WELCOME BACK
LESSON 10

Spring/Spring Boot Crash Course

For TD Bank

YOUR CRASH COURSE TEAM WOULD LIKE TO THANK YOU

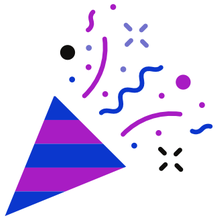
TANGY F.
CEO

HAL M.
DEVELOPER

WILLIAM D.
DEVELOPER

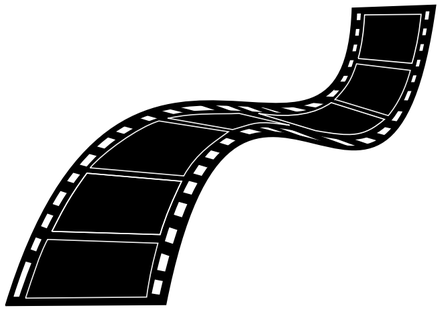
LESSON 10

AGENDA

1. Rapid Review 
2. Rapid Trivia 
3. Pluralsite IQ *
   Post Assessment 

Creative Devs. Software

BITE SIZE
MOVIE REVIEW APP



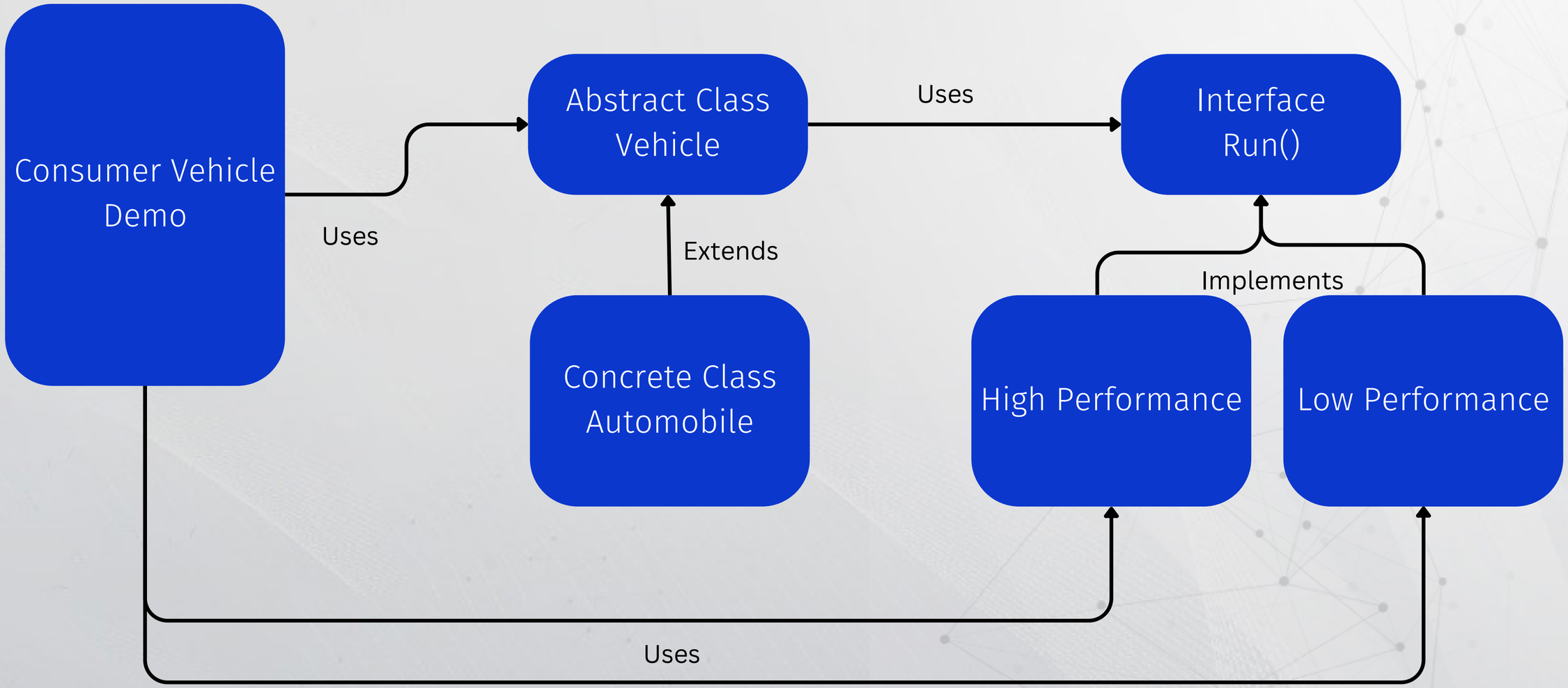
WE HAVE BUILT

Creative Devs. Software

LESSON 10

Rapid Review
Bridge Pattern

BRIDGE PATTERN ABSTRACT



LESSON 10

Review of Lessons 1 - 9

LESSON 10
Review

Dependency Injection

>>Achieving loose coupling between classes by removing the responsibility of creating and managing dependencies in the class itself.
>>Different modes of Dependency Injection that we have are Constructor Injection, Setter Injection and Field Injection

Rest Services

>>Creating the entity classes for database objects.
>>Creating a REST Controller and defining the Http methods using the mappings.

LESSON 2 & 3

LESSON 10
Review

JPA Implementation

>>Configuring the database properties in properties file.
>>Creating the repository class using JpaRepository.
Implementing the Service and Controller Class using the ModelMapper

Password Encryption

>>We can encrypt the password before storing into DB using different encryption methods available in Spring Security.
>>The encryption methods that we have are:
>>>BCryptPasswordEncoder
>>>Argon2PasswordEncoder
>>>bkdf2PasswordEncoder
>>>SCryptPasswordEncoder

LESSON 4

LESSON 10
Review

Micro Services

>>The division of application in to small services which concentrate on specific business logic.
>>Various advantages of Micro services like Scalability, CI & CD, Maintainability etc.,

Song Service

>>Implementing a gradle project with the entities, DB Connection and implementing the REST Services.

Gradle

>>Build automation tool that offers flexibility, extensibility and ease of use.
>>Gradle is known for performance optimizations, incremental builds and build caching

LESSON 5 & 6

LESSON 10 Review

Authentication

Implementing the RestTemplate to get the repsonse from the Profile Service Url and validate the credentials using Authentication Manger.

JWT Token

>>Implementing the JwtAuthService with all the methods to create the JWT Token using the username.
>>Digital signature using public/ Private key.

Authorization

>>Adding preAuthorization to selected methods based on the roles.
>>Performing the authorization using the JWT token generated and implementing the methods in JwtService methods like isTokenvalid, isTokenExpired to get validations performed

LESSON 7

LESSON 10
Review

JUnit & Mockito

>>JUnit is a testing framework which provides set of annotations and assertions to implements these tests.
>>Mockito is a mocking/duping technique used in JUnit.

Hibernate

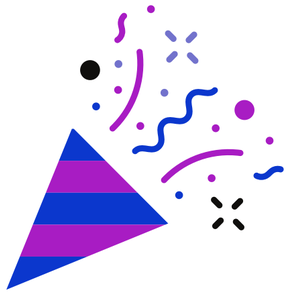
>>Hibernate offers more flexibility than JPA like caching, lazy loading and HQL
>>Hibernate is nothing but a implementation of JPA for ORM.

Eureka Server

>>It is famous Netflix Discovery Server used to given the number of instances running and its health metrics.

>>Configuring the instances to the eureka server as clients.

LESSON 10



Type your answers on a document & email them to us.

QUESTIONS 1-3

LESSON 10

Below are some declared dependencies for a Gradle project:

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.modelmapper:modelmapper:3.1.1'

    compileOnly 'org.projectlombok:lombok'
    implementation 'io.jsonwebtoken:jjwt-api:0.11.5'
    runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.11.5'
    runtimeOnly 'io.jsonwebtoken:jjwt-jackson:0.11.5'
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.1.0'
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    implementation 'io.github.resilience4j:resilience4j-spring-boot3:2.0.2'
    implementation 'org.springframework.boot:spring-boot-starter-aop'
    implementation 'org.springframework.boot:spring-boot-starter-actuator'
    implementation 'org.springframework.cloud:spring-cloud-starter-loadbalancer'

    runtimeOnly 'com.mysql:mysql-connector-j'
//Question 1 code goes here
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'org.springframework.security:spring-security-test'
//Question 2 code goes here

}
```

1. Describe how you would add an annotation processor dependency for lombok, from the group projectLombok.

2. Describe how you would add a dependency for hibernate-validator version 8.0.0.Final, from the group hibernate.validator.

3. What would the name of the gradle dependency file be in a Spring Boot project?

QUESTION 4-6

Below is the stub of a class called Profile. Below you will see a prompt that shows where the code can go. Please note that there is another way to do this.

```
public class Profile {
 //Code 4-6 goes here
}
```

4. How would you add fields named email (a String), password (a String), and roles (a String)?
5. How would you add a constructor for the class that takes no arguments, and another constructor that takes arguments for every field?
6. How would you add getters and setters for every field in this class?

QUESTION 7

LESSON 10

Below is a section of our SongService class.
Please note that we have a DTO class called SongRequestDto, a DTO class called SongResponseDto, and an entity class called Song.

```
@Autowired
private SongRepository songRepository;

@Autowired
ModelMapper modelMapper;

public SongResponseDto
uploadSong(SongRequestDto songRequestDto) {
        //Code goes here
}
```

7. Write code that will add the song data into our repository, then return the song's data.

LAST QUESTION 8

LESSON 10

Below is a section of our SongController class.

```
@Autowired
private SongService songService;

@PostMapping(value = "/upload")
@Operation(summary = "Upload song", description =
"Upload song")
 @SecurityRequirement(name = "Bearer Authentication")
public ResponseEntity<SongResponseDto>
 uploadSong(@RequestBody @Valid SongRequestDto
songRequestDto) {

//Code goes here
}
```

8. Write code that will add the song data into our repository, returning the song's data and an Http Status object indicating that the song data was accepted successfully.

HEARTFELT GRATITUDE
FOR YOUR PARTICIPATION
FROM
CRE8TIVE DEVS

Cre8tive Devs. Software/>

Crash Course
Spring/ Spring Boot
Cohort 1

CERTIFICATE
OF
COMPLETION