

## Lab06: Polimorfismo e Acoplamento Dinâmico

### INSTRUÇÕES INICIAIS:

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

1. Obtenha o arquivo **lab06.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 *Lab06*, com **somente** o arquivo *lab06.zip*. Não criem pastas no portfólio.

### Pacote *br.unicamp.ic.mc302.contaCor*:

1. Abra os arquivos *ContaCor.java* e *ExemploCoercao.java* e compile-os. O compilador acusa alguns erros. Mude o tipo de todas as variáveis da operação *main()* da classe *ExemploCoercao*, de modo que o método seja executado com a sequência *int*, *byte*, *byte* e recompile. O que aconteceu? Modifique os tipos das variáveis para *float*. Aconteceu o esperado? Por quê?

2. a) Defina um construtor adicional na classe *ContaCor* que recebe como argumentos apenas o nome do cliente, o número da conta e a senha; lembrando que o estado da conta deve ser ajustado para 1. Para evitar repetição de código, é possível usar *this()* do mesmo modo que se usa *super()*. A diferença é que, ao invés de chamar o construtor de uma superclasse, *this()* chama um outro construtor na própria classe. Modifique o construtor de *ContaCor* que recebe como argumentos o nome do cliente, o saldo inicial, o número da conta e a senha para que use o construtor que você acabou de definir. Implemente na classe *ContaCor* operações para obter os valores de *titular*, *numConta* e *senha*.

b) Crie a classe *ExemploDoisConstrutores* que contenha um método *main()*. Esse método instancia dois objetos do tipo *ContaCor* passando os mesmos valores para *titular*, *numConta* e *senha*. O primeiro objeto deve ser criado usando o construtor com quatro parâmetros, passando 0 (zero) como valor do *saldoAtual*. O segundo deve ser criado usando o construtor definido no Item (a). Inclua código no programa principal para imprimir as informações dos dois objetos criados. Compile e execute *ExemploDoisConstrutores*. Pelos resultados obtidos, que conclusões você pode tomar?

### Pacote *br.unicamp.ic.mc302.documento*:

3. Abra os arquivos *Documento.java*, *Carta.java*, *Telegrama.java* e *ExemploPolimorfismoSemRedefinicao.java*. Tente compilá-los. Por que a compilação falha? Usando o operador *instanceof* definido por Java e *casting* de objetos, modifique a classe *ExemploPolimorfismoSemRedefinicao* para que o compilador pare de acusar erros nela. O operador *instanceof* diz se um objeto é de um determinado tipo. Por exemplo:

```
ContaCor c = new ContaEsp();
if(c instanceof ContaEsp) { // true! Entra no if.
    ContaEsp cEsp = (ContaEsp)c; // casting }
if(c instanceof ContaCor) { // também true. Entra no if. }
if(c instanceof String) { // falso! c não é do tipo String. }
```

Por que o problema foi resolvido?

### Pacote *br.unicamp.ic.mc302.contaCor*:

4. Abra os arquivos *ContaCor.java*, *ContaEsp.java* e *ExemploPolimorfismoComRedefinicao.java*. Compile-os. Por que a compilação não funcionou? Coloque em comentários as linhas que têm erro no arquivo *ExemploPolimorfismoComRedefinicao.java* (preceda cada linha com o símbolo *"/"*, como no trecho de código acima). Tente compilar o arquivo novamente. Execute a classe *ExemploPolimorfismoComRedefinicao* e observe o resultado produzido. Inclua nas operações *debitarValor()* de *ContaCor* e *ContaEsp* uma linha que imprime o nome da classe na qual o método está definido. Compile os arquivos e execute novamente a classe *ExemploPolimorfismoComRedefinicao*. O que você pode dizer sobre o funcionamento do exemplo, com relação à execução da operação *debitarValor()*?

5. Adicione o modificador **final** à operação **debitaValor()** da classe **ContaCor**. Tente compilar a classe **ContaEsp**. Por que não compila? Desfaça a modificação e adicione o modificador **final** à operação **debitaValor()** da classe **ContaEsp**. Tente compilar a classe **ContaEsp**.

6. Crie uma classe **ContaEspPoup**, que herda de **ContaEsp** e redefina o método **debitaValor()**. O que acontece? Por quê?

---

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

7. Abra os arquivos **FilaVeiculo.java**, **Veiculo.java**, **Carro.java**, **Caminhao.java** e **Inicial.java**. Compile e execute a classe **Inicial**. Leia o código da operação **mostraFila()** da classe **FilaVeiculo** e da operação **mostra()** nas classes **Carro**, **Caminhao** e **Veiculo**. Modifique o programa para imprimir o estado completo dos objetos do tipo **Carro** e **Caminhao**. Modifique a visibilidade da operação **mostra()** da classe **Carro** para **private**. Tente recompilá-la. Que conclusões você tira desse resultado?

---

8. Em uma oficina mecânica trabalham 10 funcionários. A oficina executa 3 tipos de serviços diferentes, sendo que Tipo1 custa R\$ 15, Tipo2 custa R\$ 50,00 e Tipo3 custa R\$ 40,00. Cada funcionário recebe 10% por cada serviço realizado. Entretanto, se o funcionário é um gerente, ele recebe uma comissão de 15%, além de ter um salário mensal mais alto do que um funcionário comum. Considere que a oficina tenha 1 gerente. Faça um programa Java que recebe o número do funcionário (entre 1 e 10), a informação se ele é ou não gerente e quantas vezes ele executou cada um dos 3 serviços. Ao final, calcule a comissão de cada funcionário, bem como o seu pagamento do final do mês. Use o conceito de operações polimórficas na sua solução.

Crie um pacote *br.unicamp.mc302.oficina* para resolver o exercício, uma classe java **ExemploOficina** com apenas um método **main** para exemplificar a sua solução, além das classes que julgar necessárias, para representar os Funcionários. Utilize, obrigatoriamente, o conceito de polimorfismo para diferenciar Gerente de funcionários Comuns; e imprima no método **main** os resultados dos métodos que disponibilizar.