	Universidade Federal de Santa Maria Centro de Tecnologia Departamento de Processamento de Energia Elétrica Prof. Renan Duarte	Nota

DPEE1090 – Programação orientada a objetos para automação
Avaliação parcial 1

Observações:
- Para cada questão devem ser fornecidos como resposta os entregáveis especificados no enunciado
- Os códigos devem ser comentados para facilitar a interpretação

Nome: _____ Data: _____

1. [Entregável: Arquivo da classe *Retangulo*] Considere a classe abaixo que modela um retângulo com base nos atributos de *Largura* e *Altura*. Modifique a classe de modo que apenas retângulos com lados de dimensão maior que 0 e menor que 100 possam ser criados. Garanta também que mesmo após criados, os objetos *Retangulo* não possam ser modificados de forma a violar o critério acima.

```
namespace Prova
{
    internal class Retangulo
    {
        public double Largura;
        public double Altura;

        public Retangulo () { }

        public Retangulo(double largura, double altura)
        {
            Largura = largura;
            Altura = altura;
        }

        public double Area()
        {
            return Largura * Altura;
        }

    }
}
```

---

2. [Entregável: Arquivos da classe *ProdutoEletronico* e programa principal *Program*] Considere um sistema de uma loja online que vende produtos eletrônicos. Crie uma classe chamada *ProdutoEletronico* para modelar os produtos eletrônicos disponíveis na loja. Cada produto eletrônico deve ter os seguintes atributos:
    - *Nome*: o nome do produto
    - *Marca*: a marca do produto
    - *Preço*: o preço do produto – Deve **sempre** ser maior ou igual a zero
    - *Estoque*: a quantidade em estoque do produto – Deve **sempre** ser maior ou igual a zero

    Além disso, a classe deve conter:

    - Um construtor para inicializar todos os atributos do produto
    - Um construtor para inicializar apenas o nome e a marca do produto. Nesse caso, os demais atributos devem ser inicializados com valor zero

- *Getters* e *Setters* para o preço do produto
- Um método *AdicionarAoEstoque* que permite adicionar uma determinada quantidade ao estoque do produto
- Um método *Vender* que permite vender uma quantidade específica do produto, atualizando o estoque adequadamente
- Um método *ValorEmEstoque* que retorna o valor total em estoque do produto (*Preco * Estoque*)
- Um método *ToString* que retorna o produto em formato de string da seguinte maneira:

"Nome (Marca) - $12.34 cada, 56 unidades em estoque - $691.04 em estoque"

---

3. [Entregável: Arquivos da classe *ConversorTemperatura* e programa principal *Program*] Na automação industrial, umas das variáveis mais medidas e controladas é a temperatura. Crie uma classe chamada *ConversorTemperatura* que permite converter temperaturas nas escalas Celsius, Kelvin e Fahrenheit. A classe deve possuir um único método público *Converter*, mas que opera com as seguintes sobrecargas:
   - *Converter*: recebe uma temperatura e a unidade de medida original, bem como a unidade de medida desejada, e retorna a temperatura convertida para a nova unidade.
   - *Converter*: sobrecarga do método anterior que recebe apenas a temperatura original em Celsius e retorna a temperatura convertida para a nova unidade.

   Para definir as unidades de medida, utilize 'C' para Celsius, 'F' para 'Fahrenheit' e 'K' para Kelvin. Exemplo:

```
// Conversão de 25 Fahrenheit para Kelvin
Converter(25, 'F', 'K');

// Conversão de 25 Celsius para Fahrenheit
Converter(25, 'F');

// Conversão de 25 Celsius para Kelvin
Converter(25, 'K');
```

   A conversão de temperatura entre as diferentes escalas é feita da seguinte forma:

De/Para	Celsius	Fahrenheit	Kelvin
Celsius	1	$\frac{9}{5}C + 32$	$C + 273{,}16$
Fahrenheit	$\frac{5}{9}(F - 32)$	1	$\frac{5}{9}(F - 32) + 273{,}16$
Kelvin	$K - 273{,}16$	$\frac{9}{5}(K - 273{,}16) + 32$	1