ITE-002 - TÓPICOS ESPECIAIS - II

PROF. ALEXANDRE GARCIA

1. Lista 2

Exercício 1. *Escreva um método que receba um Object, e que retorne uma String genérica de Drop Table.*

Exercício 2. *O método getModifiers() das classes Field e Method, nos retorna um inteiro que representa seu modificador. O método static Modifier.toString() mostra todos os modificadores presentes em um membro da classe. Faça um método que mostre na tela todos os membros e seus modificadores.*

Exercício 3. *O Método getInterfaces() da classe Class nos retorna um vetor de Class contendo todas as interfaces implementadas por uma dada classe. Implemente um método que liste todas as interfaces de um objeto passado via parâmetro.*

Exercício 4. *Faça um método que liste todos os métodos protected e static de uma dada classe.*

Exercício 5. *Crie um método que receba um object e mostre na tela todos os métodos protected e public, e todos os atributos default, CASO este for instância de AbstractList.*

Exercício 6.

(1) *Crie uma Annotation Positivo;*
(2) *Faça um método validador, para atributos inteiros, este deve válidar se o número é positivo;*
(3) *Faça um teste que cheque se o atributo é válido ou não, mostrando na tela os dois casos;*

Exercício 7.

(1) *Crie uma Annotation Senha;*

(2) *Faça um método validador para senhas, esta String deve possuir no mínimo um caractér numérico;*
(3) *Faça um método que cheque se a senha é válida ou não, mostrando na tela os dois casos;*

Exercício 8. *Faça um método que receba um object e coloque todos os atributos inteiros com o valor -89.*

Exercício 9. *Faça um método que retorne o nome do tipo do atributo 5 de uma classe qualquer caso exista.*

Exercício 10. *Escreva um método que receba um Object, e que retorne uma String genérica de Select da forma: "SELECT* $atributo_1$*, ..,* $atributo_N$ *FROM tabela WHERE* $atributo_0$ *= ?"*

Exercício 11. *Escreva um método que receba um Object...(Vários Objects), e que retorne uma String genérica de Select da forma: "SELECT * FROM* $tabela_1$ *AS T1,* $tabela_2$ *AS T2 , ... ,* $tabela_N$ *AS TN WHERE* $T1.chave_1 = T2.chave_1$ *AND ... AND* $T(N-1).chave_{n-1} = TN.chave_{n-1}$*". Para a string acima ocorrer é necessário que as tabelas a frente tenham o atributo chave estrangeira da anterior. Exemplo, em T2 deve ter a chave estrangeira de T1. Use uma annotation para chaves estrangeiras para facilitar sua vida.*

Exercício 12. *Obtenha uma instância da classe Secret abaixo, mude seu atributo e mostre-o na tela. Liste seus métodos, quais você obteve? Inovoque-os via invoke e descubra para que servem, se existir.*

```
public class A {

    private class Secret{
        private int superSecretInt=9000;
    }

}
```

Exercício 13. *Crie dois métodos, um chamado soma que recebe dois inteiros e retorna sua soma, e outro chamado dobro que recebe um inteiro e retorna seu dobro numa classe chamada Calculadora. Feito isso crie um método que receba uma Calculadora e um int...(Pode receber*

vários inteiros), e mostre(println) via ***reflection*** *o retorno dos métodos acima na tela. Use o método invoke da classe Method.*

Exercício 14. (1) *Crie uma Annotation MuitosParaMuitos que tenha como atributo um Class chamado tabela();*

(2) *Faça um método validar que verifica se há a Annotation em questão em duas classes, onde uma referencia a outra pelo atributo de chave primária. Ex: Deseja criar Cliente = id × nome; Produto = idProd × nomeProd; CliProd = id × idProd, então nas classes Cliente e Venda os atributos id e idProd deverão estão marcadas pelo @MuitosParaMuitos(tabela = Venda.class) e @MuitosParaMuitos(tabela = Cliente.class) respectivamente;*

(3) *Faça um método que gere String de create table genérica para uma tabela resolução de uma relação muitos-para-muitos;*

Exercício 15. *Faça um método que receba um Object e execute o construtor da instância em questão que possua parâmetros String e Integer. Você deve instanciar um objeto Constructor, e pelo método getDeclaredConstructor() passando via parâmetro um vetor de Class indicando os parâmetros do construtor para o mesmo ser localizado. Para executar o construtor chame o método newInstance() com os parâmetros na ordem certa.*

Exercício 16. *Faça um método que mostre na tela uma superclasse de um objeto passado via parâmetro.*

Exercício 17. *Faça um método que liste todos os métodos privados e uma classe. Execute todos que não possuem parâmetro via reflection. Para os que possuem parâmtero liste nome e tipo.*

Exercício 18. *Para cada atributo double e int faça uma String SQL genérica que execute todas as funções de grupo conhecidas. Para executar tal função passe um parâmetro inteiro que determine esta execução, ou seja, 1- Média, 2- MAX, 3- MIN, 4- COUNT, etc... Guarde esses valores em um ArrayList e retorne-o.*

Exercício 19. *Faça uma annotation OUT que marca atributos que não devem ser criados via create table. Faça uma string genérica de create table que não inclua os atributos marcados com OUT.*

Exercício 20. *Faça uma String genérica de upadate, onde os campos a serem alterados devem ser passados via parâmetro, se não existir*

tal atributos mostre uma mensagem de erro. Caso o atributo tenha a annoation OUT marcada, um erro deverá ser mostrado.

Exercício 21. *Idem acima, só que faça com o Delete.*

Exercício 22. *Faça uma annonatation com um atributo inteiro. Faça um método que lista os atributos de uma classes marcadaos com está annotation, e os mostre na tela de acordo com o número. Você deverá mostrar em ordem crescente.*

Exercício 23. *Faça uma annotation para marcar uma classe. Crie um método que receba um ArrayList de Object e mostre na tela as classes marcadas com esta annotation.*

Exercício 24. *Escreva um método que receba um Object, e que retorne uma String genérica de Select da forma: "SELECT* $atributo_1$*, ..,* $atributo_N$ *FROM tabela WHERE* $atributo_0$ *= ?"*

Exercício 25. *Montando um banco de dados para um jogo de RPG, percebeu-se a necessidade de criar validações para um cadastro de armas. Uma arma é válida se esta possui nome não-nulo, um preço de venda maior que 0 e um bônus de ataque maior que 0. Considerando que a classe Arma com os atributos (nome, preço e bonus) já esteja criada, crie um método (Na classe Validacao) para validar armas. Use reflection e annotations.*