Exame de Paradigmas de Programação

Exame de Época Normal - 29/06/2017

1º Ano da Licenciatura em Engenharia Informática do ISEP

Parte prática (14 valores); Cotações: 1. – 15%; 2. – 40%; 3. – 20%; 4. – 25%

Prova sem consulta; Duração: 2h00

Responda a cada pergunta numa folha separada, identificada com número e nome.
Recomenda-se a leitura integral do enunciado antes de começar a responder.

Pretendem-se classes Java para uma aplicação de apoio à venda de *smartphones* e computadores pessoais. Estes computadores subdividem-se em três categorias: portátil, híbrido e *desktop*. Na venda de um aparelho, o cliente pode optar por incluir um seguro.

Responda às alíneas seguintes tendo sempre em consideração os principais princípios da programação orientada por objetos: abstração, encapsulamento, herança e polimorfismo.

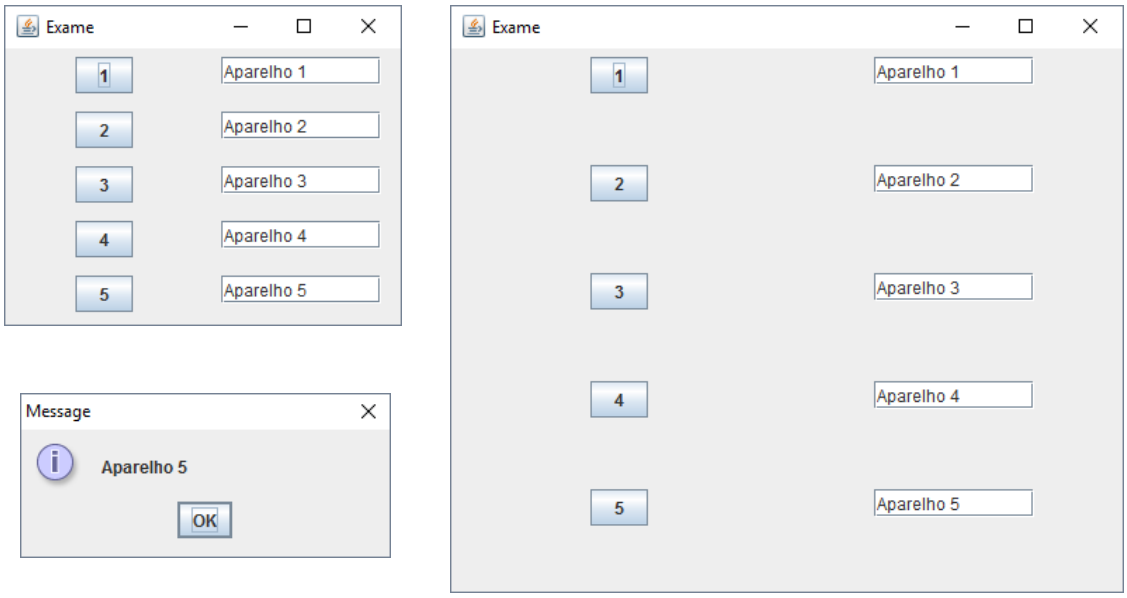
1. Defina a classe `Seguro` para representar seguros, caracterizados por um `identificador` (número inteiro) e pelo `custo`. Assuma existentes os construtores, completo e sem parâmetros, o método reescrito `toString`, os métodos de consulta (`get`) dos atributos da classe e o método de modificação (`set`) do `identificador`. Especifique os atributos da classe e os seguintes membros:
   - Construtor de cópia.
   - Método `set` para modificar o `custo` do seguro. No caso do parâmetro recebido ser inválido, o método deve lançar uma exceção do tipo `IllegalArgumentException` com mensagem apropriada para o utilizador.
   - Método `equals` reescrito. Considere iguais os seguros que têm o mesmo `custo`.

2. Elabore classes representativas dos aparelhos, com e sem seguro, e que satisfaçam os seguintes requisitos:
   - Os computadores são caracterizados pela `marca`, `modelo`, `categoria` (apenas `portátil`, `híbrido` ou `desktop`), `preço base` e `margem de lucro` (sobre o `preço base`). A `margem de lucro` é igual para todos os computadores e pode ser alterada, sendo 20% por omissão.
   - Os *smartphones* são caracterizados pela `marca`, `modelo`, `número de cartões SIM`, `preço base` e `margem de lucro` (sobre o `preço base`). A `margem de lucro` é igual para todos os *smartphones* e pode ser alterada, sendo 30% por omissão.
   - Os computadores e os *smartphones* vendidos com seguro são ainda caracterizados pelo seguro. A classe `Seguro` deve ser usada por composição.
   - Deve ser declarada e aplicada uma interface, designada `Segurado`, para identificar aparelhos com seguro. Esta interface deve definir o método `getSeguro` que devolva a instância da classe `Seguro`.
   - Devem ser disponibilizados métodos para obter o preço de venda dos aparelhos. No caso dos aparelhos vendidos com seguro, o custo deste deve ser incluído no preço de venda.
   - Cada classe deve fornecer um construtor sem parâmetros e o método `toString` reescrito.
   - As classes devem estar preparadas para permitir a ordenação de uma lista de suas instâncias, por ordem crescente do preço de venda, através de código nativo do Java.
   - Em cada classe, considere apenas existentes: construtor completo, método `equals` reescrito, bem como os métodos `get` e `set` dos atributos, à exceção dos métodos `get` e `set` dos atributos usados por composição.

3. Defina a classe `ListaAparelhos` para representar listas de aparelhos. Os aparelhos devem ser armazenados num contentor do tipo `ArrayList`. Especifique os atributos da classe e apenas os seguintes métodos:
   - Método `adicionarAparelho` para adicionar um computador ou *smartphone*, recebido por parâmetro, caso não exista no contentor. Se for adicionado um elemento ao contentor, o método deve retornar `true`, caso contrário deve devolver `false`.
   - Método `obterListaSmartphones` para retornar uma lista dos *smartphones* armazenados no contentor e por ordem decrescente do preço de venda.
   - Método para armazenar o contentor num ficheiro binário, cujo nome é passado por parâmetro. Deve retornar um valor booleano para indicar o sucesso/insucesso da ação executada. Considere a possibilidade de ser gerada uma exceção do tipo `IOException`. No caso de considerar necessário alterar as classes desenvolvidas anteriormente, indique estas modificações.

4. Pretende-se a implementação da janela ilustrada nas figuras abaixo. Após um clique num dos botões de comando, surge a caixa de diálogo `Message` (componente do tipo `JOptionPane`) com o conteúdo do campo de texto respetivo.

   Deve usar gestores de posicionamento para manter a posição relativa entre componentes quando a janela é redimensionada, tal como ilustrado.



   Assim:

   a) Desenhe um esquema com o posicionamento escolhido para os diferentes componentes gráficos da janela. Indique os nomes das variáveis que usará para identificar cada um dos componentes, assim como o gestor de posicionamento usado em cada painel.

   b) Elabore o código de acordo com o esquema desenhado e que **facilite a alteração** da quantidade de botões de comando e de campos de texto.

   Nota:

   A interface `ActionListener` define o método `void actionPerformed(ActionEvent event)`.