MC302 – Programação Orientada a Objetos 03/05/2016
Instituto de Computação – UNICAMP
Profa.: Cecília Mary Fischer Rubira

Lab10: Interfaces

INSTRUÇÕES INICIAIS:

Nota: Os exercícios devem ser executados na ordem apresentada, pois o nível de dificuldade é crescente.

1. Obtenha o arquivo **lab10.zip** (disponível no ensino aberto);
2. Descompacte esse arquivo no seu diretório de trabalho (workspace do Eclipse);
3. Crie um novo projeto Java no Eclipse. Clicar em File > New > Java Project, manter o lugar padrão e escrever como nome do projeto exatamente o nome da pasta descompactada.
4. Ao final do lab, compacte a **pasta raiz do projeto**, em um arquivo com extensão .**zip**, e suba no ensino aberto no seu portfólio*, disponibilizando-o para os formadores. Suba apenas um arquivo. **Outros formatos não serão avaliados, implicando em nota 0 (zero) na atividade.**
5. Suba um arquivo de texto com a resposta para todas as questões conceituais **dentro** da pasta compactada. Serão aceitos os formatos .**txt** e **pdf**. **Coloque seu nome e RA neste arquivo.**

***obs:** criar um item no portfólio, com nome Lab10, com **somente** o arquivo lab10.zip. Não criem pastas no portfólio.*

As questões de 1 a 7 devem ser feitas em sala e entregues até às 19h. A questão 8 pode ser entregue até às 23h59 de quinta-feira. Para isso, **edite** o item atual e substitua o zip com a **atividade completa**.

Pacote *br.unicamp.ic.mc302.exemploSimples*:

1. Abra os arquivos `I1.java`, `C1.java` e `Principal.java`. Estude o código do método main( ) da classe Principal. Que conclusões você consegue tirar desse pequeno trecho de código? Compile os três arquivos e execute a classe Principal. Modifique o método m1( ) da classe C1 para o seguinte:

   ```
   public void m1();
   ```

   Compile novamente o arquivo `C1.java`. Tente explicar o que aconteceu. Antes de prosseguir para o próximo item, desfaça a alteração feita.
2. Crie uma nova classe chamada C2 que também implementa a interface I1. Adicione ao método main( ) da classe Principal código responsável por criar uma variável chamada obj2 do tipo I1 e inicializá-la com um objeto do tipo C2. Adicione também código responsável por chamar o método m1( ) de obj2. Compile as classes C2 e Principal e execute a última.
3. Defina uma nova interface chamada I2. Essa interface deve definir apenas um método, m2( ), que não recebe parâmetros de entrada e retorna void. Modifique a classe C1 para que ela implemente a interface I2. Altere o método main( ) da classe Principal para que crie uma variável chamada obj3 do tipo I2 e a inicialize com um objeto do tipo C1. Adicione código para chamar o método m2( ) de obj3. Compile as classes C1 e Principal e execute a última.
4. Abra os arquivos `I3.java` e `C3.java`. Modifique a interface I3 para que estenda a interface I1. Compile a classe C3. Por que a compilação falha? Corrija o problema e recompile a classe C3. Altere o método main( ) da classe Principal para que crie uma variável obj4 do tipo I3 inicializada com um objeto do tipo C3. Adicione código que chama os método m1( ) e m3( ) de obj4. Compile e execute a classe Principal. Modifique a interface I3 para que estenda também a interface I2. Compile novamente a classe C3. Por que a compilação falha?

Pacote *br.unicamp.ic.mc302.usoInterfaces*:

5. Abra os arquivos `I4.java`, `I5.java`, `C4.java` e `Principal2.java`. Compile-os e execute a classe Principal2. Adicione ao método main( ) da classe Principal2 uma linha que chama o método m5( ) a partir objeto obj. Compile novamente a classe Principal2. Por que ocorreu um erro? Modifique o tipo da variável obj para C4. Recompile o arquivo, execute-o e explique.

Pacote *br.unicamp.ic.mc302.banco*:

6. Compile os arquivos e execute a classe Principal3. As classes RepositorioContasVector e RepositorioContasArray são responsáveis pelo armazenamento das contas e diferem com relação às suas implementações: a primeira é implementada através de um objeto do tipo Vector (um *array* dinâmico) enquanto a segunda usa *arrays* simples. Ambas implementam a interface IRepositorioContas. No método main( ) da classe Principal3, coloque em comentário o trecho de código rotulado "Alternativa 1" e descomente o trecho rotulado "Alternativa 2". Compile e execute a classe Principal3. Explique o que aconteceu.

7. Observe que as classes RepositorioContasVector e RepositorioContasArray herdam de uma classe comum, Repositorio, com o fim de evitar repetição de código. Essa classe define um método concreto atualizar( ) e dois métodos abstratos, inserir( ) e remover( ), que também são definidos pela interface IRepositorioContas. Remova esses dois últimos métodos da classe Repositorio e recompile-a. Estude as declarações das três classes e corrija o erro de compilação sem precisar modificar a relação de **herança** entre elas e evitando repetição de código.

Pacote *br.unicamp.ic.mc302.*`veiculos`:

8. Compile os arquivos e execute a classe Inicial. Crie uma interface chamada Motorizado que define um único método chamado ligar( ), que não recebe parâmetros e retorna void. Modifique as classes Carro e Caminhao para que implementem essa interface (os métodos implementados devem simplesmente imprimir uma mensagem na tela). Implemente um método ligarVeiculos( ) na classe FilaVeiculo que chama o método ligar( ) de cada veículo motorizado na fila.
   Dicas:
   (i) se baseie na implementação do método mostraFila( ) definido na mesma classe;
   (ii) *casting* e uso do operador instanceof podem ser necessários.

   Insira código no método main( ) da classe Inicial para chamar ligarVeiculos( ) a partir de fila. Recompile FilaVeiculo e Inicial e execute a classe Inicial.