**Programação Orientada a Objetos, turma 128**

**Prof. Bernardo Copstein**

**Polimorfismo**

**Lista de exercícios**

Em todas as questões abaixo explore ao máximo as características de herança e polimorfismo da linguagem Java.

1. Uma loja de departamentos vende vários tipos de produtos. A classe *Produto* (ver abaixo) corresponde ao topo da hierarquia de classes que modela os produtos vendidos pela loja. No caso dos eletrodomésticos é necessário armazenar um “boolean” que indica se o aparelho tem voltagem 220v ou não. No caso de eletropesados a margem de lucro reduz para 20% caso o produto custe mais de R$ 999,00. Por fim, os produtos de telefonia tem o imposto aumentado em 50% caso sejam importados. Considerando este contexto, faça o que se pede:
   1. Implemente uma hierarquia de classes capaz de modelar os diferentes tipos de produtos vendidos pela loja;
   2. Escreva um programa Java que armazena em um *ArrayList<Produto>* produtos dos mais diversos tipos. Em seguida percorra a lista imprimindo a descrição do produto e o correspondente valor para o consumidor final.

|  |
| --- |
| **public** **class** Produto {  **private** **int** codigo;  **private** String descricao;  **private** **double** preco;    **public** Produto(**int** codigo, String descricao, **double** preco) {  **super**();  **this**.codigo = codigo;  **this**.descricao = descricao;  **this**.preco = preco;  }  **public** **double** getPreco() { **return** preco; }  **public** **void** setPreco(**double** preco) { **this**.preco = preco; }  **public** **int** getCodigo() { **return** codigo; }  **public** String getDescricao() { **return** descricao; }    **public** **double** getImposto() { **return** preco\*0.1; }    **public** **double** getMargemLucro() { **return** preco\*0.4; }    **public** **double** calculaPrecoConsumidor() {  **return** getPreco()+getImposto()+getMargemLucro();  }  } |

1. Implemente a aplicação de conta bancária descrita nos slides/capítulo 9 do livro. Veja o diagrama de classes abaixo.

|  |
| --- |
|  |

1. Escreva uma classe chamada “Catalogo” que armazena as diferentes embalagens produzidas por uma fábrica. Sobre cada embalagem armazena-se o nome do cliente, o modelo de embalagem (ver questão 2 da lista de exercícios sobre herança) e o tempo de produção (em dias) para o lote de 10000 unidades. A classe deve ter métodos para:
   1. Cadastrar uma embalagem no catálogo;
   2. Imprimir o catálogo (contendo nome do cliente, modelo e volume de armazenamento para cada embalagem cadastrada).
   3. Verificar se uma bagagem é “frágil”. Este método recebe uma bagagem por parâmetro e retorna “true” quando se trata de bagagem frágil e “false” caso contrário. OBS: este método pode ser “static”?
2. Implemente uma classe chamada “Prova” cujas questões podem ser dos tipos exemplificados em aula. A classe deve ter métodos para:
   1. Cadastrar questões (a primeira questão inserida corresponde a primeira questão da prova, a segunda inserida a segunda questão da prova e assim sucessivamente);
   2. Recuperar uma questão (pelo número da questão);
   3. Remover uma questão (pelo número da questão);
   4. toString (exibindo todas as questões numeradas e em ordem).
3. Implemente uma classe chamada “AplicacaoDeProva”. Esta deve armazenar uma referência para uma prova além de uma resposta para cada uma das questões da prova. A classe deve ter métodos para:
   1. Definir a prova
   2. Cadastrar uma resposta (informa a string com a resposta e o número da questão)
   3. Retornar a nota prova (questões textuais ou numéricas valem 2 pontos, questões de escolha simples 1 ponto e questões de escolha múltipla 1,5 pontos).
   4. Imprimir um relatório indicando para cada questão
      1. O texto da questão
      2. A resposta informada
      3. Indicação de que a resposta está correta ou indicação de que está incorreta seguida da resposta correta.
      4. Indicar a nota no final do relatório

**Para os próximos exercícios leia o quadro abaixo:**

|  |
| --- |
| Entendendo a classe “Object”: em Java todas as classes são derivadas da classe “Object”, ou seja, “Object” é superclasse de todas as classes de Java por definição. A classe “Object” define alguns métodos importantes que são herdados por todas as classes e que podem ser sobrescritos na herança conforme a necessidade. Os mais usados são:   * “toString”: a versão original retorna o nome da classe seguido do valor da referência para a instância corrente. * “equals”: compara a instância corrente com a recebida por parâmetro. Na versão original compara o conteúdo de todos os atributos da classe, só retornando “true” se todos os atributos forem iguais. * “clone”: retorna um cópia do objeto. A versão original lança uma “CloneNotSuportedException”. * “getClass”: retorna um descritor da classe da instância corrente (classe “Class”). * “hashCode”: devolve um código hash para o objeto. |

1. Considere a classe “Funcionario” da lista de exercícios sobre herança. Reescreva a mesma de maneira que dois funcionários sejam o considerados “o mesmo” (equals) apenas se os CPF forem os mesmos. Escreva um exemplo de uso.
2. Altere a classe “Catalogo” do exercício 3 de maneira que o método “clone” correspondente seja capaz de retornar uma cópia do catálogo de embalagens. Discuta a relação do método clone com o encapsulamento de uma classe.
3. Escreva um método chamado “kitDeFinalDeAno”. Este método recebe uma instância de funcionário por parâmetro e imprime na tela o nome do funcionário e seu tempo de serviço (se for administrativo) e os brindes que irá receber. Todos os funcionários recebem uma cadeira de praia e um guarda sol. Funcionários administrativos com mais de 10 anos de casa recebem também uma “geladeira de praia para cerveja” e os diretores, além da cadeira e do guarda-sol, recebem óculos escuros. O operador “instanceof” permite definir se uma instância é de um determinado tipo. Para acessar um método que não pertence a interface da superclasse, use um “cast” para converter a instância para a subclasse. Exemplo:

|  |
| --- |
| **public** **static** **void** teste(Funcionario f){  System.***out***.println(f);  **if** (f **instanceof** Administrativo){  Administrativo a = (Administrativo)f;  System.***out***.println(a.getAnoContratacao());  }  } |

1. Escreva um método que recebe uma lista de funcionários por parâmetro e retorna à quantidade de “geladeiras de praia para cerveja” que devem ser adquiridas para a distribuição de final de ano.
2. Defina e implemente uma classe que modele uma estrutura de dados do tipo “pilha” (LIFO) capaz de armazenar apenas instância de “Object” e suas derivadas. Escreva um exemplo de uso onde se use esta estrutura para armazenar instâncias de “String” explorando inserções e retiradas.