REVISÃO PARA PROVA 2

Nome:_____

1. [2 pontos] Transforme a Classe **Produto** mostrada em uma classe genérica, onde o código do produto pode ser um inteiro, ou uma String ou, eventualmente, um objeto da classe **Codigo**. Além disso, o preço pode ser double ou inteiro.

```
public class Produto {
    private int codigo;
    private String descricao;
    private double preco;
    public Produto(int codigo, String descricao, double preco){
        this.codigo = codigo;
        this.descricao = descricao;
        this.preco = preco;
    }
    public int getCodigo() {return codigo;}
    public String getDescricao() {return descricao;}
    public double getPreco() {return preco;}
    @Override
    public String toString() {
        return "Produto{" + "codigo=" + codigo + ", descricao=" +
descricao + ", preco=" + preco + '}';
    }
}
```

```
public class Produto<T,U> {
    private T codigo;
    private String descricao;
    private U preco;
    public Produto(T codigo, String descricao, U preco){
        this.codigo = codigo;
        this.descricao = descricao;
        this.preco = preco;
    }
    public T getCodigo() {return codigo;}
    public String getDescricao() {return descricao;}
    public U getPreco() {return preco;}
    @Override
    public String toString() {...}
}
```

2. [2 pontos] Implemente o seguinte método estático:

```
public static List<Integer> intercala(List<Integer> arr1,
List<Integer> arr2);
```

   - Este método retorna uma nova lista com a intercalação de **arr1** e **arr2**. Por exemplo, se arr1 = [1,2,3] e arr2 = [7,8,9], então a função deve retornar a lista [1,7,2,8,3,9]. Note que ambas ***as listas têm o mesmo tamanho***.

```
public static List<Integer> intercala(List<Integer> arr1, List<Integer> arr2) {
    List<Integer> arr12 = new ArrayList<Integer>();
    for (Integer i = 0; i < arr1.size(); i++) {
        arr12.add(arr1.get(i));
        arr12.add(arr2.get(i));
    }
    return arr12;
}
```

3. [2 pontos] Um programa mantém um dicionário de sinônimos em uma estrutura Map:

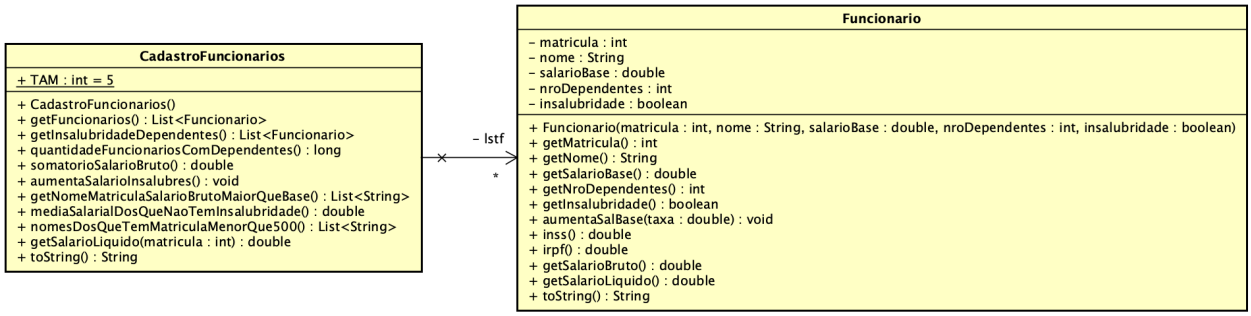
   - `private Map<String,List<String>> dicSinonimos;`

   Ou seja, cada palavra (uma string) tem uma lista de sinônimos (strings). Escreva o método estático abaixo que retorna *`true`* se ***sinonimo*** está entre os sinônimos de ***palavra***, ou *`false`* caso contrário.

   - `public static boolean ehSinonimo(String palavra, String sinonimo)`

```
public static boolean ehSinonimo(String palavra, String sinonimo){
  for (String l : dicSinonimos.get(palavra)){
    if (l.equals(sinonimo))
      return true;
  }
  return false;
}
```

4. [4 pontos] A partir das classes **Funcionario** e **CadastroFuncionarios** mostradas abaixo, implemente os métodos indicados **FAZENDO USO** de ***streams*** e ***funções lambda***:



a. `public long quantidadeFuncionariosComDependentes()`

   - Esse método deve retornar a quantidade de funcionários que tem dependentes

```
public long quantidadeFuncionariosComDependentes(){
  return lstf
    .stream()
    .filter(f->(f.getNroDependentes() > 0))
    .count();
}
```

b. `public List<String> getNomeMatriculaSalarioBrutoMaiorQueBase()`

   - Esse método deve retornar uma lista de strings com nome e a matricula de todos os funcionários cujo salário bruto é mais de 10% maior que o salário base

```
public List<String> getNomeMatriculaSalarioBrutoMaiorQueBase(){
  return lstf
    .stream()
    .filter(f->(f.getSalarioBruto() > f.getSalarioBase()*1.1))
    .map(f->f.getNome()+", "+f.getMatricula())
    .collect(Collectors.toList());
}
```