1) Tendo uma planilha com 1000000 itens em uma planilha do excel no seguinte formato

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Nome | email |  |
| 1 | Joaquim | joaquim@teste |  |
| 2 | Manoel | manoel@teste |  |
| 3 | Maria | maria@teste |  |

Pretendo criar uma quarta coluna que ‘agrupe’ as linhas em grupos de 100, ou seja, a cada 100 itens, um nome diferente. Que solução você usaria?

2) Quando e por que usar o ForkJoin ou o ExecutorService?

3) Considere o cenário: Uma loja precisa fazer um controle de estoque. Ela vende itens de vestuário e eletrônicos mas pretende expandir no futuro suas atividades. Visando construir um sistema, o estagiário alimenta a planilha seguinte:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Produto | Descrição | Categoria | Quantidade | Fornecedor | Cliente | Observação |
| Camisa | Camisa manga longa verde | Vestuario | 10 | Fabrica das camisas |  | Nota fiscal 88 |
| Calça | Calça Jeans preta | Vestuario | 10 | Fabrica das Camisas |  | Nota fiscal 88 |
| Sapato | Sapatos de couro artezanal | Vestuario | 6 | San Marino |  | Nota fiscal 77 |
| Sapato | Sapatos de couro artezanal | Vestuario | 1 |  | Joaquim | Nota fiscal 0001 |
| Televisor | TV 20 polegadas com função netflix | Eletrodomesticos | 30 | Casas bahia |  | Nota fiscal a2212a |

Como ficaria a modelagem de dados observando as boas práticas de normalização?

4) Encontre os erros e aponte melhorias nos códigos Java a seguir:

a)

**public** String recuperarApenasNumeros(String s) {

StringBuilder sb = **new** StringBuilder();

**for** (**int** i=0; i<= s.length(); i++) {

**if** (s.charAt(i)>'0' && s.charAt(i) < '9') {

sb.append(s.charAt(i));

}

}

**return** sb.toString();

}

b)

public class Teste {

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

List<Animal> lst = Arrays.*asList*(**new** Cachorro("Rex"),

**new** Gato("Bichano"),

**new** Cachorro("Rintintim"),

**new** Gato("Garfield"),

**new** Cachorro("Spike"));

**for** (Animal item: lst) {

**if** (item **instanceof** Cachorro) {

System.***out***.println( item.getNome() +

" é um " +((Cachorro)item).getEspecie() );

}

**if** (item **instanceof** Gato) {

System.***out***.println( item.getNome() +

" é um " +((Gato)item).getEspecie() );

}

}

}

}

**class** Animal {

String nome;

**public** String getNome() {

**return** nome;

}

}

**class** Cachorro **extends** Animal{

Cachorro(String nome){

**this**.nome = nome;

}

**public** String getEspecie() {

**return** "Caninus";

}

}

**class** Gato **extends** Animal {

Gato(String nome){

**this**.nome = nome;

}

**public** String getEspecie() {

**return** "Felinus";

}

}

5) Complete o Javascript seguinte

function teste(){

let l = [];

for (var i=0; i< 10000; i++){

l.push({id: i, nome: "Item "+i, endereco:”Rua tal número”+i});

}

//TODO IMPRIMIR APENAS NÚMEROS DE 201 e 315

console.log(l);

//TODO criar uma variável contendo uma lista contendo apenas Nome e Id

//TODO criar uma variável contendo uma lista contendo apenas Endereço e Id

}

6)Tenho a seguinte interface

<html>

<head>  
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>  
</head>

<body>

<div class=”erros”></div>

<form>

<label for="nome">Nome</label> <input type="text" name="nome" id="nome" class="meuinput">

<br/>

<label for="telefone">Telefone</label> <input type="text" name="telefone" id="telefone" class="meuinput">

<br/>

<label for="cpf">CPF</label> <input type="text" name="cpf" id="cpf" class="meuinput">

<br/>

<button onclick="validar()">

Validar

</button>

</form>

</body>

</html>

Construa uma função que:

\* Marque os campos inválidos de vermelho. Regras: O nome precisa ter ‘nome e sobrenome’; O telefone deve ter no mínimo 8 dígitos e no máximo 9 e não pode conter letras; O CPF não pode conter letras.

\* Empilhe na div “mensagem” os erros encontrados.

7) Existe um novo sistema pra fazer. Este sistema é um sistema onde a maior parte é composta de CRUDs e relatórios. Proponha uma arquitetura possível.

8) Existe um novo sistema pra fazer. Este sistema é um sistema crítico de alta disponibilidade e precisa atender no mínimo 5000 requisições por segundo. Proponha uma arquitetura possível.