Curso Estruturas de Dados e Algoritmos Expert
Prof. Dr. Nelio Alves
Capítulo: Conjuntos e dicionários

DEVSUPERIOR
https://devsuperior.com.br

DESAFIO: Banco de dados sem ORM

Forma de entrega: link do programa salvo no Gist do Github
Linguagens aceitas: Javascript, Java, C#, Python

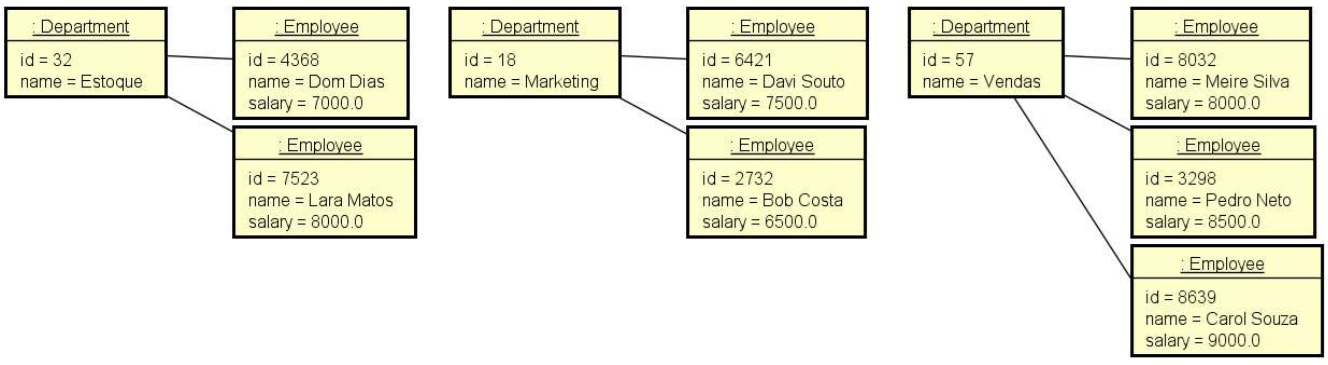
Você foi contratado para trabalhar em um sistema legado que não utiliza ferramenta de ORM (mapeamento objeto-relacional) para acessar o banco de dados, ou seja, a tradução dos dados entre o formato relacional (tabelas) para objetos em memória, deve ser feita programaticamente pelos desenvolvedores.

Assim, quando uma consulta é feita ao banco de dados, os dados são retornado em forma tabular, e esses dados precisam ser convertidos para objetos em memória. Por exemplo, os dados abaixo são os registros resultantes de uma consulta ao banco de dados para retornar os departamentos e seus respectivos funcionários:

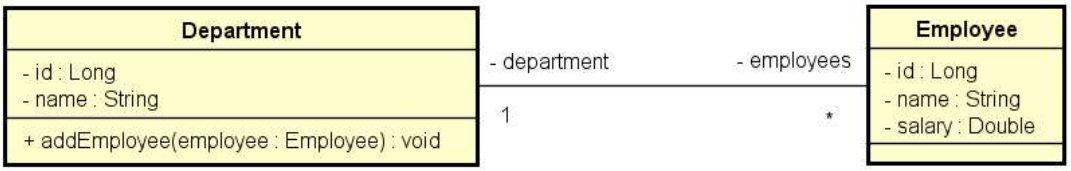
dept_id	dept_name	employee_id	employee_name	salary	department_id
57	Vendas	8032	Meire Silva	80000,0	57
32	Estoque	4368	Dom Dias	70000,0	32
57	Vendas	3298	Pedro Neto	85000,0	57
57	Vendas	8639	Carol Souza	90000,0	57
18	Marketing	6421	Davi Souto	75000,0	18
32	Estoque	7523	Lara Matos	80000,0	32
18	Marketing	2732	Bob Costa	65000,0	18

Você deve criar uma função **convertRecords** (veja assinaturas ao final deste documento). Esta função deve receber um array dos registros acima no formato CSV, e deve retornar uma lista de objetos do tipo **Department**, ordenados por nome, e associados com seus respectivos funcionários (objetos tipo **Employee**) na ordem em que aparecem no array original.

Para o exemplo acima, os objetos em memória devem ficar conforme mostrado abaixo. Repare que cada funcionário (objeto do tipo **Employee**) deve ter uma referência para seu departamento (objeto do tipo **Department**), e cada departamento deve ter um array/lista de referências para seus funcionários.



A seguir está o projeto do modelo de domínio Department-Employee para te auxiliar na implementação dessas classes. Repare que há um método addEmployee na classe Department, para associar um novo funcionário ao departamento:



Atenção: não pode haver instanciação repetida de departamentos. Você deve manter os departamentos já instanciados em um **dicionário** pois, durante o processamento, se surgir um novo funcionário de um mesmo departamento já instanciado, este novo funcionário deverá apontar para o mesmo departamento. Assim, o dicionário vai permitir a recuperação eficiente dos departamentos já instanciados.

Você deve criar a função **convertRecords** já mencionada, e você deve criar um programa para navegar na lista resultante da função **convertRecords**, mostrando um relatório na saída, conforme exemplo abaixo.

Exemplo 1:

Entrada

```
[
  "57,Vendas,8032,Meire Silva,8000.0,57",
  "32,Estoque,4368,Dom Dias,7000.0,32",
  "57,Vendas,3298,Pedro Neto,8500.0,57",
  "57,Vendas,8639,Carol Souza,9000.0,57",
  "18,Marketing,6421,Davi Souto,7500.0,18",
  "32,Estoque,7523,Lara Matos,8000.0,32",
  "18,Marketing,2732,Bob Costa,6500.0,18"
]
```

Saída

```
Estoque:
  4368: Dom Dias, $ 7000.00
  7523: Lara Matos, $ 8000.00
Marketing:
  6421: Davi Souto, $ 7500.00
  2732: Bob Costa, $ 6500.00
Vendas:
  8032: Meire Silva, $ 8000.00
  3298: Pedro Neto, $ 8500.00
  8639: Carol Souza, $ 9000.00
```

Exemplo 2:

Entrada

```
[
  "57,Vendas,8032,Meire Silva,8000.0,57",
  "18,Marketing,6421,Davi Souto,7500.0,18",
  "18,Marketing,2732,Bob Costa,6500.0,18"
]
```

Saída

```
Marketing:
  6421: Davi Souto, $ 7500.00
  2732: Bob Costa, $ 6500.00
Vendas:
  8032: Meire Silva, $ 8000.00
```

Assinaturas:

Javascript:
```
function convertRecords(records)
```

Java:
```
static List<Department> convertRecords(String[] records)
```

C#:
```
static List<Department> ConvertRecords(string[] records)
```

Python:
```
def convert_records(records)
```