Centro Universitário

Estácio | FIC

Exercícios

1) [SOBRECARGA] Crie uma classe chamada Gerente para definir os objetos que representarão os gerentes do banco. Defina dois métodos de aumento salarial nessa classe. O primeiro deve aumentar o salário com uma taxa fixa de 10%. O segundo deve aumentar o salário com uma taxa variável. Teste os métodos de aumento salarial definidos na classe Gerente.
2) [GERAL] Crie uma classe que represente as contas do banco. Essa classe deve conter três atributos: numero, limite e saldo. Crie uma classe chamada TestaConta. Dentro dessa classe, crie um objeto do tipo Conta. Receba do teclado os valores para os atributos numero, saldo e limite. Depois crie um laço que permita que o usuário escolha a operação que ele deseja realizar. As operações que ele pode realizar são: depositar, sacar e imprimir extrato.
3) [GERAL] Crie uma classe que represente os funcionários do banco. Essa classe deve conter dois atributos: nome e salario. No domínio do banco, obrigatoriamente, os funcionários devem possuir um salário inicial de R$200,00. Crie uma classe chamada TestaFuncionario. Dentro dessa classe, crie um objeto do tipo Funcionario. Receba do teclado o valor para o atributo nome. Depois crie um laço que permita que o usuário possa alterar o nome e o salário dos funcionários e também visualizar os dados atuais.
4) [ARRAYS] Crie um programa que imprima na tela os argumentos passados na linha de comando para o método Main.
5) [ARRAYS] Faça um programa que ordene o array de strings recebido na linha de comando.
6) [ARRAYS] Faça um programa que calcule a média dos elementos recebidos na linha de comando. Dica: para converter strings para double utilize o método ToDouble().
7) [ARRAYS] Crie um programa que encontre o maior número entre os valores passados na linha de comando.
8) [HERANÇA] Defina uma classe para modelar os funcionários do banco. Sabendo que todo funcionário possui nome e salário, inclua as propriedades dos atributos. Crie uma classe para cada tipo específico de funcionário herdando da classe Funcionario. Considere apenas três tipos específicos de funcionários: gerentes, telefonistas e secretarias. Os gerentes possuem um nome de usuário e uma senha para acessar o sistema do banco. As telefonistas possuem um código de estação de trabalho. As secretarias possuem um número de ramal. Teste o funcionamento dos três tipos de funcionários criando um objeto de cada uma das classes: Gerente, Telefonista e Secretaria. Implemente um método para calcular uma bonificação para todos os funcionários (padrão 10%). Reescreva o cálculo da bonificação do gerente para 15%.

9) Crie uma classe Animal, com três métodos (Acordar(), Comer(), Dormir()) e uma propriedade (Nome);
10) Crie uma classe Mamifero, que herda de Animal (criado como no item acima) e que possui o método Mamar(). Crie, ainda, as classes Morcego (que possui o método Voar()) e Baleia (que possui o método Nadar()), derivadas de Mamifero;
11) Crie uma enumeração Sexo (masculino e f, com valores 1 e 10, respectivamente). Após isso, crie uma classe abstrata Pessoa que possui os métodos Acordar(), Comer() (esse é abstract) e Dormir() e as propriedades int Codigo, String Nome e Sexo Sexo. Crie duas classes (Homem e Mulher) que herdam de Pessoa;
12) Crie uma classe Calculadora que faça as quatro operações básicas (soma, subtração, multiplicação e divisão). Crie uma classe derivada CalculadoraCientifica que, além das operações básicas, faça extração de raízes;
13) Crie uma classe abstract Figura, com um campo double x (privado). Crie um construtor para fornecer o valor para esse campo. Crie dois métodos abstract CalcularArea() e CalcularPerimetro() que (nas derivadas) retornarão os valores da área e do perímetro de uma figura regular, baseados na medida x. Crie as classes derivadas Equilatero, Quadrado, Circulo, Pentagono, Hexagono, Heptagono e Octogono. Sobrescreva o método ToString() de Figura.
14) O que significam os termos abaixo (alguns não existem!!! identifique-os):
    a. public (usado em tipos);
    b. internal (usado em tipos);
    c. protected (usado em tipos);
    d. protected internal (usado em tipos);
    e. private (usado em tipos);
    f. abstract (usado em tipos);
    g. static (usado em tipos);
    h. out (usado em parâmetro do tipo reference em métodos);
    i. out (usado em parâmetro do tipo value em métodos);
    j. ref (usado em parâmetro do tipo reference em métodos);
    k. ref (usado em parâmetro do tipo value em métodos);
    l. private (usado num construtor);
    m. private (usado na declaração de um membro de uma interface).
15) Responda:
    a. O que é OOP?
    b. O que é polimorfismo?
    c. O que é abstração?
    d. O que é encapsulamento?
    e. Quando devo usar uma classe abstrata e quando devo usar uma interface?
    f. Existe herança múltipla (de classes) em C#?
    g. O que é boxing e unboxing?
16) Na programação orientada a objetos, há um mecanismo que permite definir modificadores de acesso. Quando se define um atributo de uma classe com o modificador de acesso privado, significa que:
    a. o acesso à classe é privado.
    b. o atributo é acessível a um programa que tenha uma referência a um objeto da classe.
    c. a classe é abstrata.

d. o atributo é acessível somente aos métodos da classe.

17) Em POO (Programação Orientada a Objetos), dizer que a classe A estende a classe B é o mesmo que dizer que:
    a. a classe B é subclasse de A;
    b. a classe A é superclasse de B;
    c. a classe A é derivada de B;
    d. a classe B é derivada de A;
    e. as classes A e B são irmãs.
18) Em POO (programação orientada a objetos), dizer que a classe A
    a. é superclasse de B é o mesmo que dizer que:
    b. A é derivada de B;
    c. A estende B;
    d. B é derivada de A;
    e. B implementa A;
    f. A implementa B.
19) Uma das características da programação orientada a objetos está relacionada com a proteção dos atributos internos dos objetos contra modificações diretas. As alterações dos atributos devem ocorrer por meio de métodos adequados, criados para acesso e modificação desses atributos. Essa característica é conhecida como
    a. encapsulamento.
    b. herança.
    c. generalização.
    d. polimorfismo.
    e. sobrecarga de operador.
20) A herança um princípio de orientação a objetos que permite que classes compartilhem atributos e métodos é utilizada para reaproveitar código ou comportamento generalizado ou especializar operações ou atributos.
    a. Certo
    b. Errado