Lista de Exercício 5 – Java Tiger e IO

1. Construa um programa que leia um arquivo texto no formato A (posicional) e exporte as mesmas informações no formato B (cvs – dados separados por ponto e vírgula), Script SQL, XML 1, XML 2, Objeto e JSON. Segue detalhamento dos formatos:

Formato A:

Número do Banco: 5 caracteres

Número da Agência: 6 caracteres

CPF do cliente: 11 caracteres

Saldo da Conta: 18 caracteres, sendo os dois últimos o valor dos centavos

Formato B:

CPF;Banco;Agência;Saldo

Script SQL:

insert into TB001\_MOVIMENTACAO (CPF,COD\_BANCO,COD\_AGENCIA,SALDO) values (<cpf>,<numero banco>,<numero da agencia>,<saldo>);

Exemplo:

insert into TB001\_MOVIMENTACAO (CPF,COD\_BANCO,COD\_AGENCIA,SALDO) values (‘81646674987’,’104’,’14600’,12578.34);

Onde os valores dos dados deverão substituir as tags <>, e cada linha do arquivo A deverá gerar uma instrução SQL de insert.

XML 1:

<?xml version=”1.0” encoding=”UTF-8”?>

<contas>

<conta>

<cpf>81646674987</cpf>

<banco>1</banco>

<agencia>14600</agencia>

<saldo>15780.13</saldo>

</conta>

...

</contas>

XML2:

<?xml version=”1.0” encoding=”UTF-8”?>

<contas>

<conta cpf=”1646674987” banco=”1” agencia=”14600” saldo=”1.10”/>

<conta cpf=”7646674987” banco=”2” agencia=”13400” saldo=”8.50”/>

...

</contas>

Objeto:

Cada linha deverá ser populado num objeto equivalente e a lista destes objetos deverá ser serializada num arquivo.

JSON:

{ “propriedade”: “valor”, “propriedade”: { “propriedade”: “valor” }}

{

“contas”: [

{ “cpf”: “81646674987”, “banco”: “001”, “agencia”: “12121”, “saldo”: 3123123 },

{ “cpf”: “81646674987”, “banco”: “001”, “agencia”: “12121”, “saldo”: 3123123 },

{ “cpf”: “81646674987”, “banco”: “001”, “agencia”: “12121”, “saldo”: 3123123 }

]

}

1. Construa um programa Java que leia o arquivo de objetos serializado do exercício 1 e seja capaz de exportar para os outros formatos.
2. Adapte o exercício para uma solução genérica capaz de trabalhar com qualquer objeto. Considere que as posições do dado de cada campo no arquivo original possa ser definido através de anotação no campo com atributos inicio e fim, conforme exemplo de uso abaixo:

@Substring(inicio=11, fim=22

private String cpf;

1. Crie uma anotação @ToString que deve ser anotado nos campos de uma classe. O método toString() deverá detectar os campos anotados com a mesma e utilizar os valores destes campos para compor uma representação String do objeto da seguinte forma: [campo: valor, ...]
2. Evolua a solução do exercício 4 para adicionar uma anotação @Format que possui um atributo formatter do tipo Class<Formatador>. Formatador é uma interface que recebe um objeto qualquer e retorna uma representação String do mesmo. Crie implementações desta interface para formatação de datas, horas, numéricos, etc. Utilize generics para definira interface Formatador.