Fundamentos Java e Orientação a Objetos

Por
Thiago Faria

5.22. Desafio: polimorfismo e classes abstratas

Continuando a série de desafios sobre o sistema financeiro, adicionaremos agora mais métodos e classes ao projeto.

Esperamos que você esteja se divertindo, trabalhando com diversas classes ao mesmo tempo e simulando um projeto real.

Neste desafio, você não precisou pedir nada ao seu analista. Você mesmo decidiu melhorar o código-fonte.

Ao analisar as classes do projeto, percebemos que a classe Pessoa serve somente como uma classe pai para Cliente e Fornecedor. Não gostaríamos que a classe Pessoa fosse instanciada em nenhum momento. Para chegar a essa conclusão, é só pensarmos: faz sentido, em um sistema financeiro, existir um objeto de uma pessoa? Quem seria essa pessoa? Um cliente? Um fornecedor? Um funcionário?

Chegamos a conclusão que Pessoa é muito abstrata para ser instanciada, portanto, está mais do que claro que essa classe deve receber a palavra-chave abstract na declaração dela.

```
public abstract class Pessoa {

	...

}
```

Se pensarmos nesse mesmo sentido, a classe Conta também está eleita a ser abstrada, afinal, não existe motivo em termos uma instância apenas de Conta. Vamos analisar: qual seria o sentido de existir um objeto da classe Conta? Que tipo de conta seria? Conta a pagar? Conta a receber? Temos que transformar a classe Conta também em abstrata!

```
public abstract class Conta {

	...

}
```

Achou que o exercício seria apenas isso? Você se enganou! Vamos começar agora o desafio. :)

Você precisa exibir relatórios de contas a pagar e receber, pois ajudará no controle do que tem para pagar e receber na empresa. Para isso, precisará de uma nova classe chamada RelatorioContas. Essa nova classe deve possuir um método que recebe um array de contas e exibe o detalhamento de todas elas.

```
package com.algaworks.cursojava.financeiro;

import com.algaworks.cursojava.financeiro.modelo.Conta;

public class RelatorioContas {

	public void exibirListagem(Conta[] contas) {
		// codifique a listagem e detalhamento aqui
	}

}
```

A classe RelatorioContas não deve conhecer os detalhes das subclasses de Conta (ou seja, ContaPagar e ContaReceber). Não seria uma boa prática essa classe obter os detalhes das contas para mostrá-los na tela, por isso, temos uma excelente ideia de uso da orientação a objetos. Na classe Conta, adicione um método abstrato (não implementado), como no código abaixo:

```
public abstract class Conta {

	...

	public abstract void exibirDetalhes();

	...

}
```

Ao fazer isso, você será obrigado a implementar esse novo método nas subclasses ContaPagar e ContaReceber. Então, faça isso! Esse método deve exibir todos os detalhes do objeto em um formato legal para aparecer em um relatório.

Agora você pode voltar à classe RelatorioContas e invocar o método exibirDetalhes() durante a listagem. Veja que a classe RelatorioContas só deve conhecer o nome do método exibirDetalhes(), mais nada!

Para testar, compile e execute a classe Principal abaixo:

```
package com.algaworks.cursojava.financeiro;

import com.algaworks.cursojava.financeiro.modelo.ContaPagar;
import com.algaworks.cursojava.financeiro.modelo.ContaReceber;
import com.algaworks.cursojava.financeiro.modelo.Conta;
import com.algaworks.cursojava.financeiro.modelo.Fornecedor;
import com.algaworks.cursojava.financeiro.modelo.Cliente;

public class Principal {
```

```
	public static void main(String[] args) {
		// instanciando fornecedores
		Fornecedor imobiliaria = new Fornecedor();
		imobiliaria.setNome("Casa & Cia Negócios Imobiliários");
		Fornecedor mercado = new Fornecedor();
		mercado.setNome("Mercado do João");

		// instanciando clientes
		Cliente atacadista = new Cliente();
		atacadista.setNome("Triângulo Quadrado Atacadista");
		Cliente telecom = new Cliente();
		telecom.setNome("FoneNet Telecomunicações");

		// instanciando contas a pagar
		ContaPagar contaPagar1 = new ContaPagar();
		contaPagar1.setDescricao("Aluguel da matriz");
		contaPagar1.setValor(1230d);
		contaPagar1.setDataVencimento("10/05/2012");
		contaPagar1.setFornecedor(imobiliaria);

		ContaPagar contaPagar2 = new ContaPagar(mercado, "Compras do mês", 390d, "19/05/2012");

		// instanciando contas a receber
		ContaReceber contaReceber1 = new ContaReceber();
		contaReceber1.setDescricao("Desenvolvimento de projeto de logística em Java");
		contaReceber1.setValor(89500d);
		contaReceber1.setDataVencimento("23/05/2012");
		contaReceber1.setCliente(atacadista);

		ContaReceber contaReceber2 = new ContaReceber(telecom, "Manutenção em sistema de conta online",
			53200d, "13/05/2012");

		// exibe listagem de todas as contas com detalhamento
		RelatorioContas relatorio = new RelatorioContas();
		Conta[] contas = new Conta[]{contaPagar1, contaPagar2, contaReceber1, contaReceber2};

		relatorio.exibirListagem(contas);
	}

}
```

Acesse o código-fonte desta aula

Comentários sobre esta aula

Paulo Mauricio Salles Rodrigue - 16/05/2012 às 22:04

Normandes,
Tentei fazer alterações na classe RelatorioContas, conforme abaixo, mas o Eclipse dá a seguinte mensagem de erro na linha do IF:
"The method getFornecedor() is undefined for the type Conta"

Por que?
Abraço.

```
public void exibirListagem(Conta[] contas) {
// codifique a listagem e detalhamento aqui
System.out.println("-------------------------------------------------------------");
System.out.println("RELATÓRIO DE CONTAS A PAGAR E RECEBER");
System.out.println("-------------------------------------------------------------");
for (int i=0;i<contas.length;i++){
if ((contas[i]).getFornecedor().getNome() !=null) {
System.out.println("Fornecedor          Descrição        Valor   Vencimento Situação");
System.out.println("-------------------------------------------------------------------");
} else {
System.out.println("Cliente            Descrição        Valor   Vencimento Situação");
System.out.println("-------------------------------------------------------------------");
}
((Conta) contas[i]).exibirDetalhes();
}
System.out.println("-------------------------------------------------------------------------------");
}
```

Normandes Júnior INSTRUTOR - 14/06/2012 às 21:05

Desculpa, só hoje vi sua pergunta. Acho que a notificação veio duplicada e eu acabei me confundindo. Espero ainda estar em tempo de ajudar.
O método getFornecedor() não é definido na classe Conta, somente na classe ContaPagar. Lembre-se que você só consegue chamar os métodos que estão no tipo de dados, apesar do método ser executado no objeto.
Ou seja,

```
Conta conta = new ContaPagar()
```

Você só pode chamar os métodos que estão definidos na classe Conta, mas eles serão executados na classe ContaPagar.

Agora, se você quiser realmente chamar o método getFornecedor(), você precisaria fazer um cast neste objeto:

```
ContaPagar contaPagar = (ContaPagar) conta;
```

Se ainda não está claro, por favor, não deixe de perguntar.

Compartilhe esta aula com seus amigos

Twitter Facebook

1. Introdução

1.1. Como aprender Java? 5m 50s GRÁTIS
1.2. A história do Java 2m 46s GRÁTIS
1.3. As plataformas Java e como elas evoluem 10m 31s GRÁTIS
1.4. Máquina virtual Java 8m 45s GRÁTIS
1.5. Baixando, instalando e configurando a JDK 7m 59s GRÁTIS
1.6. Exercício: instalação da JDK GRÁTIS

2. Fundamentos da linguagem

2.1. Codificando, compilando e executando o programa "oi mundo" 13m 10s GRÁTIS
2.2. Exercício: codificando um primeiro programa GRÁTIS
2.3. Comentários 3m 3s GRÁTIS
2.4. Sequências de escape 5m 14s GRÁTIS
2.5. Palavras reservadas 3m 32s GRÁTIS
2.6. Convenções de código 2m 28s GRÁTIS
2.7. Trabalhando com variáveis 6m 18s GRÁTIS
2.8. Nomeando variáveis 5m 42s GRÁTIS
2.9. Operadores aritméticos 9m 36s GRÁTIS
2.10. Exercício: variáveis e operadores aritméticos GRÁTIS
2.11. Tipos primitivos 12m 0s GRÁTIS
2.12. Outros operadores de atribuição 4m 43s GRÁTIS
2.13. Conversão de tipos primitivos 12m 39s GRÁTIS
2.14. Promoção aritmética 6m 25s GRÁTIS
2.15. Exercício: tipos primitivos e outros operadores de atribuição GRÁTIS
2.16. Trabalhando com strings 7m 5s GRÁTIS
2.17. Recebendo entrada de dados 7m 41s GRÁTIS
2.18. Operadores de comparação e igualdade 6m 40s GRÁTIS
2.19. Estruturas de controle if, else if e else 12m 23s GRÁTIS
2.20. Exercício: Strings, entrada de dados, operadores de comparação e if else GRÁTIS
2.21. Escopo de variáveis 6m 3s GRÁTIS
2.22. Operadores lógicos 15m 13s GRÁTIS
2.23. Exercício: operadores lógicos GRÁTIS
2.24. Estrutura de controle switch 7m 10s GRÁTIS
2.25. Operador ternário 6m 49s GRÁTIS
2.26. Operadores de incremento e decremento 8m 11s GRÁTIS
2.27. Estrutura de controle while 5m 45s GRÁTIS
2.28. Estrutura de controle do-while 3m 47s GRÁTIS
2.29. Estrutura de controle for 4m 15s GRÁTIS
2.30. Cláusulas break e continue 7m 2s GRÁTIS
2.31. Exercício: operador ternário, decremento e estruturas de repetição GRÁTIS
2.32. Introdução e instalação do Eclipse IDE 13m 40s GRÁTIS
2.33. Depurando códigos com o Eclipse 8m 43s GRÁTIS
2.34. Exercício: instalando o Eclipse IDE GRÁTIS

3. Orientação a Objetos - parte 1

3.1. O que é POO? 2m 57s GRÁTIS
3.2. Classes e objetos 5m 16s GRÁTIS
3.3. Criando uma classe com atributos 2m 48s GRÁTIS
3.4. Instanciando objetos 7m 59s GRÁTIS
3.5. Acessando atributos de objetos 8m 32s GRÁTIS
3.6. Exercício: instanciando e acessando atributos do objeto GRÁTIS
3.7. Composição de objetos 9m 28s GRÁTIS
3.8. Valores padrão 5m 59s GRÁTIS
3.9. Variáveis referenciam objetos 9m 22s GRÁTIS
3.10. Criando, nomeando e chamando métodos 8m 2s GRÁTIS
3.11. Métodos com retorno 11m 13s GRÁTIS
3.12. Passando argumentos para métodos 5m 25s GRÁTIS
3.13. Argumentos por valor ou referência 7m 0s GRÁTIS
3.14. Exercício: composição de objetos e chamada de métodos GRÁTIS

4. Wrappers, boxing e arrays

4.1. Wrappers do java.lang 3m 31s GRÁTIS
4.2. Boxing 6m 47s GRÁTIS
4.3. Desafio: wrappers e boxing GRÁTIS
4.4. Trabalhando com arrays 16m 37s GRÁTIS
4.5. Exercício: arrays GRÁTIS

5. Orientação a Objetos - parte 2

5.1. Introdução à UML e diagrama de classes 7m 31s GRÁTIS
5.2. Desafio: diagrama de classes GRÁTIS
5.3. O objeto this 8m 18s GRÁTIS
5.4. Construtores 11m 43s GRÁTIS
5.5. Encapsulamento e modificadores de acesso public e private 11m 7s GRÁTIS
5.6. Criando JavaBeans 8m 40s GRÁTIS
5.7. Desafio: objeto this, construtores e JavaBeans GRÁTIS
5.8. Organizando os projetos em pacotes 11m 51s GRÁTIS
5.9. Modificador de acesso default 6m 55s GRÁTIS
5.10. Modificadores static e final 12m 40s GRÁTIS
5.11. Desafio: static e final GRÁTIS
5.12. Enumerações 17m 26s GRÁTIS
5.13. Desafio: pacotes e enumerações GRÁTIS
5.14. Herança e modificador protected 10m 42s GRÁTIS
5.15. Classe java.lang.Object 4m 13s GRÁTIS

6. Tópicos avançados

AlgaWorks Softwares, Treinamentos e Serviços Ltda
Av. Afonso Pena, 3538, Átrio Business Center
CEP. 38400-710 - Uberlândia/MG - Brasil
Tel. +55 (34) 8400-6931 - comercial@algaworks.com