PUC Minas

Programação Modular

PUC Minas
Bacharelado em Engenharia de Software
Prof. Daniel Kansaon

Singleton

```
class Equipe {
    private String nome;

    public Usuario(String nome) {
        this.nome = nome;
    }
}
```

Singleton

```
class GerenciadorDeEquipes {
    private static GerenciadorDeEquipes instance;
    private List<Equipe> equipes;

    private GerenciadorDeEquipes() {
        equipes = new ArrayList<>();
    }

    public static GerenciadorDeEquipes getInstance(){
        if (instance == null) {
            instance = new GerenciadorDeEquipes();
        }
        return instance;
    }
}
```

Singleton

```
class GerenciadorDeEquipes {
    private static GerenciadorDeEquipes instance;
    private List<Equipe> equipes;

   public void adicionarEquipe(Equipe equipe) {
        equipes.add(equipe);
   }
}
```

Singleton

```
public static void main(String[] args) {

    GerenciadorDeEquipes equipes = GerenciadorDeEquipes.getInstance();

    equipes.adicionar(New Equipe("Equipe 1"));
}
```

Singleton

```
public static void main(String[] args) {

    GerenciadorDeEquipes equipes = GerenciadorDeEquipes.getInstance();
    GerenciadorDeEquipes equipes2 = GerenciadorDeEquipes.getInstance();

    equipes.adicionar(New Equipe("Equipe 1"));
    equipes2.adicionar(New Equipe("Equipe 2"));
}
```

Exercícios

```java
public class FuncionarioService {
    private SMSNotificador notificador = new SMSNotificador();

    public void calcularSalario(String tipo, double valorBase) {
        if (tipo.equals("CLT")) {
            System.out.println("Salário CLT: " + (valorBase - 0.1 * valorBase));
        } else if (tipo.equals("PJ")) {
            System.out.println("Salário PJ: " + valorBase);
        } else {
            System.out.println("Tipo desconhecido");
        }
    }

    public void gerarRelatorio(Funcionario f) {
        System.out.println("Relatório: " + f.getNome() + " - " + f.getCargo());
    }

    public void enviarNotificacaoSMS(String numero, String mensagem) {
        notificador.enviarSMS(numero, mensagem);
    }
```

Exercício

A) Liste quais princípios SOLID estão sendo violados no código acima e por quê.

B) Refatore o código, aplicando boas práticas e os princípios SOLID.

Contexto adicional: A empresa deseja, futuramente, calcular salários de estagiários e freelancers com regras específicas e também permitir que o envio de notificações possa ser feito também por e-mail. Além de que os relatórios poderão ser exportados em diferentes formatos, como PDF e CSV.

```
public class FuncionarioService {
    private SMSNotificador notificador = new SMSNotificador();

    public void calcularSalario(String tipo, double valorBase) {
        if (tipo.equals("CLT")) {
            System.out.println("Salário CLT: " + (valorBase - 0.1 * valorBase));
        } else if (tipo.equals("PJ")) {
            System.out.println("Salário PJ: " + valorBase);
        } else {
            System.out.println("Tipo desconhecido");
        }
    }

    public void gerarRelatorio(Funcionario f) {
        System.out.println("Relatório: " + f.getNome() + " - " + f.getCargo());
    }

    public void enviarNotificacaoSMS(String numero, String mensagem) {
        notificador.enviarSMS(numero, mensagem);
    }
```

DIP

```java
public class FuncionarioService {
    private SMSNotificador notificador = new SMSNotificador();

    public void calcularSalario(String tipo, double valorBase) {
        if (tipo.equals("CLT")) {
            System.out.println("Salário CLT: " + (valorBase - 0.1 * valorBase));
        } else if (tipo.equals("PJ")) {
            System.out.println("Salário PJ: " + valorBase);
        } else {
            System.out.println("Tipo desconhecido");
        }
    }

    public void gerarRelatorio(Funcionario f) {
        System.out.println("Relatório: " + f.getNome() + " - " + f.getCargo());
    }

    public void enviarNotificacaoSMS(String numero, String mensagem) {
        notificador.enviarSMS(numero, mensagem);
    }
```

SRP e OCP

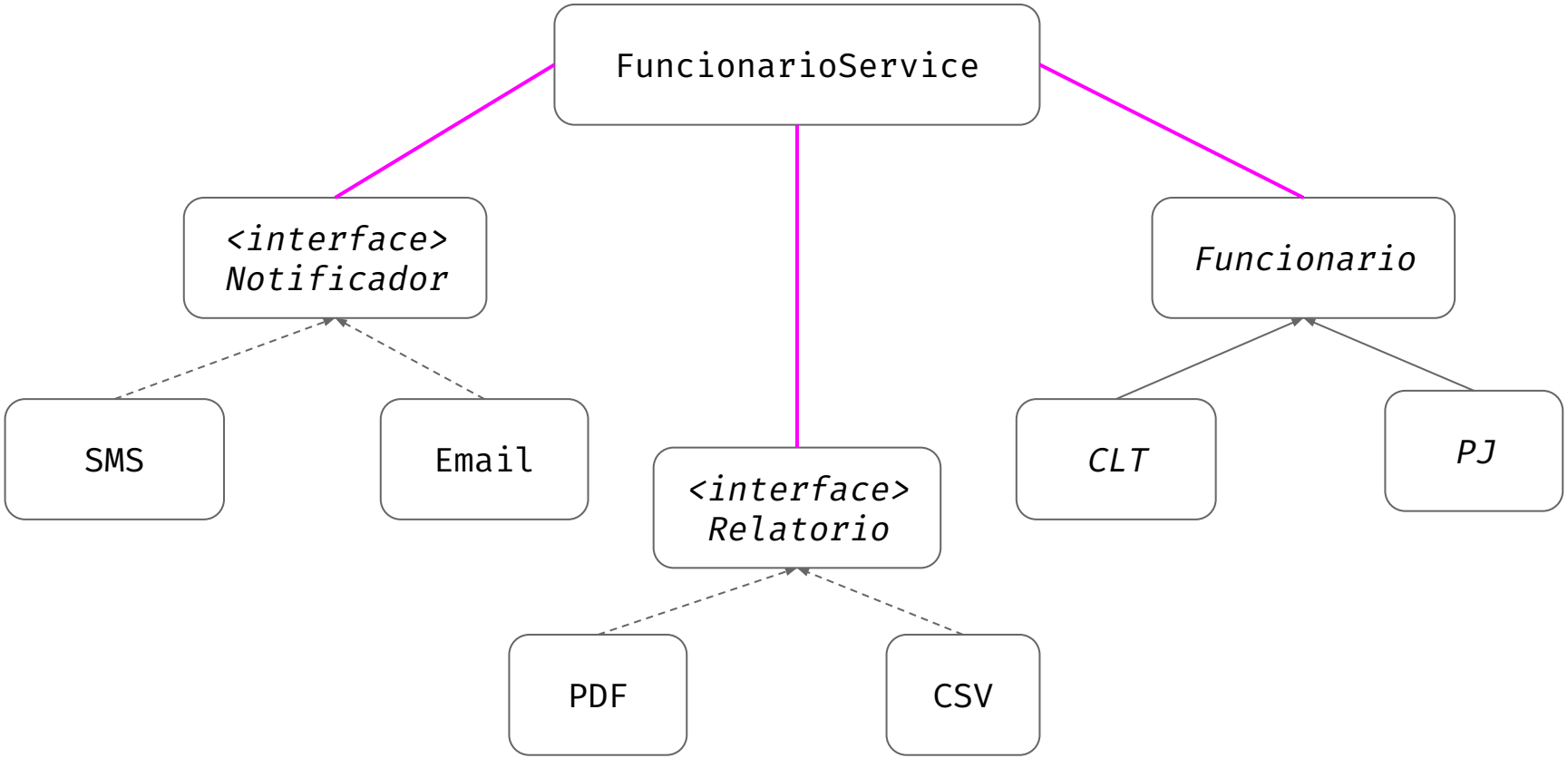
```java
public class FuncionarioService {
    private SMSNotificador notificador = new SMSNotificador();

    public void calcularSalario(String tipo, double valorBase) {
        if (tipo.equals("CLT")) {
            System.out.println("Salário CLT: " + (valorBase - 0.1 * valorBase));
        } else if (tipo.equals("PJ")) {
            System.out.println("Salário PJ: " + valorBase);
        } else {
            System.out.println("Tipo desconhecido");
        }
    }

    public void gerarRelatorio(Funcionario f) {
        System.out.println("Relatório: " + f.getNome() + " - " + f.getCargo());
    }

    public void enviarNotificacaoSMS(String numero, String mensagem) {
        notificador.enviarSMS(numero, mensagem);
    }
```

SRP

```java
public class FuncionarioService {
    private SMSNotificador notificador = new SMSNotificador();

    public void calcularSalario(String tipo, double valorBase) {
        if (tipo.equals("CLT")) {
            System.out.println("Salário CLT: " + (valorBase - 0.1 * valorBase));
        } else if (tipo.equals("PJ")) {
            System.out.println("Salário PJ: " + valorBase);
        } else {
            System.out.println("Tipo desconhecido");
        }
    }

    public void gerarRelatorio(Funcionario f) {
        System.out.println("Relatório: " + f.getNome() + " - " + f.getCargo());
    }

    public void enviarNotificacaoSMS(String numero, String mensagem) {
        notificador.enviarSMS(numero, mensagem);
    }
```

SRP

Exercício


FuncionarioService
<interface>
Notificador
Funcionario
SMS
Email
<interface>
Relatorio
CLT
PJ
PDF
CSV

```
public class FuncionarioService {
    private Relatorio relatorioService;
    private Notificador notificacaoService;

    public FuncionarioService(RelatorioService r, NotificacaoService n) {
        this.relatorioService = r;
        this.notificacaoService = n;
    }

    public void gerarRelatorio(Funcionario f) {
        relatorioService.gerar(f);
    }

    public void enviarNotificacao(Funcionario f, String mensagem) {
        String contato = f.getContato();
        notificacaoService.enviar(contato, mensagem);
    }
}
```

```
public class FuncionarioService {
    private RelatorioService relatorioService;
    private Notificador notificacaoService;

    public FuncionarioService(RelatorioService r, Notificador n) {
        this.relatorioService = r;
        this.notificacaoService = n;
    }

    public void gerarRelatorio(Funcionario f) {
        relatorioService.gerar(f);
    }

    public void enviarNotificacao(Funcionario f, String mensagem) {
        notificacaoService.notificarFuncionario(f, mensagem);
    }
}
```

Não são duas responsabilidades?

Não. Padrão de Projeto Fachada.

Exercício

Implemente uma aplicação em Java que simule a fila de atendimento de uma clínica médica, respeitando a ordem de chegada dos pacientes (sem prioridade). Utilize a classe LinkedList para armazenar os pacientes. Implemente duas funcionalidades:

Adicionar um novo paciente na fila.

Atender o próximo paciente.

```java
public class FilaDeAtendimento {

private Queue<Paciente> filaPacientes;

public FilaDeAtendimento() {
        this.filaPacientes = new LinkedList<>();
}
public void adicionarPaciente(Paciente paciente) {
        filaPacientes.offer(paciente);
}

public Paciente atenderProximoPaciente() {
        Paciente pacienteAtendido = filaPacientes.poll();
        return pacienteAtendido;
}
```

```java
public class Paciente {
    private String nome;
    private String id;
}
```

```java
FilaDeAtendimento fila = new FilaDeAtendimento();

fila.adicionarPaciente(new Paciente("João Silva", "111"));
fila.adicionarPaciente(new Paciente("Maria Oliveira", "222"));
fila.adicionarPaciente(new Paciente("Pedro Costa", "333"));

Paciente p = fila.atenderProximoPaciente();
Paciente p = fila.atenderProximoPaciente();
```

Dúvidas?