1. Crie uma classe para representar uma conta corrente, com métodos para depositar uma quantia, sacar uma quantia e obter o saldo. Para cada saque será debitada também uma taxa de operação equivalente à 0,5% do valor sacado. Crie, em seguida, uma subclasse desta classe anterior para representar uma conta corrente de um cliente especial. Clientes especiais pagam taxas de operação de apenas 0,1% do valor sacado. Faça testes com as duas classes e verifique seus resultados.

2. Crie uma hierarquia de classes de domínio para uma loja que venda livros, CDs e DVDs. Sobrescreva o método toString() para que imprima:

   Para livros: nome, preço e autor;

   Para CDs: nome, preço e número de faixas;

   Para DVDs: nome, preço e duração.

   Evite ao máximo repetição de código utilizando a palavra super no construtor e no método sobrescrito. Em seguida, crie uma classe Loja com o método main() que adicione 5 produtos diferentes (a sua escolha) a um vetor e, por fim, imprima o conteúdo do vetor.

3. Modifique o código do programa anterior, da seguinte forma:

   a) Adicione um atributo que represente o código de barras do produto (é um valor obrigatório e, portanto, deve ser pedido no construtor);

   b) Sobrescreva o método equals() retornando true se dois produtos possuem o mesmo código de barras;

   c) Na classe Loja, implemente um simples procedimento de busca que, dado um produto e um vetor de produtos, indique em que posição do vetor se encontra o produto especificado ou imprima que o mesmo não foi encontrado;

   d) No método Loja.main(), após a impressão do vetor (feita na questão 2a), escolha um dos 5 produtos e crie duas novas instâncias idênticas a ele: uma com o mesmo código de barras e outra com o código diferente. Efetue a busca deste produto no vetor utilizando as duas instâncias e verifique o resultado.

4. Ainda modificando o código do programa anterior, faça com que Produto implemente a interface Comparable, e implemente a comparação por nome. Ao final do método Loja.main(), ordene o vetor utilizando o método java.util.Arrays.sort() e imprima-o novamente. Depois altere a implementação da comparação para ordenar por preço e verifique o resultado.

5. Crie a seguinte hierarquia de classes:

   Uma interface para representar qualquer forma geométrica, definindo métodos para cálculo do perímetro e cálculo da área da forma;

   Uma classe abstrata para representar quadriláteros. Seu construtor deve receber os tamanhos dos 4 lados e o método de cálculo do perímetro já pode ser implementado;

   Classes para representar retângulos e quadrados. A primeira deve receber o tamanho da base e da altura no construtor, enquanto a segunda deve receber apenas o tamanho do lado;

   Uma classe para representar um círculo. Seu construtor deve receber o tamanho do raio.

   No programa principal, pergunte ao usuário quantas formas ele deseja criar. Em seguida, para cada forma, pergunte se deseja criar um quadrado, um retângulos ou um círculo, solicitando os dados necessários para criar a forma. Todas as formas criadas devem ser armazenadas em um vetor. Finalmente, imprima: (a) os dados (lados ou raio); (b) os perímetros; e (c) as áreas de

todas as formas. Para (b) e (c), tire vantagem do polimorfismo, enquanto que para (a) utilize instanceof e downcast.

6. Implemente a classe Funcionario e a classe Gerente.

   crie a classe Assistente, que também é um funcionário, e que possui um número de matrícula (faça o método GET). Sobrescreva o método exibeDados().

   sabendo que os Assistentes Técnicos possuem um bônus salarial e que os Assistentes Administrativos possuem um turno (dia ou noite) e um adicional noturno, crie as classes Tecnico e Administrativo.

7. Criar uma estrutura hierárquica que contenha as seguintes classes:

   Veiculo (classe abstracta), Bicicleta e Automóvel.
   Os métodos da classe Veiculo são todos abstractos e possuem a seguinte assinatura:

   • listarVerificacoes() • ajustar()
   • limpar()

   Estes métodos são implementados nas subclasses Automóvel e Bicicleta.

   Acrescentar na classe Automóvel o método mudarOleo()

8. Crie uma classe chamada Ingresso que possui um valor em reais e um método imprimeValor().

   crie uma classe VIP, que herda Ingresso e possui um valor adicional. Crie um método que retorne o valor do ingresso VIP (com o adicional incluído).

   crie uma classe Normal, que herda Ingresso e possui um método que imprime: "Ingresso Normal".

   crie uma classe CamaroteInferior (que possui a localização do ingresso e métodos para acessar e imprimir esta localização) e uma classe CamaroteSuperior, que é mais cara (possui valor adicional). Esta última possui um método para retornar o valor do ingresso. Ambas as classes herdam a classe VIP.

9. Crie a classe Imovel, que possui um endereço e um preço.

   crie uma classe Novo, que herda Imovel e possui um adicional no preço. Crie

   métodos de acesso e impressão deste valor adicional.

   crie uma classe Velho, que herda Imovel e possui um desconto no preço. Crie métodos de acesso e impressão para este desconto.

10. Crie uma classe de Teste com o método *main*. Neste método:

    crie um assistente administrativo e um técnico. Imprima o número de matrícula e o nome de cada um deles.

    crie um animal do tipo cachorro e faça-o latir. Crie um gato e faça-o miar. Faça os dois animais caminharem.

    teste (como quiser) as classes Rica, Pobre e Miseravel.

    crie um ingresso. Peça para o usuário digitar 1 para normal e 2 para VIP. Conforme a escolha do usuário, diga se o ingresso é do tipo normal ou VIP. Se for VIP, peça para ele digitar 1 para

camarote superior e 2 para camarote inferior. Conforme a escolha do usuário, diga se que o VIP é camarote superior ou inferior. Imprima o valor do ingresso.

crie um imóvel. Peça para o usuário digitar 1 para novo e 2 para velho. Conforme a definição do usuário, imprima o valor final do imóvel.