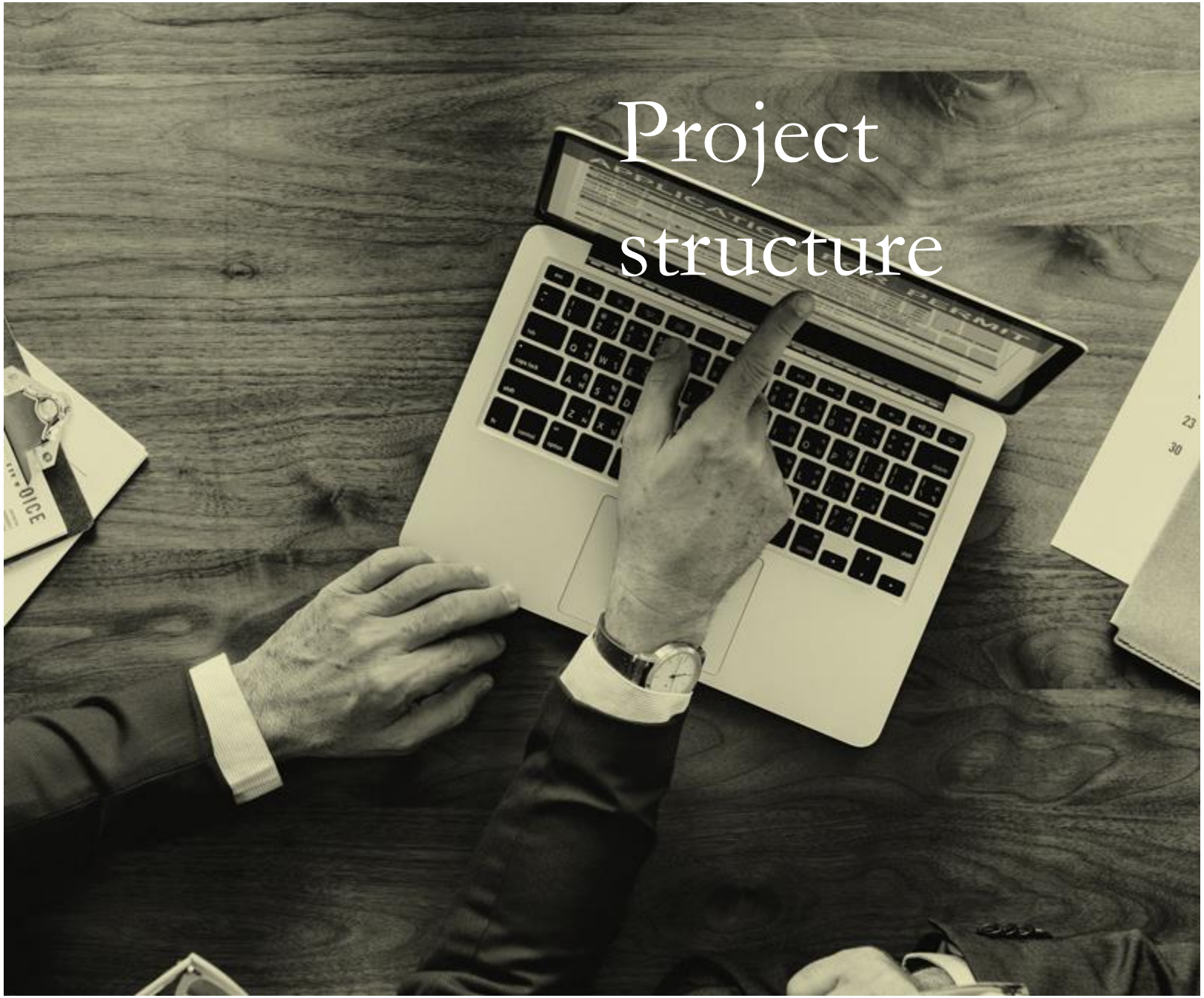
- Use names that are clear to everyone
- sayHelloToMyLittleFriend() ➔ deleteEmployee()
- Don’t use German!!!

Class names

- Should be nouns
- Customer, Employeer or Account
- Don’t use verbs
- Start with capital letters

Consistent

- Use same words for different projects
- Use common words from IT field
- Use common words for the field



Debug | Project Explorer | Servers | JUnit

- Gateways [boot]
  - src/main/java
    - com.kirilanastasov.gateways
    - com.kirilanastasov.gateways.controller
      - GatawayController.java
      - PeripheralDevicesController.java
    - com.kirilanastasov.gateways.exception
      - GatewayAlreadyCreatedException.java
      - GatewayNotFoundException.java
      - NoMoreDeviceAllowedException.java
      - PeripheralDevicesAlreadyCreatedException.java
      - PeripheralDevicesNotFoundException.java
    - com.kirilanastasov.gateways.model
    - com.kirilanastasov.gateways.model.enums
    - com.kirilanastasov.gateways.repository
    - com.kirilanastasov.gateways.service
  - src/main/resources
    - application.properties
    - data.sql
  - src/test/java
    - com.kirilanastasov.gateways
    - com.kirilanastasov.gateways.controller
      - GatewatControllerTest.java
      - PeripheralDevicesControllerTest.java
    - com.kirilanastasov.gateways.model
    - com.kirilanastasov.gateways.service
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - src
  - target
  - HELP.md
  - mvnw
  - mvnw.cmd
  - pom.xml

Method names

Method names should be verbs

calculatePayment(),

showDate()

Getters/setters (getCurrentPrice, setCurrentPrice)

Be Consistent

- Pick same words for different projects
- Use common words for the IT field or the field of the poject.

UserService... ArsConfigur... RoleDto.java Role.java User.java Airplane.java Stop.java

```java
package com.airlineBookingSystem.AmericanAirlines.model.user;

import java.util.Set;

@Getter
@Setter
@Accessors(chain = true)
@NoArgsConstructor
@Document(collection = "user")
public class User {

    @Id
    private String id;
    private String email;
    private String password;
    private String firstName;
    private String lastName;
    private String mobileNumber;

    @DBRef
    private Set<Role> roles;

    public String getFullName() {
        return firstName != null ? firstName.concat(" ").concat(lastName) : "";
    }

}
```

Model Example

UserService... | ArsConfigur... | RoleDto.java | Role.java | User.java | *Airplan

```java
package com.airlineBookingSystem.AmericanAirlines.model.airplane;

import org.springframework.data.annotation.Id;

@Getter
@Setter
@NoArgsConstructor
@Accessors(chain = true)
@Document(collection = "bus")
public class Airplane {
    @Id
    private String id;

    @Indexed(unique = true, direction = IndexDirection.ASCENDING)
    private String code;

    private int capacity;

    private String make;

    @DBRef(lazy = true)
    private Agency agency;
}
```

Model Example

ArsConfigur... | WebSecurityC... | CustomUserC... | User.java | Airplane.java | Trip.java

```java
package com.airlineBookingSystem.AmericanAirlines.controller.api;

import javax.validation.Valid;

@Controller
public class CustomUserController {

    @Autowired
    private CustomUserDetailsService userService;

    @RequestMapping(value = "/login", method = RequestMethod.GET)
    public ModelAndView login() {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("login");
        return modelAndView;
    }

    @RequestMapping(value = "/signup", method = RequestMethod.POST)
    public ModelAndView createNewUser(@Valid User user, BindingResult bindingResult) {
        ModelAndView modelAndView = new ModelAndView();
        User userExists = userService.findUserByEmail(user.getEmail());
        if (userExists != null) {
            bindingResult.rejectValue("email", "error.user",
                    "There is already a user registered with the username provided");
        }
        if (bindingResult.hasErrors()) {
            modelAndView.setViewName("signup");
        } else {
            userService.saveUser(user);
            modelAndView.addObject("successMessage", "User has been registered successfully");
            modelAndView.addObject("user", new User());
            modelAndView.setViewName("login");

        }
        return modelAndView;
    }
```

Controller Example

ArsConfigur... | WebSecurityC... | ArilineBooki... | CustomUserC... | User.java

```java
package com.airlineBookingSystem.AmericanAirlines;

import org.springframework.boot.CommandLineRunner;

@SpringBootApplication
public class ArilineBookingSystemApplication {

    public static void main(String[] args) {
        SpringApplication.run(ArilineBookingSystemApplication.class, args);
    }

    @Bean
    CommandLineRunner init(RoleRepository roleRepository) {
        return args -> {
            Role adminRole = roleRepository.findByRole("ADMIN");
            if (adminRole == null) {
                Role newAdminRole = new Role();
                newAdminRole.setRole("ADMIN");
                roleRepository.save(newAdminRole);
            }
            Role userRole = roleRepository.findByRole("USER");
            if (userRole == null) {
                Role newUserRole = new Role();
                newUserRole.setRole("USER");
                roleRepository.save(newUserRole);
            }
        };

    }
}
```

Starting point

UserSignupR... | *Response.java | ResponseErr... | RoleDto.java | TripMapper.java

```java
package com.airlineBookingSystem.AmericanAirlines.dto.response;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Getter
@Setter
@Accessors(chain = true)
@NoArgsConstructor
@JsonInclude(JsonInclude.Include.NON_NULL)
@JsonIgnoreProperties(ignoreUnknown = true)
public class Response<T> {

    private Status status;
    private T payload;
    private Object errors;
    private Object metadata;

    public static <T> Response<T> badRequest() {
        Response<T> response = new Response<>();
        response.setStatus(Status.BAD_REQUEST);
        return response;
    }

    public static <T> Response<T> ok() {
        Response<T> response = new Response<>();
        response.setStatus(Status.OK);
        return response;
    }

    public static <T> Response<T> unauthorized() {
        Response<T> response = new Response<>();
        response.setStatus(Status.UNAUTHORIZED);
        return response;
    }

    public static <T> Response<T> validationException() {
        Response<T> response = new Response<>();
        response.setStatus(Status.VALIDATION_EXCEPTION);
        return response;
    }

```

Response Example

Names Summary

- Names are not just for you but to communicate with others.
- Names should be clear so that we do not need comments to clarify the meaning.
- Use names that are easy to pronounce.
- Naming your variables is as important as naming your first child.

Course Progress

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5
Clean code	Names	Methods	Classes	Comments

Oracle naming convention

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


THANK YOU!