	COLÉGIO PEDRO II – *CAMPUS* DUQUE DE CAXIAS DATA: ___/___/_____ SÉRIE/TURMA: **DS102** DISCIPLINA: **Linguagem de Programação 2** PROFESSOR: **Judismar Arpini Júnior** NOME: Nº	NOTA
	SIMULADO 1ª CERTIFICAÇÃO – PROVA INDIVIDUAL – SEM CONSULTA – VALOR TOTAL: 5	

BOA PROVA!

Questão 1) [1,5]

Considere as seguintes afirmações referentes à Programação Orientada a Objetos. Julgue-as como verdadeiras (**V**) ou falsas (**F**).

( ) É possível executar um método de uma classe sem instanciá-la. [0,3]

( ) Classes com propriedades **static** não podem ser instanciadas. [0,3]

( ) Os métodos **públicos** são as ações que um objeto pode realizar. [0,3]

( ) Para uma classe herdar de outra, usamos a palavra reservada **extends**. [0,3]

( ) A palavra reservada **readonly** define atributos constantes: uma vez inicializados, não podem ser alterados. [0,3]

Questão 2) [2,0]

(i) Escreva, em TypeScript, a classe **encapsulada** BichoDeEstimacao, a qual deve ter um atributo **nome.** Deve ter também um método do tipo get que retorna o nome, e outro método do tipo set que altera o nome a partir de outro nome passado como argumento. [1,0]

(ii) Escreva agora, em TypeScript, a classe **encapsulada** Pessoa, a qual possui os atributos **nome** e **bichos**, sendo este último um array (lista) de BichoDeEstimacao, inicializando-o em sua definição como uma lista vazia. Deve ter também um método que retorna o **nome** (método tipo get), um método que insere um BichoDeEstimacao no array **bichos**, um método que retorna um bicho a partir de um **índice** (parâmetro do método) e um construtor que inicializa apenas o nome. [1,0]

Questão 3) [1,0]

Preencha as lacunas do código TypeScript abaixo, de maneira que o atributo **nome** seja inicializado no construtor e que seja diferente da string vazia.

```
class Funcionario
{
        private nome: _______;

        public __________ (nome: string)
        {
                this.nome = nome;
                if(nome ____ “”)
                {
                        _______________ = “Rex”;
                }
        }
}
```

Questão 4) [0,5]

Sobre **sobrescrever** métodos e **herança** em TypeScript, marque a alternativa correta:

(A) TypeScript não permite herança.

(B) É possível sobrescrever métodos da superclasse em alguma subclasse, quando esta herda por meio da palavra chave **inherits.**

(C) Podemos sobrescrever um método herdado uma **única** vez numa **mesma** subclasse.

(D) Quando várias classes herdam um método de uma superclasse, **apenas uma** delas pode sobrescrever o método.