Sample Questions for TechM Java Track

MCQ:

Topic: GIT, Version Control System

Description:

You have made few commits in last 2 weeks. You need to get a log of that. Which of the below command will retrieve the log of 2 weeks?

Option 1: git log --since-weeks=2

Option 2: git log --since="2 weeks ago" [Correct Answer]

Option 3: git log --since=2.w

Option 4: git log --weeks-since=2

Topic: Maven

Description:

In a Maven project, which lifecycle phase processes and deploys the package (if needed) before running integration tests?

Option 1: Integration-test

Option 2: Process-resources

Option 3: Pre-integration-test [Correct Answer]

Option 4: All of the above

Topic: JSP

Problem Description 1:

State whether TRUE or FALSE:

JSP pages provide a means to create **dynamic Web pages** using **HTML** and the **Java programming language.**

Option 1: True [Correct Answer]

Option 2: False

Problem Description 2:

A JSP custom tag was implemented and a class was extended to override a method.

The method that must be overridden here is:

Option 1: getTag()

Option 2: doTag() [Correct Answer]

Option 3: makeTag()

Option 4: setTag()

Topic: Stream API

Description

Determine the correct output of the code snippet given below.

```
import java.util.stream.Stream;
class Test
{
 public static void main(String[] args)
 {
  Stream<String> stream = Stream.of("A", "B","C","D");
  System.out.println(stream.peek(System.out::print).findAny().orElse("NA"));
 }
}
```

Option 1: ABCDNA

Option 2: ABCD

Option 3: NA

Option 4: AA [Correct Answer]

Topic: Spring AOP

Description

The below program is an example of a?

```
public class exampaper
{
public Exammm getsubjectId(Integer examid) {
System.out.println("method pof exam id");
return new exammm();
}
}
public class exam
{
@SuppressWarnings("resource")
public static void main(String[] args) {
ApplicationContext context = new ClassPathXmlApplicationContext
("give your exam");
exampaper ex = context.getBean(exampaper.class);
manager.getsubjectId(5);
}
}
```

Option 1: AOP[Correct Answer]

Option 2: DI and IOC

Option 3: Factory Bean

Option 4: Bean Scope

Topic: JPA/Hibernate

Description

What @**Entity association** is implemented by the below code?

```
@Entity
public class Exam {
   @Id
   private int marks;
}
@Entity
public class Result{
   @Id
   private int marks;
   @OneToOne
   private result;
}
```

Option 1: @OneToOne no directional attribute specified.

Option 2: @OneToOne unidirectional. [Correct Answer]

Option 3: @OneToOne bidirectional.

Option 4: Both 2 and 3

Topic: SpringBoot

Description
You need to implement authentication and authorization in your Spring MVC application. You want to secure certain URLs and restrict access based on user roles.
Which features of Spring MVC would you use to fulfill these requirements?

Option 1: Session management

Option 2: Spring Security [Correct Answer]

Option 3: Dependency Configuration

Option 4: Controller advice

Option 5: Request mapping annotations [Correct Answer]

Topic: CI/CD

Description

Which of the following is(are) typical benefit(s) of using containers compared to virtual machines?

Option 1: Containers offer the ability to run microservice applications while VMs don't.

Option 2: Containers are more lightweight than VMs [Correct Answer]

Option 3: Containers provide stronger isolation than VMs.

Option 4: Containers provide faster start-up time and are ideal for scaling up applications. [Correct Answer]

Topic: Jenkins

Description

The administrator is responsible for upgrading Jenkins plugins, but network restrictions prevent server IP access. Also, traditional methods of accessing the server are not helpful without a direct route to initiate plugin upgrades.

What alternative method would you use to upgrade Jenkins plugins in this scenario?

Option 1: Change the IP address of Network and delete the log file to ensure that outbound connections are allowed for plugin downloads.

Option 2: Inspect the Jenkins log files and investigate the network-related issues or server misconfigurations. [Correct Answer]

Option 3: Disable the firewall software running on the server temporarily to eliminate potential interference with plugin downloads.

Option 4: Manually download the plugin updates from the Jenkins website and upload them to the server for installation, bypassing the download error.

Java Coding:

Description

Problem Statement

You are building an application that helps users organize their seasonal activities based on the weather conditions. You want to use **EnumSet** to efficiently manage the activities for each season.

-> Create a **Season enum** that includes various types of seasons **SPRING**, **SUMMER**, **AUTUMN** (Fall), and **WINTER**.

->Create an **Activity enum** that includes various types of activities like **HIKING, SWIMMING, SKIING, PUMPKIN_CARVING**

class SeasonalActivityOrganizer

->Your task is to implement the following:

->Create an **EnumSet** for each season to store the activities that are appropriate for that season.

->getActivitiesForSeason(Season season):

- This method takes a Season enum value as input and returns a set of Activity enum values that are suitable for the given season.

  The default case in the switch statement should include appropriate error handling for cases where an unknown or unsupported season value is provided. ("Unknown season: " + season)

->addActivityForSeason(Activity activity, Season season):

- This method adds an Activity enum value to the set of activities appropriate for the specified season and then returns the updated set of activities for that season("Unknown season: " + season).

  The default case in the switch statement should include appropriate error handling for cases where an unknown or unsupported season value is provided("Unknown season: " + season).

->removeActivityFromAllSeasons(Activity activity):

- This method removes an Activity enum value from the set of activities for all seasons if it exists and then returns the updated set of all activities.

->getAllActivities():

- This method returns a set containing all the Activity enum values from all seasons.

Sample Input

```
SeasonalActivityOrganizer organizer = new SeasonalActivityOrganizer();
   organizer.addActivityForSeason(Activity.HIKING, Season.SPRING);
   organizer.addActivityForSeason(Activity.SWIMMING, Season.SUMMER);
   organizer.addActivityForSeason(Activity.SKIING, Season.WINTER);

organizer.getAllActivities()

organizer.getActivitiesForSeason(Season.SPRING)
organizer.getActivitiesForSeason(Season.SUMMER)
organizer.getActivitiesForSeason(Season.WINTER)

organizer.removeActivityFromAllSeasons(Activity.HIKING);

organizer.getActivitiesForSeason(Season.SPRING)
```

Sample Output

```
[HIKING, SWIMMING, SKIING]

[HIKING]
[SWIMMING]
[SKIING]

[]
```

NOTE:

- You can make suitable function calls and use **the RUN CODE** button to check your **main()** method output.

JSP Full Stack:

Description

Problem Description

JSP Pages

index.jsp:

Create a task creation form with the following fields:

- Task Title (input type text)
- Task Description (textarea)
- Due Date (input type date)
- Priority (select dropdown with options: High, Medium, Low)
- Include a submit button with the value "Create Task" and id as "createTask"

success.jsp:

If the task creation is successful, display a success message: **"Task Created Successfully".**

error.jsp:

If there is an error during task creation (e.g., empty fields), display an error message: **"Failed to create task. Please fill in all the required fields."**

Task.java:

Implement a **Task** class to represent task data with the following private fields:

- title
- description
- dueDate
- priority

Include appropriate constructors, getters, and setters.

CreateTaskServlet.java:

Implement a servlet to handle task creation.

- Retrieve parameters (title, description, dueDate, priority) from the request.
- Check if all parameters are not null and not empty.
- If the parameters are valid, create a new Task object, set it as a request attribute, and redirect the user to the success page **{proxy_uri}/success.jsp**.
- If parameters are invalid, redirect the user to the error page **{proxy_uri}/error.jsp**.

Note

- **Environment Variable Retrieval:** The servlet accesses an environment variable named "VSCODE_PROXY_URI" using System.getenv() method. This variable contains the URI of a proxy server, which is crucial for redirecting users to appropriate JSP pages.
- **Redirecting Users:** Depending on the authentication result, the servlet redirects users to either a success or error JSP page. The redirection URLs are constructed based on the proxy URI retrieved from the environment variable.

Spring JPA:

Description

Implement the JPQL statement to fetch the list of curtains whose price is greater than the given price and the list of curtains whose brand is equal to the given brand name.

The Curtain Model is as follows:

id : the unique id of the curtain (int)

brand : the brand name of the curtain (String)

material : the material used in curtain (String)

color : the color of the curtain (String)

price : the price of the curtain (int)

File in which you need to write the code:

- src/main/java/com/example/curtainmodel/repository/CurtainRepository.java

The task is to write the JPQL statement in the **CurtainRepository file under the repository folder**:

- **Task 1:** Create a function called "**getIdByPrice**" that returns a list of curtains with a price greater than the given price. You need to use query annotation to execute the function based on the GET query.
- **Task 2:** Create a " **getIdByBrand** " function that returns the list of curtains with a brand equal to the given brand name. You need to use query annotation to execute the function based on the GET query.

You can use the MySQL shell to view the data and query the tables.

SpringBoot Full Stack:

Description

Implement REST APIs to streamline and automate the process of managing shipments for businesses involved in shipping goods. The API should enable users to retrieve a particular

shipment information based on the specific **trackNo** and also delete the requested shipment data based on the requested **shipId**.

Entity Details

The Shipment entity should have the following attributes:

- **shipId** : the id of the Shipment (Integer) (unique)
- **trackNo** : the track number of the Shipment (String)
- **origin** : the origin of the Shipment (String)
- **destination** : the destination of the Shipment (String)
- **status** : the status of the Shipment (String)

Here is an example of a Shipment JSON object:

```
{
    "shipId": 1,
    "trackNo": "TRK001",
    "origin": "New York",
    "destination": "Los Angeles",
    "status": "In Transit"
}
```

You are provided with the implementation of the models required for all the APIs. Your task is to implement a set of REST services that exposes the endpoints, retrieves a particular shipment information based on the specific **trackNo** and delete the requested shipment data based on the requested **shipId:**

API Route	API Type	Success Response Code	Validation Error Code
/shipment/{trackNo}	GET	200	404
/shipment/{shipId}	DELETE	200	404

Task 1:

GET request to /shipment/{trackNo}

Retrieve shipment details by trackNo.

Response Body: JSON object representing the shipment details.

HTTP Status Code:

- 200 (OK) - If Shipment with the specified trackNo found and returned.
- 404 (Not Found) - If Shipment with the specified trackNo not found.

Task 2:

DELETE request to /shipment/{shipId}

Delete shipment details by shipId.

Response Body: String object should give the response as "The requested shipId-3 got deleted" for correct shipId and response

HTTP Status Code:

- 200 (OK) - Shipment Id, i.e., the shipId found and deleted.
- 404 (Not Found) - Shipment with the specified shipId not found.

Task 3:

ShipmentService

- Implement the GET method which should return a Shipment data based on trackNo.
- Implement the DELETE method which should delete a Shipment data based on shipId.

Complete the given project so that it passes all the test cases when running the provided unit tests.

Files in which you need to write the code:

- src/main/java/com/example/shipment_model/controller/ShipmentController.java
- src/main/java/com/example/shipment_model/service/ShipmentService.java

Example Requests and Responses:

GET request to /shipment/{trackNo}:

The response code is 200 and the response body, when converted to JSON, is as follows:

Request: GET - /shipment/TRK002

```
{
    "shipId": 2,
    "trackNo": "TRK002",
    "origin": "London",
    "destination": "Paris",
    "status": "Pending"
}
```


DELETE request to /shipment/{shipId}

The response code is 200 and the response body should return a message as follows:

Request: DELETE - /shipment/3

The requested shipId-3 got deleted

You can use the MySQL shell to view the data and query the tables.