Fundamentos Java e Orientação a Objetos

Por
Thiago Faria

6.12. Desafio: collections

6. Tópicos avançados

O custo com políticos no Brasil é um dos mais altos no mundo.

O valor do contra-cheque de um político corresponde a cerca de um quarto de tudo o que recebem por meio de inúmeras verbas e auxílios extraordinários.

Você, político honesto e programador Java, resolveu desenvolver um software para calcular os salários dos políticos por partido e cargo. Esse software irá lhe ajudar a propor mudanças dentro do governo.

As classes Cargo e Politico foram fornecidas para você pelo Ministério Público Eleitoral.

A classe Cargo representa o cargo de um político, como por exemplo "vereador", "deputado estadual" ou "prefeito". Um cargo possui uma descrição e o valor do salário.

```
import java.math.BigDecimal;

public class Cargo {

	private String descricao;
	private BigDecimal salario;

	public String getDescricao() {
		return this.descricao;
	}

	public void setDescricao(String descricao) {
		this.descricao = descricao;
	}

	public BigDecimal getSalario() {
		return this.salario;
	}

	public void setSalario(BigDecimal salario) {
		this.salario = salario;
	}

}
```

A classe Politico representa, errrr... um político. Essa classe possui os atributos "nome" e "cargo", que referencia um objeto do tipo Cargo.

```
public class Politico {

	private String nome;
	private Cargo cargo;

	public String getNome() {
		return this.nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public Cargo getCargo() {
		return this.cargo;
	}

	public void setCargo(Cargo cargo) {
		this.cargo = cargo;
	}

}
```

Agora que você já tem as classes Cargo e Politico, precisa criar uma outra classe para organizar os políticos e seus partidos e calcular os gastos totais dos salários.

Para sua sorte, estamos entregando para você a classe Governo, que está quase pronta. As únicas coisas que faltam são as implementações os métodos calcularGastosComSalario() e calcularGastosComSalarioParaCargo().

```
import java.util.List;
import java.util.ArrayList;
import java.util.Map;
import java.util.HashMap;
import java.math.BigDecimal;

public class Governo {
```

```
	// armazena uma lista de políticos por estado da federação
	private Map<String, List<Politico>> partidosPoliticos;

	public Governo() {
		this.partidosPoliticos = new HashMap<String, List<Politico>>();
	}

	public void adicionarPolitico(String partidoPolitico, Politico politico) {
		// recupera a lista de políticos para um partido
		List<Politico> politicos = this.partidosPoliticos.get(partidoPolitico);

		// se não existir uma lista de políticos para o partido informado,
		// devemos instanciar uma nova lista (pois é a primeira inclusão neste partido)
		if (politicos == null) {
			politicos = new ArrayList<Politico>();
		}

		// adiciona o político recebido como parâmetro à lista de políticos
		politicos.add(politico);

		// adiciona a lista de políticos ao mapa de partidos usando
		// como chave o nome do partido
		this.partidosPoliticos.put(partidoPolitico, politicos);
	}

	public BigDecimal calcularGastosComSalario(String partidoPolitico) {
		// implementar busca de políticos para o partido informado
		// e cálculo dos salários
	}

	public BigDecimal calcularGastosComSalarioParaCargo(Cargo cargo, String partidoPolitico) {
		// implementar busca dos políticos para o partido e cargo informados
		// e cálculo dos salários
	}

}
```

A classe Governo possui uma variável de instância para armazenar os políticos por partido. Para isso, é usado um mapa (interface Map).

```
private Map<String, List<Politico>> partidosPoliticos;
```

Essa linha está declarando um mapa (do Collections Framework) onde a chave é do tipo String e o valor do tipo List. Na chave do mapa, deve-se atribuir a sigla do partido. No valor, atribui-se um objeto do tipo List, como por exemplo um ArrayList.

A instanciação do mapa foi feita para você no construtor de Governo.

```
public Governo() {
	this.partidosPoliticos = new HashMap<String, List<Politico>>();
}
```

O método adicionarPolitico() inclui um novo político à lista de políticos, que está dentro do mapa. Se a lista ainda não estiver no mapa, o método deve criar um novo registro para a chave igual a sigla do partido. Essas regras já foram implementadas no método para você.

```
public void adicionarPolitico(String partidoPolitico, Politico politico) {
	...
}
```

Implemente apenas os métodos calcularGastosComSalario() e calcularGastosComSalarioParaCargo().

```
public BigDecimal calcularGastosComSalario(String partidoPolitico) {
	// implementar busca de políticos para o partido informado
	// e cálculo dos salários
}

public BigDecimal calcularGastosComSalarioParaCargo(Cargo cargo, String partidoPolitico) {
	// implementar busca dos políticos para o partido e cargo informados
	// e cálculo dos salários
}
```

Após implementar os métodos de cálculos, crie uma classe chamada Principal com o código-fonte abaixo:

```
import java.text.DecimalFormat;
import java.math.BigDecimal;

public class Principal {

	public static void main(String[] args) {
		DecimalFormat formatador = new DecimalFormat("R$ #,##0.00");

		// instancia governo
		Governo governo = new Governo();

		// instancia cargos
		Cargo vereador = new Cargo();
		vereador.setDescricao("Vereador");
		vereador.setSalario(new BigDecimal(16000));

		Cargo deputadoEstadual = new Cargo();
		deputadoEstadual.setDescricao("Deputado estadual");
		deputadoEstadual.setSalario(new BigDecimal(25000));
```

```
        Cargo prefeito = new Cargo();
        prefeito.setDescricao("Prefeito");
        prefeito.setSalario(new BigDecimal(27000));

        // adiciona vereadores
        Politico politico = new Politico();
        politico.setNome("João das Couves");
        politico.setCargo(vereador);
        governo.adicionarPolitico("RBLH", politico);

        politico = new Politico();
        politico.setNome("Zé Mané");
        politico.setCargo(vereador);
        governo.adicionarPolitico("PCOR", politico);

        politico = new Politico();
        politico.setNome("Maria Carvalho");
        politico.setCargo(vereador);
        governo.adicionarPolitico("LDRS", politico);

        politico = new Politico();
        politico.setNome("Maria Carvalho");
        politico.setCargo(vereador);
        governo.adicionarPolitico("LDRS", politico);

        // adiciona deputados estaduais
        politico = new Politico();
        politico.setNome("Josefa Silva");
        politico.setCargo(deputadoEstadual);
        governo.adicionarPolitico("LDRS", politico);

        politico = new Politico();
        politico.setNome("Fátima Gonçalves");
        politico.setCargo(deputadoEstadual);
        governo.adicionarPolitico("PCOR", politico);

        // adiciona prefeito
        politico = new Politico();
        politico.setNome("Sebastião Mendes");
        politico.setCargo(prefeito);
        governo.adicionarPolitico("PCOR", politico);

        // calcula gastos com partidos
        BigDecimal gastosPcor = governo.calcularGastosComSalario("PCOR");
        System.out.println("Gastos com partido PCOR: " + formatador.format(gastosPcor.doubleValue()));

        BigDecimal gastosLdrs = governo.calcularGastosComSalario("LDRS");
        System.out.println("Gastos com partido LDRS: " + formatador.format(gastosLdrs.doubleValue()));

        // calcula gastos com partidos para determinados cargos
        BigDecimal gastosVereadoresLdrs = governo.calcularGastosComSalarioParaCargo(vereador, "LDRS");
        System.out.println("Gastos com vereadores do partido LDRS: "
                + formatador.format(gastosVereadoresLdrs.doubleValue()));

        BigDecimal gastosDeputadosEstaduaisLdrs = governo.calcularGastosComSalarioParaCargo(deputadoEstadual, "LDRS");
        System.out.println("Gastos com deputados estaduais do partido LDRS: "
                + formatador.format(gastosDeputadosEstaduaisLdrs.doubleValue()));

        BigDecimal gastosPrefeitosLdrs = governo.calcularGastosComSalarioParaCargo(prefeito, "LDRS");
        System.out.println("Gastos com prefeitos do partido LDRS: "
                + formatador.format(gastosPrefeitosLdrs.doubleValue()));
    }

}
```

Compile e execute. Quando estiver funcionando, altere a classe Principal para incluir novos partidos e políticos.

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
6.16. Desafio: javadoc GRÁTIS
6.17. Próximos passos 4m 8s GRÁTIS
6.18. Conclusão 2m 6s 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