METHODS

Java Clean code

Outline

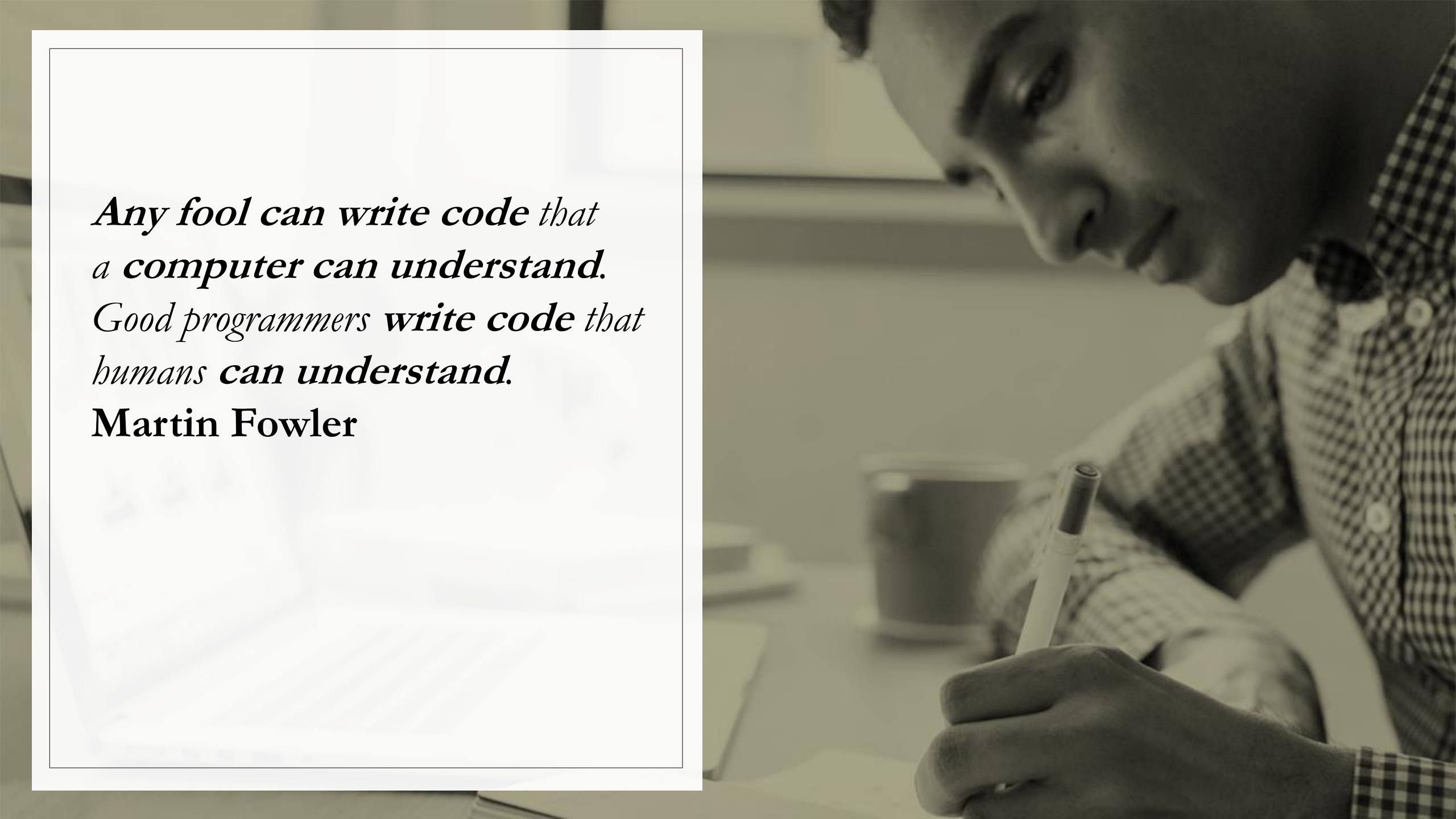
Lesson 1.
Clean code

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

Methods/Functions

Code is easier to read when in methods

- Methods should be responsible for one thing
- All the code goes in the methods
- Avoid nested if and try stamens in methods.
- Methods allow us reuse of code.

Methods size

- Screen fall method
- Methods should be small
- Strive to have 10 lines
- Descriptive names

Is it effective

- New programmers start faster
- Less Debug time
- Efficiency

Large methods

- Save yourself a bit of time
- Large methods violate the SRP
- Large methods need to be refactored
- Methods should do one thing do it well and do it only

Format date

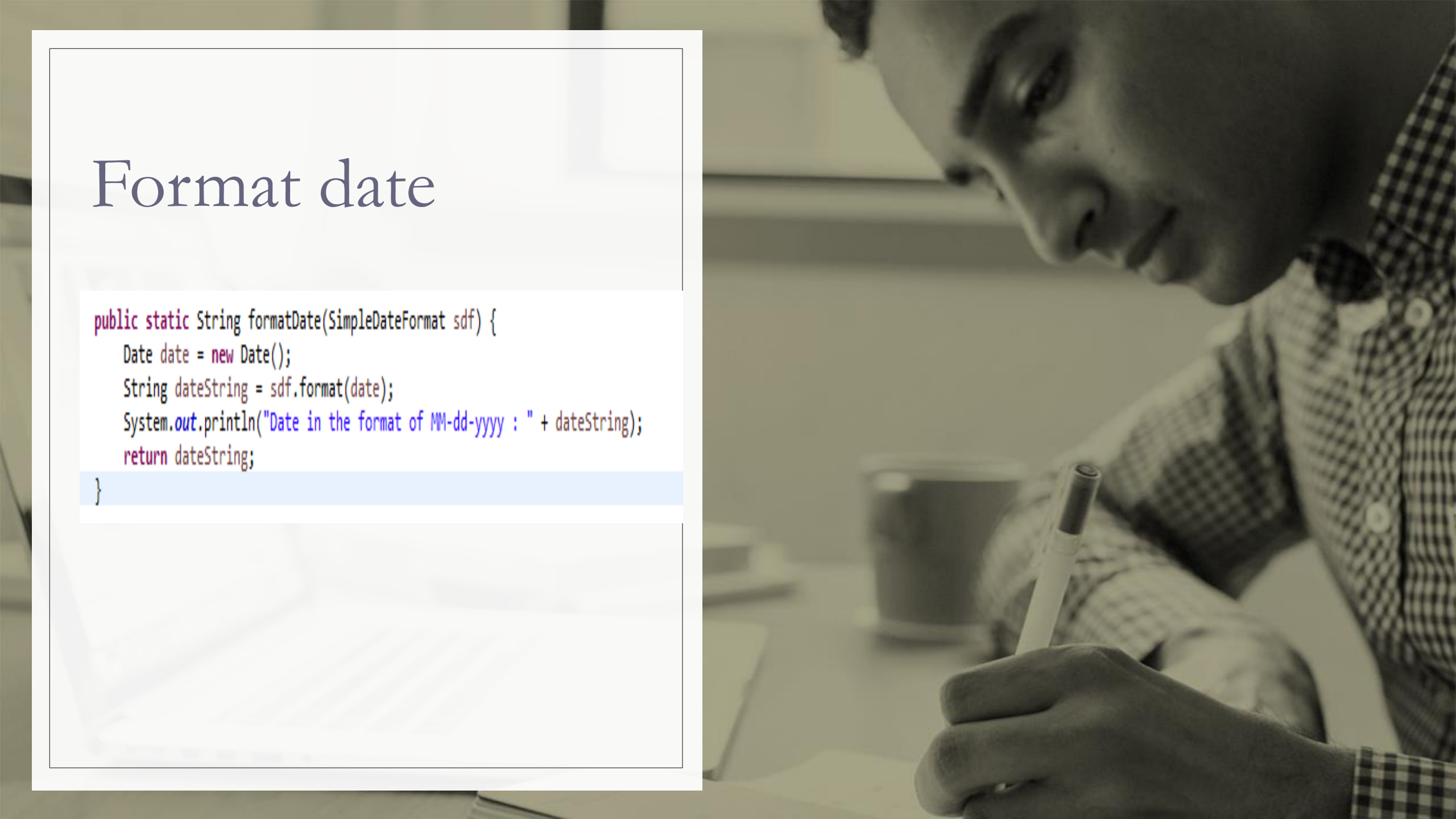
```
SimpleDateFormat sdf = new SimpleDateFormat("MM-dd-yyyy");
Date date = new Date();
String dateString = sdf.format(date);
System.out.println("Date in the format of MM-dd-yyyy : " + dateString);

sdf = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss");
date = new Date();
dateString = sdf.format(date);
System.out.println("Date in the format of MM-dd-yyyy : " + dateString);

sdf = new SimpleDateFormat("E, dd/MMM/yyyy HH:mm:ss z");
date = new Date();
dateString = sdf.format(date);
System.out.println("Date in the format of MM-dd-yyyy : " + dateString);
```

Format date

```
public static String formatDate(SimpleDateFormat sdf) {
    Date date = new Date();
    String dateString = sdf.format(date);
    System.out.println("Date in the format of MM-dd-yyyy : " + dateString);
    return dateString;
}
```



Save driver

```
public void saveDriver(String firstname,
String lastname, String height, String
phoneNumber, String email, int workHours,
boolean isSingle ) {
        Driver driver= new Driver();
        driver.setFirstName(firstname);
        driver.setLastName(lastName);
        driver.setHeight(height);
        driver.setPhoneNumber(phoneNumber);
        driver.setEmail(email);
        driver.setWorkHours(workHours);
        driver.setIsSingle(isSingle);
        driverRepository.save(driver);
          model.addAttribute("driver",
driverRepository.findAll());
          return "redirect:/index";
        }
```

Save driver

```
public void saveDriver(Driver driver) {
        driverRepository.save(driver);
         model.addAttribute("driver",
driverRepository.findAll());
         return "redirect:/index";
}
```

UserReposit... | CustomUserD... | *Response.java | ResponseErr... | Airplane.java | Agency.java | RoleReposit...

```
    public User findUserByEmail(String email) {
        return userRepository.findByEmail(email);
    }

    public void saveUser(User user) {
        user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));
        Role userRole = roleRepository.findByRole("ADMIN");
        user.setRoles(new HashSet<>(Arrays.asList(userRole)));
        userRepository.save(user);
    }

    public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {

        User user = userRepository.findByEmail(email);
        if (user != null) {
            List<GrantedAuthority> authorities = getUserAuthority(user.getRoles());
            return buildUserForAuthentication(user, authorities);
        } else {
            throw new UsernameNotFoundException("username not found");
        }
    }

    private List<GrantedAuthority> getUserAuthority(Set<Role> userRoles) {
        Set<GrantedAuthority> roles = new HashSet<>();
        userRoles.forEach((role) -> {
            roles.add(new SimpleGrantedAuthority(role.getRole()));
        });

        List<GrantedAuthority> grantedAuthorities = new ArrayList<>(roles);
        return grantedAuthorities;
    }

    private UserDetails buildUserForAuthentication(User user, List<GrantedAuthority> authorities) {
        return new org.springframework.security.core.userdetails.User(user.getEmail(), user.getPassword(), authorities);
    }
```

UserService... | TempService.... | CustomUserD... | UserService.... | Role.java | User.java | Agency.java | RoleReposit...

```java
    public void saveUser(User user) {
        user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));
        Role userRole = roleRepository.findByRole("ADMIN");
        user.setRoles(new HashSet<>(Arrays.asList(userRole)));
        userRepository.save(user);
    }

    @Override
    public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {

        User user = userRepository.findByEmail(email);
        if (user != null) {
            List<GrantedAuthority> authorities = getUserAuthority(user.getRoles());
            return buildUserForAuthentication(user, authorities);
        } else {
            throw new UsernameNotFoundException("username not found");
        }
    }

    private List<GrantedAuthority> getUserAuthority(Set<Role> userRoles) {
        Set<GrantedAuthority> roles = new HashSet<>();
        userRoles.forEach((role) -> {
            roles.add(new SimpleGrantedAuthority(role.getRole()));
        });

        List<GrantedAuthority> grantedAuthorities = new ArrayList<>(roles);
        return grantedAuthorities;
    }

    private UserDetails buildUserForAuthentication(User user, List<GrantedAuthority> authorities) {
        return new org.springframework.security.core.userdetails.User(user.getEmail(), user.getPassword(), authorities);
    }

```

PropertiesC... | TicketMappe... | TripMapper.java | TripSchedul... | UserMapper.java

```java
package com.kirilanastasov.reservationservices.dto.mapper;

import com.kirilanastasov.reservationservices.dto.model.user.RoleDto;

@Component
public class UserMapper {

    public static UserDto toUserDto(User user) {
        return new UserDto()
                .setEmail(user.getEmail())
                .setFirstName(user.getFirstName())
                .setLastName(user.getLastName())
                .setMobileNumber(user.getMobileNumber())
                .setRoles(new HashSet<RoleDto>(user
                        .getRoles()
                        .stream()
                        .map(role -> new ModelMapper().map(role, RoleDto.class))
                        .collect(Collectors.toSet())));
    }

}
```

```java

    public ApiJWTAuthenticationFilter(AuthenticationManager authenticationManager) {
        this.authenticationManager = authenticationManager;
        this.setRequiresAuthenticationRequestMatcher(new AntPathRequestMatcher("/api/auth", "POST"));
    }

    @Override
    public Authentication attemptAuthentication(HttpServletRequest req, HttpServletResponse res)
            throws AuthenticationException {
        try {
            User user = new ObjectMapper().readValue(req.getInputStream(), User.class);
            return authenticationManager.authenticate(
                    new UsernamePasswordAuthenticationToken(user.getEmail(), user.getPassword(), new ArrayList<>()));
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
```

```
47
48      @Override
49      protected void successfulAuthentication(HttpServletRequest req, HttpServletResponse res, FilterChain chain,
50              Authentication auth) throws IOException, ServletException {
51          if (auth.getPrincipal() != null) {
52              org.springframework.security.core.userdetails.User user = (org.springframework.security.core.userdetails.User) auth
53                      .getPrincipal();
54              String login = user.getUsername();
55              if (login != null && login.length() > 0) {
56                  Claims claims = Jwts.claims().setSubject(login);
57                  List<String> roles = new ArrayList<>();
58                  user.getAuthorities().stream().forEach(authority -> roles.add(authority.getAuthority()));
59                  claims.put("roles", roles);
60                  String token = Jwts.builder().setClaims(claims)
61                          .setExpiration(new Date(System.currentTimeMillis() + EXPIRATION_TIME))
62                          .signWith(SignatureAlgorithm.HS512, SECRET).compact();
63                  res.addHeader(HEADER_STRING, TOKEN_PREFIX + token);
64              }
65          }
66      }
67  }
68
```

Method Structure

- 3 Arguments max
- Avoid Booleans and Nulls
- Methods structure

Methods Lesson Summary

- Methods should be small
- Well named methods will save everyone time
- Methods do one thing!
- Should have 3 arguments max
- Should not pass Booleans and nulls

Course Progress

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


THANK YOU!