02

Extraindo responsabilidades da classe NotaFiscal

Suponha que as nossas classes foram alteradas por outro programador para adicionar novas funcionalidades. Substitua o código das classes `BalancoEmpresa` e `Divida` em seu projeto pelo seguinte:

```
public class BalancoEmpresa {
    private HashMap<String, Divida> dividas = new HashMap<String, Divida>();

    public void registraDivida(String credor, String cnpjCredor, double valor) {
        Divida divida = new Divida();
        divida.setTotal(valor);
        divida.setCredor(credor);
        divida.setCnpjCredor(cnpjCredor);
        dividas.put(cnpjCredor, divida);
    }

    public void pagaDivida(String cnpjCredor, double valor, String nomePagador, String cnpjPaga
        Divida divida = dividas.get(cnpjCredor);
        if (divida != null) {
            Pagamento pagamento = new Pagamento();
            pagamento.setCnpjPagador(cnpjPagador);
            pagamento.setPagador(nomePagador);
            pagamento.setValor(valor);
            divida.registra(pagamento);
        }
    }
}
```

```
public class Divida {
    private double total;
    private double valorPago;
    private String credor;
    private String cnpjCredor;
    private ArrayList<Pagamento> pagamentos = new ArrayList<Pagamento>();

    public boolean cnpjValido() {
        return primeiroDigitoVerificadorDoCnpj() == primeiroDigitoCorretoParaCnpj()
                && segundoDigitoVerificadorDoCnpj() == segundoDigitoCorretoParaCnpj();
    }
    public String getCnpjCredor() {
        return this.cnpjCredor;
    }
    public String getCredor() {
        return this.credor;
    }
    public double getTotal() {
        return this.total;
    }
    public double getValorPago() {
        return this.valorPago;
    }
```

```
    private void paga(double valor) {
        if (valor < 0) {
            throw new IllegalArgumentException("Valor invalido para pagamento");
        }
        if (valor > 100) {
            valor = valor - 8;
        }
        this.valorPago += valor;
    }
    public ArrayList<Pagamento> pagamentosAntesDe(Calendar data) {
        ArrayList<Pagamento> pagamentosFiltrados = new ArrayList<Pagamento>();
        for (Pagamento pagamento : this.pagamentos) {
            if (pagamento.getData().before(data)) {
                pagamentosFiltrados.add(pagamento);
            }
        }
        return pagamentosFiltrados;
    }
    public ArrayList<Pagamento> pagamentosComValorMaiorQue(double valorMinimo) {
        ArrayList<Pagamento> pagamentosFiltrados = new ArrayList<Pagamento>();
        for (Pagamento pagamento : this.pagamentos) {
            if (pagamento.getValor() > valorMinimo) {
                pagamentosFiltrados.add(pagamento);
            }
        }
        return pagamentosFiltrados;
    }
    public ArrayList<Pagamento> pagamentosDo(String cnpjPagador) {
        ArrayList<Pagamento> pagamentosFiltrados = new ArrayList<Pagamento>();
        for (Pagamento pagamento : this.pagamentos) {
            if (pagamento.getCnpjPagador().equals(cnpjPagador)) {
                pagamentosFiltrados.add(pagamento);
            }
        }
        return pagamentosFiltrados;
    }
    private int primeiroDigitoCorretoParaCnpj() {
        // Calcula o primeiro dígito verificador correto para
        // o CNPJ armazenado no atributo valor
        return 0;
    }
    private int primeiroDigitoVerificadorDoCnpj() {
        // Extrai o primeiro dígito verificador do CNPJ armazenado
        // no atributo valor
        return 0;
    }
    public void registra(Pagamento pagamento) {
        this.pagamentos.add(pagamento);
        paga(pagamento.getValor());
    }
    private int segundoDigitoCorretoParaCnpj() {
        // Calcula o segundo dígito verificador correto para
        // o CNPJ armazenado no atributo valor
        return 0;
    }
    private int segundoDigitoVerificadorDoCnpj() {
        // Extrai o segundo dígito verificador do CNPJ armazenado
        // no atributo valor
```

```
        return 0;
    }
    public void setCnpjCredor(String cnpjCredor) {
        this.cnpjCredor = cnpjCredor;
    }
    public void setCredor(String credor) {
        this.credor = credor;
    }
    public void setTotal(double total) {
        this.total = total;
    }
    public double valorAPagar() {
        return this.total - this.valorPago;
    }
}
```

Além disso foi criada a classe `Pagamento`. Crie tal classe em seu projeto e cole o seguinte código:

```
public class Pagamento {
    private String pagador;
    private String cnpjPagador;
    private double valor;
    private Calendar data;
    public String getPagador() {
        return this.pagador;
    }
    public void setPagador(String pagador) {
        this.pagador = pagador;
    }
    public String getCnpjPagador() {
        return this.cnpjPagador;
    }
    public void setCnpjPagador(String cnpjPagador) {
        this.cnpjPagador = cnpjPagador;
    }
    public double getValor() {
        return this.valor;
    }
    public void setValor(double valor) {
        this.valor = valor;
    }
    public Calendar getData() {
        return this.data;
    }
    public void setData(Calendar data) {
        this.data = data;
    }
}
```

Vamos aplicar a primeira refatoração que vimos nesta aula. Crie a classe `Cnpj` que guarde um valor e saiba validá-lo e faça com que a classe `Divida` a utilize para armazenar o CNPJ do credor.

Faça com que a classe `Cnpj` torne obrigatório a passagem de um valor inicial para o objeto. Crie um construtor que receba o valor inicial como parâmetro e preencha o atributo da classe com esse valor.

Altere o método `registraDivida` da classe `BalancoEmpresa` para que ele receba um `Cnpj` ao invés de uma `String` de CNPJ.

Cole aqui o código das classes `Cnpj` , `Divida` e `BalancoEmpresa` .

Responda

INSERIR CÓDIGO | FORMATAÇÃO