**Programação Orientada a Objetos**

**Prof. Bernardo Copstein**

**Herança**

1. Uma locadora de veículos trabalha com veículos de passeio e veículos de carga. Sobre cada veículo é necessário armazenar o número do chassi, a placa e o consumo (em Km/l). Sobre veículos de passeio é necessário armazenar também o número máximo de passageiros e sobre veículos de carga é necessário armazenar a capacidade de carga (em toneladas). Para calcular o combustível necessário para uma certa distância (em Km), basicamente basta dividir a distância pelo consumo. No caso dos veículos de passeio, porém, acima de dois passageiros o consumo aumenta 2% por passageiro adicional, enquanto que no caso dos veículos de carga o consumo cresce 0,5% por tonelada acima de 3 toneladas. Modele e implemente uma hierarquia de classes capaz de representar esta situação. Escreva também um exemplo de uso destas 3 classes explorando o princípio da substituição.
2. Identifique a superclasse e a subclasse em cada um dos seguintes pares de classes:
   1. Empregado, gerente
   2. Estudante de graduação, estudante
   3. Pessoa, estudante
   4. Empregado, professor
   5. ContaBancaria, ContaComChequeEspecial
   6. Carro, Veiculo
   7. Veiculo, Minivan
   8. Carro, Minivan
   9. Caminhão, Veiculo
3. Considere que a classe A é superclasse das classes B e C e que a classe B é superclasse das classes D e E. Considere ainda que todas as classes sobrescrevem o método *toString*. Em cada uma, o método toString retorna o nome da classe concatenado com o da superclasse. Analise o trecho de código abaixo e indique o que será impresso na tela após a execução do mesmo:

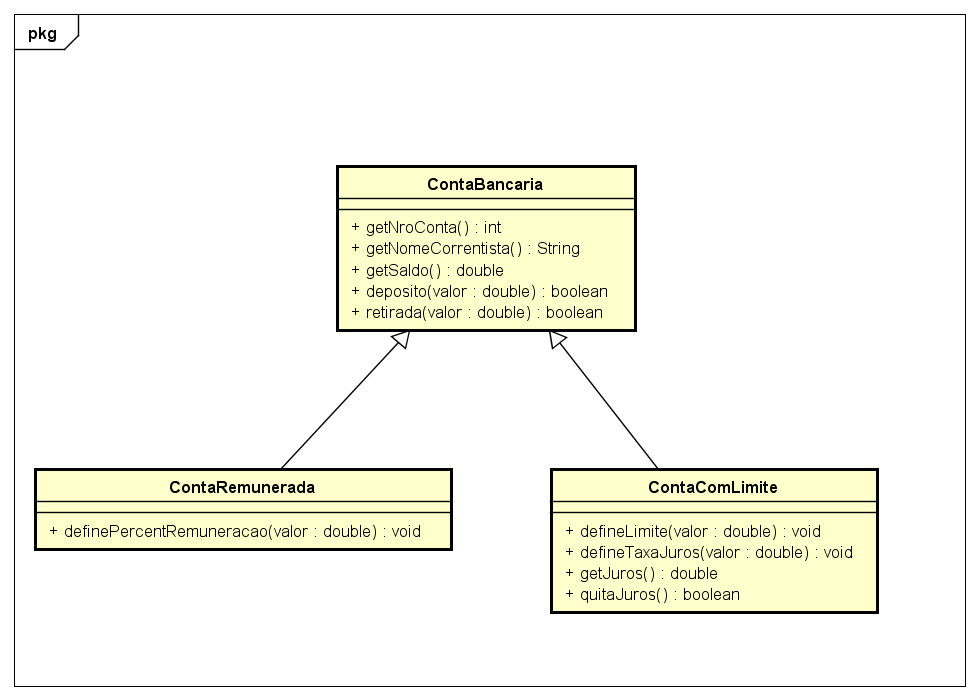
|  |
| --- |
| A a = new E();  E e = new E();  B b = new E();  A a2 = new B();  System.out.println(a+”,”+e+”,”+b+”,”+a2); |

1. Analise o código da classe *Funcionario* que segue e crie as seguintes subclasses:
   1. *Servente*: classe derivada da classe *Funcionario*. Um servente recebe um adicional de 5% a título de insalubridade
   2. *Motorista*: classe derivada da classe *Funcionario*. Para cada motorista é necessário armazenar o número da carteira de motorista.
   3. MestreDeObras: classe derivada da classe *Servente*. Para cada mestre de obras é necessário armazenar quantos funcionarios estão sob sua supervisão. Um mestre de obras ganha um adicional de 10% para cada grupo de 10 funcionários que estão sob seu comando.

Em todas as classes devem ser acrescentados os métodos *get*/*set* necessários. O método *toString* deve ser sobrecarregado quando for conveniente.

|  |
| --- |
| public class Funcionario {  private String nome;  private int codigo;  private double salarioBase;  public Funcionario(int codigo, String nome, double salarioBase) {  this.codigo = codigo;  this.nome = nome;  this.salarioBase = salarioBase;  }  public String getNome() { return nome; }  public int getCodigo() { return codigo; }  public double getSalarioBase(){ return(salarioBase);}  public void setSalarioBase(double salarioBase){  this.salarioBase = salarioBase;  }  public double getSalarioLiquido(){  double inss = salarioBase \* 0.1; double ir = 0.0;  if (salarioBase > 2000.0){  ir = (salarioBase-2000.0)\*0.12;  }  return(salarioBase - inss - ir );  }  public String toString() {  return (getClass().getSimpleName() + "\n Codigo: "+  getCodigo()+ "\n Nome: " + getNome() +  "\n Salario Base: "+getSalarioBase()+  "\n Salario liquido: "+getSalarioLiquido());  }  } |

1. Escreva um exemplo de uso que crie pelo menos uma instância de cada tipo de funcionário criado no exercício 2. O exemplo deve criar as instancias e exibir os dados armazenados por cada uma delas.
2. Desenhe o diagrama de classes correspondente as classes desenvolvidas no exercício 2.
3. Uma fábrica de embalagens para bebidas armazena sobre cada modelo de embalagem que fabrica seu código de referência e custo unitário. O volume de líquido que cada uma pode armazenar varia conforme o modelo: embalagens do tipo caixa (paralelepípedo) tem seu volume calculado multiplicando-se a área da base (lado1 \* lado2) pela altura; as embalagens cilíndricas usam a mesma lógica, só que a área da base é a área de um círculo (PI \* raio ^2); as embalagens cônicas usam a seguinte fórmula: (Pi\*raio^2\*altura)/3 e as que correspondem a um tronco de cone usam a formula (1/3 \* Pi \* altura \* (raio\_da\_base^2 + raio\_da\_base \* raio\_do\_tipo + raio\_do\_topo^2)). Implemente uma hierarquia de classes que modele esta situação. Defina um método “toString” capaz de retorna uma string com o código do modelo, o modelo de embalagem (**this**.getClass().getName()), o custo unitário e o volume que armazena. Faça um exemplo de uso.
4. Analise a figura que segue e implemente a hierarquia de classes definida na mesma:
   1. A “conta bancaria” implementa as funcionalidades básicas de uma conta corrente.
   2. A “conta remunerada” remunera cada depósito usando o percentual definido. O crédito é feito no momento do depósito.
   3. A “conta com limite” permite que sejam feitos saques a descoberto até o limite informado. Os juros são calculados no momento da retirada que ultrapassa o saldo. O método “getJuros” informa o montante devido de juros até o momento. O método “quitaJuros” zera o valor devido em juros debitando os mesmos da conta desde que haja saldo para tanto.



1. Recupere o código da hierarquia de questões apresentada em aula. Acrescente classes para armazenar questões de escolha múltipla e questões de resposta numérica.