Programação Orientada por Objectos

2013/2014

Trabalho de Laboratório – Curso EI

7

Objectivos:

- Classes abstractas e interfaces em Java.

Programas:

Pretende-se criar um programa que represente os diversos tipos de brinquedos de uma loja. Deverá também ser possível efectuar algumas operações com alguns dos brinquedos como, por exemplo, carregar a bateria de um brinquedo a bateria ou obter o número de peças já colocadas para um brinquedo de construção.

Regras de implementação:

- Criar uma aplicação Java de consola.
- Implementar o código necessário e testar no fim de cada nível.
- Não é necessário obter dados do utilizador. Forneça os dados ao nível do código.
- Use as convenções de codificação adoptadas habitualmente para a linguagem Java.
- Na classe do programa não declare nenhum atributo nem crie nenhum método além do *main*.

Implementação:

Nível 1:

- Implemente uma classe abstracta **Brinquedo**. Esta classe deve ter como atributos o nome, a faixa etária a que se destina, o preço e a data do fim da garantia.
- Inclua os construtores e os métodos para obter e alterar o valor dos atributos criados.
- Acrescente um método que verifica se um brinquedo está na garantia.
- Acrescente um outro método, abstracto, à classe **Brinquedo** de forma a obrigar à existência de um método getDescricao nas classes derivadas. Este método deverá retornar uma String com o nome e o tipo de brinquedo.

Nível 2:

- Crie uma classe **BrinquedoConstrucao**, derivada da classe Brinquedo.
- Defina atributos para o número total de peças e para o número de peças colocadas. Acrescente um método "colocarPeca" que incrementa o número de peças colocadas e um método "verificarConstrucao" que verifica se uma construção está concluída.
- Crie uma classe abstracta **BrinquedoAEnergia** que herda da classe Brinquedo. Defina o atributo estado (**ON ou OFF).** Acrescente os métodos "ligar" e "desligar" que alteram o atributo estado.
- Acrescente às classes criadas os respectivos construtores e os métodos que achar necessários para a atribuição e obtenção dos atributos guardados.

Nível 3:

- Crie uma classe **BrinquedoPilhas** que herde da classe **BrinquedoAEnergia**. Defina para esta classe os atributos número de pilhas, tipo de pilhas (ex: "AA" ou "AAA") e carga das pilhas (0 a 100%).
- Crie o método "getTempoAutonomia". Este método determina quanto tempo de autonomia em minutos tem o brinquedo, sabendo que cada 10% de carga da pilha permite uma autonomia de 15 minutos).

- Crie também o método "isDescarregado". Caso a pilha esteja totalmente descarregada (menos do que 10% da carga) este método devolve true.
- Crie uma classe **BrinquedoEletrico** que herde da classe **BrinquedoAEnergia**. Defina para esta classe o atributo voltagem.
- Para as classes definidas, adapte os métodos "ligar" e "desligar", sabendo que para ligar os brinquedos os elétricos basta ligar o botão e que para os brinquedos a Pilhas é preciso verificar antes se têm carga.

Nível 4:

- Crie uma interface **Carregavel** onde se declaram os métodos "carregar", "descarregar" e "isDescarregado". O método "carregar" coloca a carga a 100%, o método "descarregar" descarrega um valor percentual que é passado como argumento (ex: 10%) e o método "isDescarregado" verifica se não existe carga.
- Implemente a interface na classe **BrinquedoPilhas.**
- Crie um *array* de Brinquedos e preencha-o com um brinquedo de construção, dois brinquedos a pilhas e um brinquedo eléctrico.
- Mostre a descrição de todos os brinquedos que estão no *array*.

Nível 5:

- Crie agora uma classe **Alarme** que tem como único atributo o valor da carga da bateria interna (0 a 100%). Esta classe também implementa a interface **Carregavel.**
- Com o passar do tempo torna-se necessário carregar todos os elementos da loja que funcionam a baterias. Para simular esta situação coloque esses elementos num *array* de **Carregavel.** Para este efeito percorra o *array* criado no nível anterior e passe para o *array* de **Carregavel** todos os elementos que forem carregáveis. A seguir acrescente ainda um objecto da classe **Alarme.**
- Faça agora um ciclo com 20 iterações para simular a passagem do tempo que descarrega, a cada passagem, 10% da carga dos elementos que estão no *array* de **Carregável.** No mesmo ciclo, verifique se depois de ter descarregado um elemento ele fica descarregado. Se isso acontecer carregue-o.

Notas:
Ao testar as classes poderá usar o operador *instanceof* para verificar qual a instância de uma classe ou de uma interface. Use ainda o polimorfismo para testar os métodos criados quando se justificar.

Para os identificadores siga as convenções adoptadas normalmente, em particular:

1) A notação camelCase para o nome das variáveis locais e identificadores de atributos e métodos.
2) A notação PascalCase para os nomes das classes.
3) Não utilize o simbolo '_', nem abreviaturas nos identificadores.