**Nome: \_\_\_\_\_\_\_\_\_\_\_ Data: \_\_\_\_\_\_\_\_\_\_\_**

*Responda em português ou inglês.*

1. **Write a Java program to solve the following problem:**

**You are tasked with creating a utility function for a text-processing application. The function must generate all possible anagrams from a given group of distinct letters. For example, the input {a, b, c} should return the output: abc, acb, bac, bca, cab, cba.**

**Additional Requirements:**

1. **The program should accept any group of distinct letters as input and produce the correct result.**
2. **Optimize for readability and clarity.**
3. **Include basic validation (e.g., ensure the input is non-empty and contains only letters).**
4. **Write unit tests to validate your function with at least three different test cases, including edge cases (e.g., input with a single letter or empty input).**
5. **Document your code clearly, explaining the logic for generating anagrams.**

1. **Provide an example scenario where overriding the equals() method is necessary in Java. Explain the key considerations when implementing this method, such as ensuring it aligns with the hashCode() method. Include code examples if possible.**

**R = A substituição do metodo equals é necessario quando deseja compara seu conteudo e não a referencia de memória ex:**

**Se voce tem a clase carro onde tem 2 variaveis marca e cor o equal vai comaprar referencia de memória e duas instancia diferentes podem ser consideradas iguais**

**import java.util.Objects;**

**public class Carro {**

**private String marca;**

**private String cor;**

**public Carro(String nome, String cpf) {**

**this.marca = marca;**

**this.cor = cor;**

**}**

**@Override**

**public boolean equals(Object obj) {**

**if (this == obj) {**

**return true;**

**}**

**if (obj == null || getClass() != obj.getClass()) {**

**return false;**

**}**

**Carro carro = (carro) obj;**

**return Objects.equals(marca, carro.marca) && Objects.equals(cor, carro.cor);**

**}**

**@Override**

**public int hashCode() {**

**return Objects.hash(marca, cor);**

**}**

**public static void main(String[] args) {**

**Carro carro1 = new Carro ("Honda", "branco");**

**Carro carro 2 = new Carro ("Honda", "branco");**

**System.out.println(carro 1.equals(carro 2));**

**System.out.println(carro 1.hashCode() == carro 2.hashCode());**

**}**

**}**

1. **Explain how you would use a design pattern to decouple your code from a third-party library that might be replaced in the future. Describe the advantages and limitations of your chosen approach, and provide a small code snippet illustrating its application.**

**R = Utilizaria um padrão de adpter junto com o principio de inversao do SOLID para abstrações assim tendo mais flexibilidade melhor manutenção , desacoplamento e mais facil testar.**

**Porem teriamos sobrecarga de Código aumentaria a complexidade entre outras coisas por isso tudo deve ser bem analisado pros e contras para ver se vale realmente a pena a implementação.**

**public interface EmailService {**

**void sendEmail(String recipient, String subject, String body);**

**}**

**public class EmailNotificationService {**

**private final EmailService emailService;**

**public EmailNotificationService(EmailService emailService) {**

**this.emailService = emailService;**

**}**

**public void notifyUser(String userEmail) {**

**emailService.sendEmail(userEmail, "Welcome!", "Thank you for signing up!");**

**}**

**public static void main(String[] args) {**

**EmailService emailService = new LibraryXEmailAdapter();**

**EmailNotificationService notificationService = new EmailNotificationService(emailService);**

**notificationService.notifyUser("user@example.com");**

**}**

**}**

1. **Describe your experience with Angular, including its core features and use cases. Provide an example of a practical application where you used Angular and include a code snippet demonstrating a key feature, such as component communication, data binding, or service integration.**
2. **Discuss the techniques you use to prevent SQL injection attacks in web applications. Provide examples of code showing secure implementations, such as using parameterized queries or ORMs. Mention any additional measures you take to secure the database layer.**
3. **Describe the steps you would take to diagnose and improve the performance of a batch process that interacts with a database and an FTP server. Explain how you would identify bottlenecks, optimize database queries, improve logic execution, and enhance file transfer efficiency. Provide examples of tools or techniques you would use during the analysis.**

|  |  |
| --- | --- |
| **Salesperson** | **Customer** |
| |  |  |  |  | | --- | --- | --- | --- | | **ID** | **Name** | **Age** | **Salary** | | 1 | Abe | 61 | 140000 | | 2 | Bob | 34 | 44000 | | 5 | Chris | 34 | 40000 | | 7 | Dan | 41 | 52000 | | 8 | Ken | 57 | 115000 | | 11 | Joe | 38 | 38000 | | |  |  |  |  | | --- | --- | --- | --- | | **ID** | **Name** | **City** | **Industry Type** | | 4 | Samsonic | Pleasant | J | | 6 | Panasung | Oaktown | J | | 7 | Samony | Jackson | B | | 9 | Orange | Jackson | B | |

|  |
| --- |
| **Orders** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **ID** | **order\_date** | **customer\_id** | **salesperson\_id** | **Amount** | | 10 | 8/2/96 | 4 | 2 | 540 | | 20 | 1/30/99 | 4 | 8 | 1800 | | 30 | 7/14/95 | 9 | 1 | 460 | | 40 | 1/29/98 | 7 | 2 | 2400 | | 50 | 2/3/98 | 6 | 7 | 600 | | 60 | 3/2/98 | 6 | 7 | 720 | | 70 | 5/6/98 | 9 | 7 | 150 | |

1. **Given the tables above, write the SQL query that:   
     
   a. Returns the names of all Salesperson that don’t have any order with Samsonic.**

**b. Updates the names of Salesperson that have 2 or more orders. It’s necessary to add an ‘\*’ in the end of the name.**

**c. Deletes all Ssalesperson that placed orders to the city of Jackson.**

**d. The total sales amount for each Salesperson. If the salesperson hasn’t sold anything, show zero.**

1. **The customer has a system called XYZ and intends to start updates split into 3 phases. The requirements for the first phase are as follows:**
2. **Enable new data entries in the system, which will serve as input for the second phase.**
3. **Implement functionality to create, update, delete, and search plants.**
   * **Plants should have the following attributes:**
     + **Code: Numeric only, mandatory, and unique.**
     + **Description: Alphanumeric, up to 10 characters, optional.**
   * **Only admin users can delete plants.**
4. **Ensure that the system prevents duplication of plant codes.**

**Task:  
Based on the above information:**

1. **Write a use case or user story for this scenario, ensuring that it clearly addresses the requirements.**
2. **Highlight any business rules or assumptions relevant to the solution.**
3. **Describe any validations or security measures you would implement in the system.**
4. **Suggest how you would test this functionality, including examples of edge cases.**
5. **Consider the following description of a system functionality:  
   User Registration**

* **A screen allows users to insert, delete, or update user information.**
* **Each user has properties: name, email, address, and phone, where name and email are mandatory fields.**
* **Emails must be unique across all users.**
* **Only admin users can delete other users.**

**Task:**

1. **Describe the types of tests you would implement (e.g., unit, integration, or end-to-end tests) and explain the scenarios you would test to ensure the functionality works as expected.**
2. **Provide examples of edge cases and how you would handle them.**
3. **Include an example of a test case in code or pseudocode for one or more scenarios.**