Instituto Federal de Educação Ciência e Tecnologia do RN
Curso Técnico Integrado em Informática para Internet
Programação Estruturada e Orientada a Objetos

Lista de Exercícios nº 01

1. O que define a assinatura de um método?

2. O que é um construtor padrão? Como os atributos de um objeto são inicializados se uma classe possui apenas um construtor padrão definido implicitamente?

3. O que são os membros de um objeto?

4. Qual o objetivo dos atributos de um objeto?

5. Explique porque em uma classe devemos, preferencialmente, definir os atributos com visibilidade privada.

6. Explique o princípio do encapsulamento e seus principais benefícios.

7. Qual a diferença entre objeto e classe?

8. Por que podemos dizer que uma classe serve como um “molde” de suas instâncias?

9. O que constitui o estado de um objeto? Objetos de mesma classe possuem estados iguais?

10. O que define o comportamento de um objeto?

11. O que define a interface de um objeto?

12. Qual o objetivo dos construtores?

13. Implemente uma classe `Circulo` que define os métodos `Area()` e `Perimetro()`.Também defina a propriedade `raio` com permissão de leitura. No construtor, forneça o raio do círculo.

14. Escreva uma classe `PopulacaoBaratas` que simule o crescimento de uma população de baratas, sendo que a quantidade inicial da população é fornecida ao construtor. O método `Aguardar()` simula um período no qual a população dobra. O método `Pulverizar()` simula a pulverização com inseticida, que reduz a população em 10%. A propriedade `baratas` (somente leitura) deve informar o número atual de baratas. Também implemente um programa de testes que simule uma cozinha que inicia com dez baratas. Aguarde, pulverize e imprima a contagem de baratas. Repita três vezes.

15. Implemente uma classe `Carro` a partir das seguintes informações. Um carro tem certa taxa de consumo (medida em km/l) e certa quantidade de combustível no tanque. A taxa de consumo é especificada no construtor e o nível de combustível inicial é zero. Forneça um método `Dirigir` que simula dirigir o carro por determinada distância, reduzindo o nível de combustível no tanque. Crie um método (ou propriedade de leitura) que retorna o nível de combustível atual. Também escreva o método `AdicionarCombustivel` para colocar combustível. O código a seguir é um exemplo de uso de um objeto `Carro`. Escreva um programa de teste (método `Main`) para testar sua implementação.

```
Carro meuFusca = new Carro(7); //7 quilômetros por litro
meuFusca.AdicionarCombustivel(20); //abastece com 20 litros
meuFusca.Dirigir(100); //dirigir por 100 km
Console.WriteLine(meuFusca.combustivel);
```

16. Crie uma classe chamada `ContaBancaria`. Esta classe deve incluir uma propriedade (somente leitura) do tipo `double` para representar o saldo da conta. A classe deve prover um construtor que recebe o saldo inicial da conta. O construtor deve verificar se este valor é maior ou igual a zero. Em caso contrário, o construtor deve inicializar o atributo com zero e exibir uma mensagem de erro no console indicando que o valor foi inválido. A classe deve conter três métodos. O método `Credito` deve adicionar uma quantia informada como parâmetro ao saldo corrente. O método `Debito` deve retirar uma quantia (informada como parâmetro) em dinheiro da conta caso a quantia a ser retirada não seja superior ao saldo da conta. Neste caso, o saldo deve ser deixado intacto e a função deve imprimir uma mensagem indicando que a quantia a retirar é superior ao saldo. Implemente um programa que cria duas instâncias da classe `ContaBancaria` e testa a implementação da classe.

17. Crie uma classe chamada `Fatura` que será utilizada para representar a fatura de um item vendido por uma loja de eletrodomésticos. Uma `Fatura` deve incluir quatro propriedades: número (`string`), descrição (`string`), quantidade do item sendo vendido (`int`) e o preço do item (`float`). A classe deve possuir um construtor que inicializa os quatro atributos. Todas as propriedades devem possuir permissão de leitura e escrita. Adicionalmente, crie um método chamado `Total` que calcula o total da fatura (i.e., multiplica a quantidade pelo preço do item) e então retorna o valor encontrado. Se o preço do item não for positivo, então ele deve ser definido como igual a zero. Escreva um programa que demonstra as capacidades da classe `Fatura`.

18. Implemente uma classe chamada `Empregado` que inclui três partes de informação como propriedades: primeiro nome, último nome e salário mensal. Esta classe deve prover um construtor que inicializa as três propriedades. As três propriedades devem permitir leitura e escrita. Se o salário mensal não for positivo, ajuste seu valor para zero. Escreva um programa que cria duas instâncias de `Empregado` e exibe o salário anual de cada objeto. Em seguida o programa deve dar um aumento de 10% para cada objeto e exibir novamente o salário anual de cada empregado. Faça um desenho indicando o estado final de cada objeto.

19. Considerando o código a seguir, faça um desenho mostrando o estado de cada objeto ao final do método `Main`.

```
class A{
  private int i;
  private String id;
  public A(String id){
    i = 0;
    this.id = id;
  }
  public A(){
    i = 0;
    id = "sem id";
  }
  public void Incrementar(){ i++; }
  public void SetId(String new_id){ id = new_id; }
  public int GetId(){ return i; }
}

class B{
  private double valor;
  public int ordem;
  public B(double v){
    valor = v;
    ordem = 1;
  }
  public double GetValor(){ return valor; }
  public void SetValor(double v){ valor = v; }
```

```
  static void Main(string[] args){
    A a1 = new A();
    A a2 = new A("ky");
    B b1 = new B(25.3);
    B b2 = new B(19.0);
    for(int i = 0; i < 5; i++){
      a1.Incrementar();
      if(i % 2 == 0){
        a2.Incrementar();
      }
    }
    b1.SetValor(a1.GetId());
    a2.SetId("BG");
    b1.SetValor(b1.GetValor() + 32.0);
    b2.ordem = a2.GetId();
  }
}
```