WELCOME BACK
DAY 3

<Cre8tive Devs. Software/>

Spring/Spring Boot Crash Course

Cohort 1

For TD Bank

MEET YOUR CRASH COURSE TEAM

TANGY F.
CEO

HAL M.
DEVELOPER

WILLIAM
DEVELOPER

<Cre8tive Devs. Software/>
MOBILE APPS•WEBSITES•AR•VIDEO GAMES•SECURITY

AGENDA

DAY 3

1. Recap Day two:
   >>Rapid review of day two

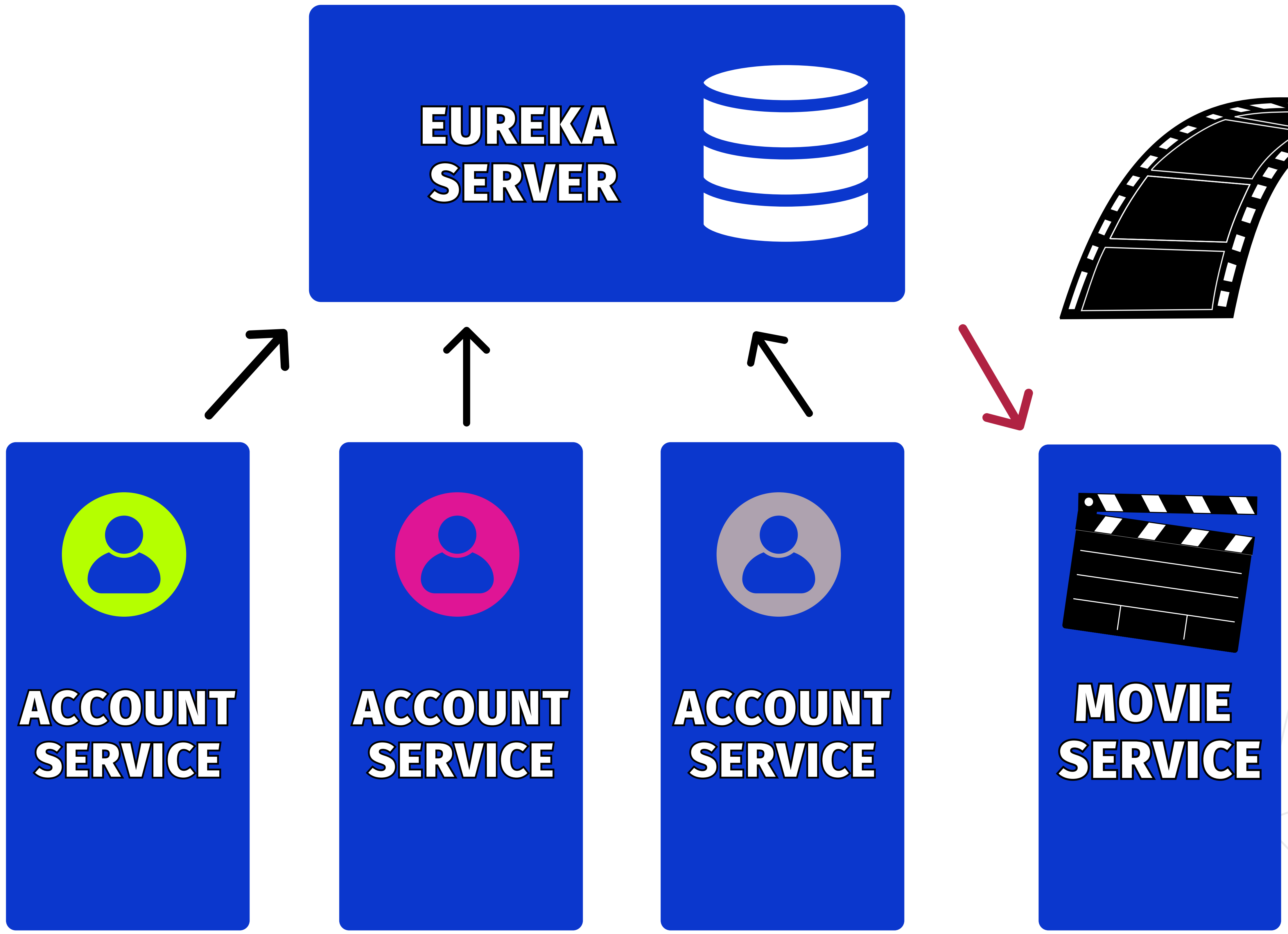
2. Lesson- Spring Security I

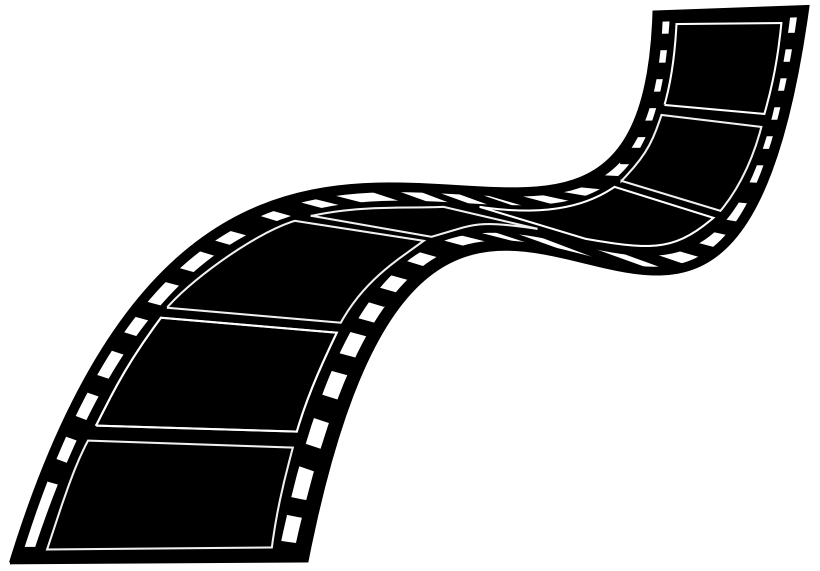
3. Q.A.

<Cre8tive Devs. Software/>

WE ARE BUILDING

BITE SIZE
MOVIE REVIEW APP





RAPID REVIEW

Trivia

Cre8tive Devs. Software/
MOBILE APPS•WEBSITES•AR•VIDEO GAMES•SECURITY

Rapid Review of **LESSON 1**

Rapid Review Trivia

Why do we use a DTO instead of just accessing the entity directly?

<Cre8tive Software/> Devs.
MOBILE APPS•WEBSITES•AR•VIDEO GAMES•SECURITY

Rapid Review
of
LESSON 2

Rapid Review Trivia

What would be the other ways to access the DataBase other than the one we discussed?

LESSON 3

Spring Security I

LESSON 3

PASSWORD ENCRYPTION

1. Why Spring Security
2. Password Encryption
3. Types of Password Encryptions Available
4. Other Security methods in Spring Boot

KEEP A LOOK OUT FOR 👀

12 FACTOR

Factor 2: Dependencies

Watch for how we declare Spring Security dependency, and how we don't require Spring Security dependencies to be installed system-wide.

Factor 3: Config

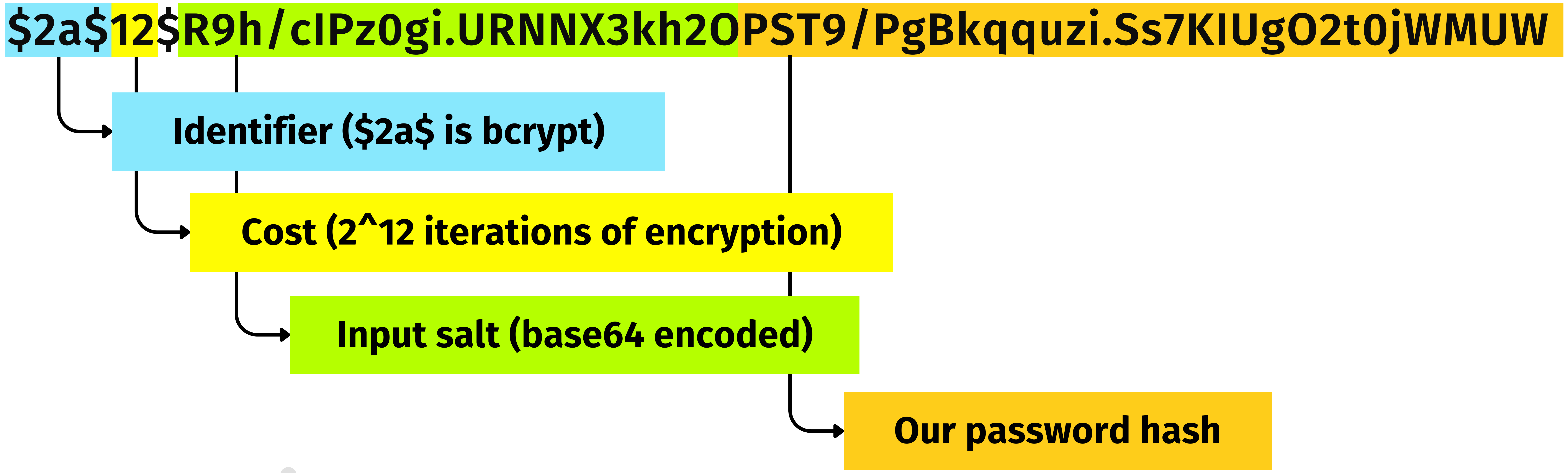
Watch for where we store our security config information, and consider how we could share this code to a repository without exposing sensitive information.

DESIGN PATTERN

Singleton

BCRYPT PASSWORD FORMAT

LESSON 3


$2a$12$R9h/cIPz0gi.URNNX3kh2OPST9/PgBkqquzi.Ss7KIUgO2t0jWMUW
Identifier ($2a$ is bcrypt)
Cost (2^12 iterations of encryption)
Input salt (base64 encoded)
Our password hash

Password Encryption

BCryptPasswordEncoder

Argon2PasswordEncoder

Pbkdf2PasswordEncoder

SCryptPasswordEncoder

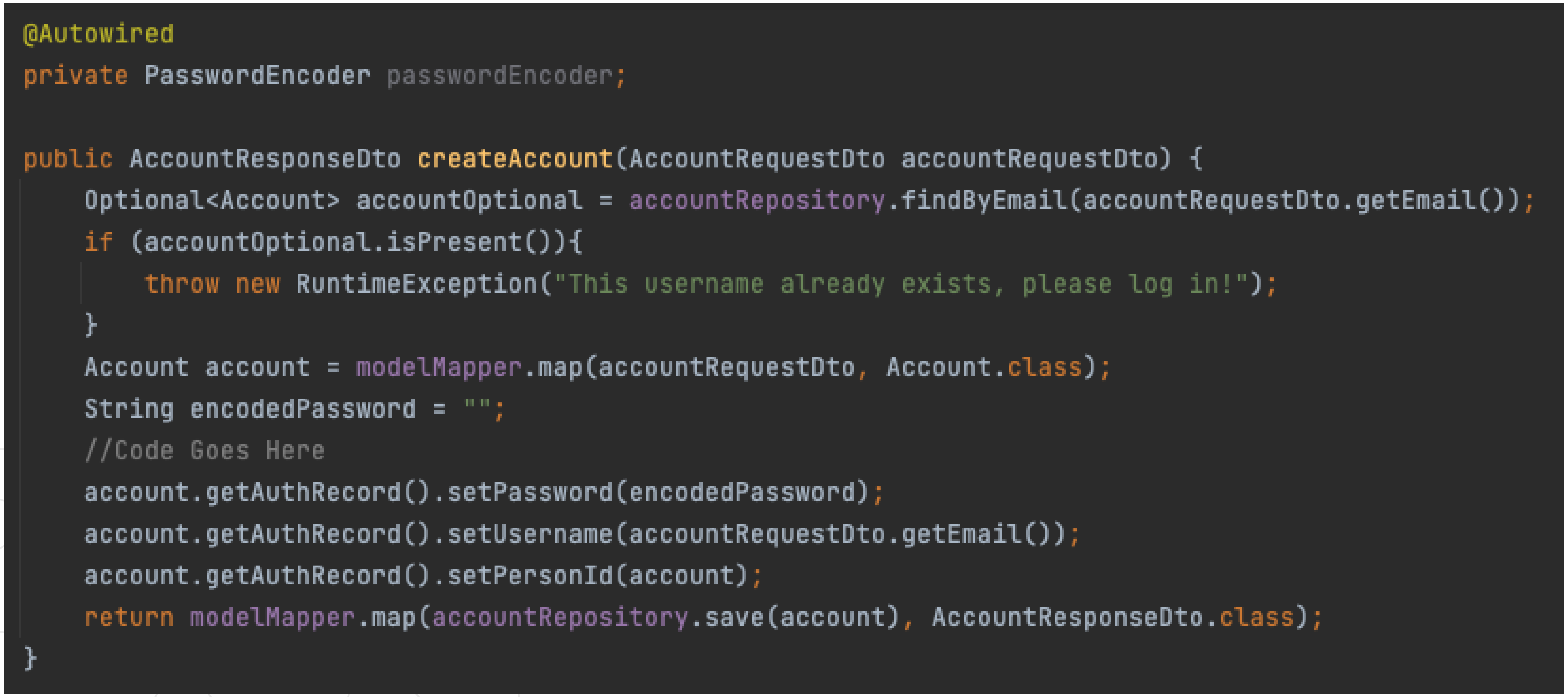
Other Security Methods

OAuth2.0

JWT

CODING EXCERCISE #1

The below code is intended to save a new account to the database, along with an encrypted password. Write one line of code, to be inserted into the function below, that will save an encrypted version of the password entered by the user into the string encodedPassword.

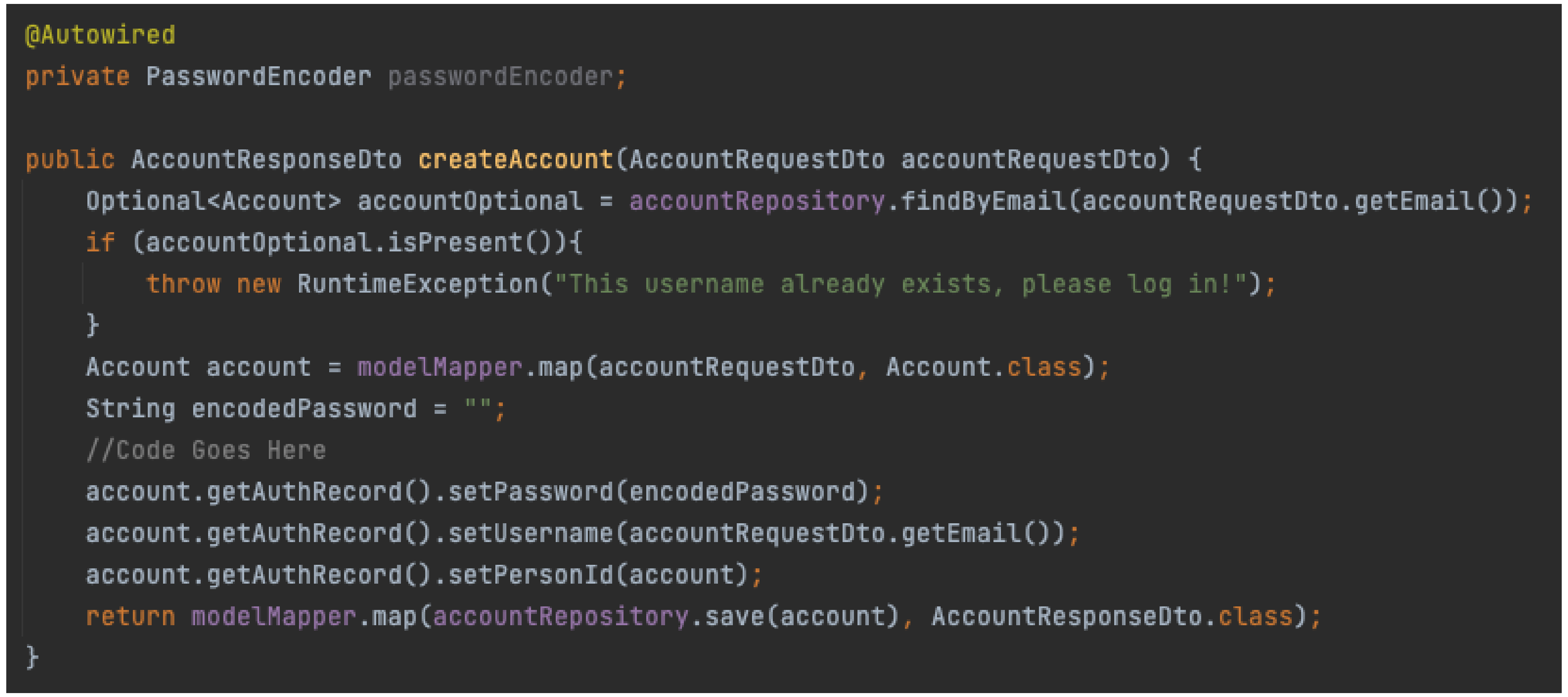


```java
@Autowired
private PasswordEncoder passwordEncoder;

public AccountResponseDto createAccount(AccountRequestDto accountRequestDto) {
    Optional<Account> accountOptional = accountRepository.findByEmail(accountRequestDto.getEmail());
    if (accountOptional.isPresent()){
        throw new RuntimeException("This username already exists, please log in!");
    }
    Account account = modelMapper.map(accountRequestDto, Account.class);
    String encodedPassword = "";
    //Code Goes Here
    account.getAuthRecord().setPassword(encodedPassword);
    account.getAuthRecord().setUsername(accountRequestDto.getEmail());
    account.getAuthRecord().setPersonId(account);
    return modelMapper.map(accountRepository.save(account), AccountResponseDto.class);
}
```

CODING EXCERCISE #1

The below code is intended to save a new account to the database, along with an encrypted password. Write one line of code, to be inserted into the function below, that will save an encrypted version of the password entered by the user into the string encodedPassword.



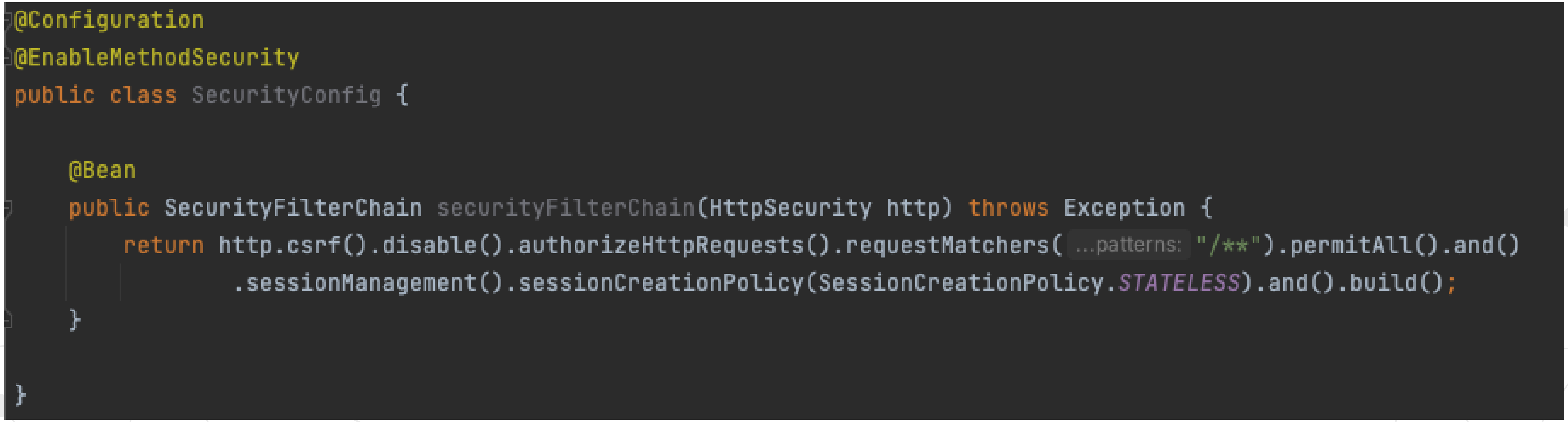
```java
@Autowired
private PasswordEncoder passwordEncoder;

public AccountResponseDto createAccount(AccountRequestDto accountRequestDto) {
    Optional<Account> accountOptional = accountRepository.findByEmail(accountRequestDto.getEmail());
    if (accountOptional.isPresent()){
        throw new RuntimeException("This username already exists, please log in!");
    }
    Account account = modelMapper.map(accountRequestDto, Account.class);
    String encodedPassword = "";
    //Code Goes Here
    account.getAuthRecord().setPassword(encodedPassword);
    account.getAuthRecord().setUsername(accountRequestDto.getEmail());
    account.getAuthRecord().setPersonId(account);
    return modelMapper.map(accountRepository.save(account), AccountResponseDto.class);
}
```

Answer: String encodedPassword = passwordEncoder.encode(accountRequestDto.getAuthRecord().getPassword());

CODING EXERCISE #2

Below is the security config class for our Account Service application, intended to disable Spring Security's default login so we can code our own later. Which part of this code is incorrect?



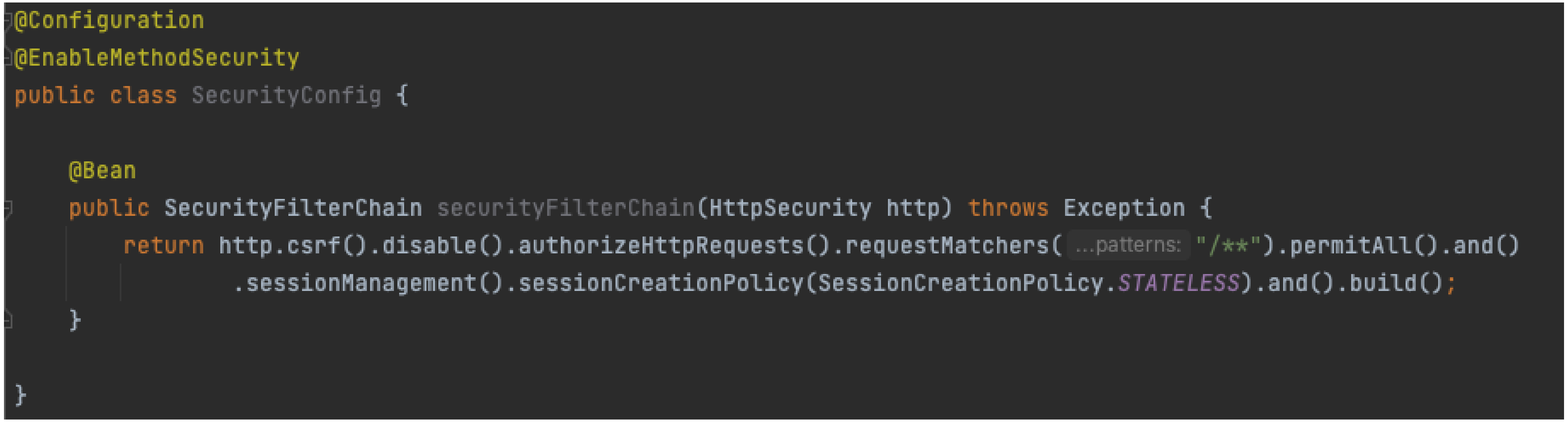
```
@Configuration
@EnableMethodSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        return http.csrf().disable().authorizeHttpRequests().requestMatchers( ...patterns: "/**").permitAll().and()
                .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and().build();
    }

}
```

CODING EXERCISE #2

Below is the security config class for our Account Service application, intended to disable Spring Security's default login so we can code our own later. Which part of this code is incorrect?



```
@Configuration
@EnableMethodSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        return http.csrf().disable().authorizeHttpRequests().requestMatchers( ...patterns: "/**").permitAll().and()
                    .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and().build();
    }

}
```

Answer:
Web Security is not enabled. The class declaration should look like the following:

@Configuration
@EnableWebSecurity
@EnableMethodSecurity
public class SecurityConfig {

THANK YOU

<Cre8tive Devs. Software/>

Crash Course

Tomorrow