Fundamentos Java e Orientação a Objetos

Por
Thiago Faria

5.24. Exercício: interfaces e polimorfismo

Seu tio tem uma corretora de seguros e precisa de um sistema para calcular o valores de apólices para facilitar o processo de venda de seguros.

No momento, a corretora trabalha apenas com seguros de carros e imóveis, mas em breve serão adicionados outros tipos de contratos de seguros. Essa informação é muito importante, pois o software deve ter a capacidade de receber novos tipos de seguros facilmente, como por exemplo, notebook, barcos, aeronaves, etc.

Cada tipo de apólice possui uma fórmula diferente para calcular seu valor. Uma apólice de imóvel é calculada diferente de uma de um carro.

Usando interfaces e polimorfismo, precisamos desenvolver algo simples e inteligente para resolver o problema do tiozão.

1. Vamos criar uma interface chamada Seguravel.

```
public interface Seguravel {

	public double calcularValorApolice();
	public String obterDescricao();

}
```

A interface Seguravel deve ser implementada por classes que representam objetos que podem ser assegurados. Essas classes devem implementar os métodos da interface.

2. Agora criamos a classe Carro, que implementa a interface Seguravel, pois a corretora está apta a vender seguros para carros.

```
public class Carro implements Seguravel {

	...

}
```

3. Se Carro diz que implementa a interface Seguravel, ela deve implementar todos os métodos especificados na interface, pois uma interface é um contrato, ou seja, a classe deve garantir que faz o que ela pede.

```
public class Carro implements Seguravel {

	private double valorMercado;
	private int anoFabricacao;

	public Carro(double valorMercado, int anoFabricacao) {
		this.valorMercado = valorMercado;
		this.anoFabricacao = anoFabricacao;
	}

	public String obterDescricao() {
		return "Carro ano " + this.anoFabricacao + " com valor de mercado "
			+ this.valorMercado;
	}

	public double calcularValorApolice() {
		// cálculos fictícios do valor de uma apólice de um carro
		double valorApolice = this.valorMercado * 0.04;
		if (anoFabricacao < 2000) {
			valorApolice = valorApolice * 0.90;
		}
		return valorApolice;
	}

}
```

No código acima, incluímos os métodos obterDescricao() e calcularValorApolice(), além de dois atributos e um construtor. Os atributos são usados pelos métodos para calcular o valor da apólice e retornar a descrição do carro.

4. Criamos a classe Imovel e implementamos os métodos da interface. Veja que a fórmula do cálculo do seguro é diferente, além de essa classe possuir atributos também diferentes.

```
public class Imovel implements Seguravel {

	private double valorMercado;
	private int areaConstruida;

	public Imovel(double valorMercado, int areaConstruida) {
		this.valorMercado = valorMercado;
		this.areaConstruida = areaConstruida;
	}

	public String obterDescricao() {
		return "Imóvel com área construída de " + this.areaConstruida + "m2 e valor de mercado "
```

```
                + this.valorMercado;
        }

        public double calcularValorApolice() {
                // cálculos fictícios do valor de uma apólice de um imóvel
                double valorApolice = this.valorMercado * 0.003;
                valorApolice = valorApolice + (areaConstruida * 0.5);
                return valorApolice;
        }

}
```

As fórmulas para calcular os valores das apólices são meramente fictícias. Claro que, na realidade, as fórmulas são totalmente diferentes e mais complexas.

5. Agora que já temos as classes que implementam a interface Seguravel, vamos criar uma outra classe chamada CorretoraSeguros. Ela será responsável por fazer a proposta de seguro do bem segurável e mostrar na tela.

```
public class CorretoraSeguros {

        public void fazerPropostaSeguro(Seguravel objetoSeguravel) {
                System.out.println("--------------------------------");
                System.out.println("Corretora de Seguros - Proposta");
                System.out.println("--------------------------------");
                System.out.println(objetoSeguravel.obterDescricao());
                System.out.println("Valor da apólice: " + objetoSeguravel.calcularValorApolice());
                System.out.println("--------------------------------
");
        }

}
```

Veja que o método fazerPropostaSeguro() recebe como parâmetro um objeto do tipo Seguravel, ou seja, pode ser um Carro ou um Imovel, mas no futuro poderia receber também uma Aeronave ou Barco, desde que essas classes implementem a interface Seguravel.

6. Para testar tudo que foi feito, vamos criar uma classe chamada Principal, que possui o método main().

```
public class Principal {

        public static void main(String[] args) {
                CorretoraSeguros corretora = new CorretoraSeguros();

                Carro meuCarro = new Carro(45000d, 2012);
                Imovel minhaCasa = new Imovel(920000, 320);

                corretora.fazerPropostaSeguro(meuCarro);
                corretora.fazerPropostaSeguro(minhaCasa);
        }

}
```

No método main(), instanciamos uma corretora de seguros, um carro e um imóvel, depois chamamos o método fazerPropostaSeguro() da corretora passando como parâmetro o carro e depois o imóvel.

8. Compile tudo e execute a classe Principal.

9. Quando tudo estiver funcionando, crie uma classe chamada Barco e outra chamada Notebook. As duas classes devem implementar a interface Seguravel. Implemente os métodos da interface e adicione algumas linhas na classe Principal para testar as novas classes.

Acesse o código-fonte desta aula

Comentários sobre esta aula

Nenhum comentário para esta aula. Efetue login para enviar uma mensagem.

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
5.16. Sobreposição 7m 48s GRÁTIS
5.17. Desafio: herança e sobreposição GRÁTIS
5.18. Sobrecarga 7m 48s GRÁTIS
5.19. Exercício: sobrecarga GRÁTIS
5.20. Polimorfismo, casting de objetos e instanceof 18m 49s GRÁTIS
5.21. Classes abstratas 9m 49s GRÁTIS
5.22. Desafio: polimorfismo e classes abstratas GRÁTIS
5.23. Interfaces 11m 49s GRÁTIS
5.24. Exercício: interfaces e polimorfismo GRÁTIS

6. Tópicos avançados

6.1. Coleta de lixo 8m 40s GRÁTIS
6.2. Classe java.lang.Math 16m 6s GRÁTIS
6.3. Desafio: classe java.lang.Math GRÁTIS
6.4. Tratando e lançando exceções 29m 12s GRÁTIS
6.5. Desafio: exceções GRÁTIS
6.6. Classes String, StringBuffer e StringBuilder 8m 26s GRÁTIS
6.7. Trabalhando com datas 19m 28s GRÁTIS
6.8. Desafio: datas GRÁTIS
6.9. Trabalhando com números 9m 12s GRÁTIS
6.10. Desafio: números GRÁTIS
6.11. Collections Framework 22m 25s GRÁTIS
6.12. Desafio: collections GRÁTIS
6.13. Arquivos JAR 6m 19s GRÁTIS
6.14. Exercício: arquivos JAR GRÁTIS
6.15. Documentação javadoc 9m 55s GRÁTIS

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