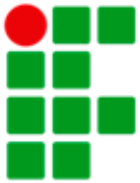


INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Bahia

Linguagem de Programação II

Introdução aos Padrões de Projeto (Design Patterns) - Singleton;

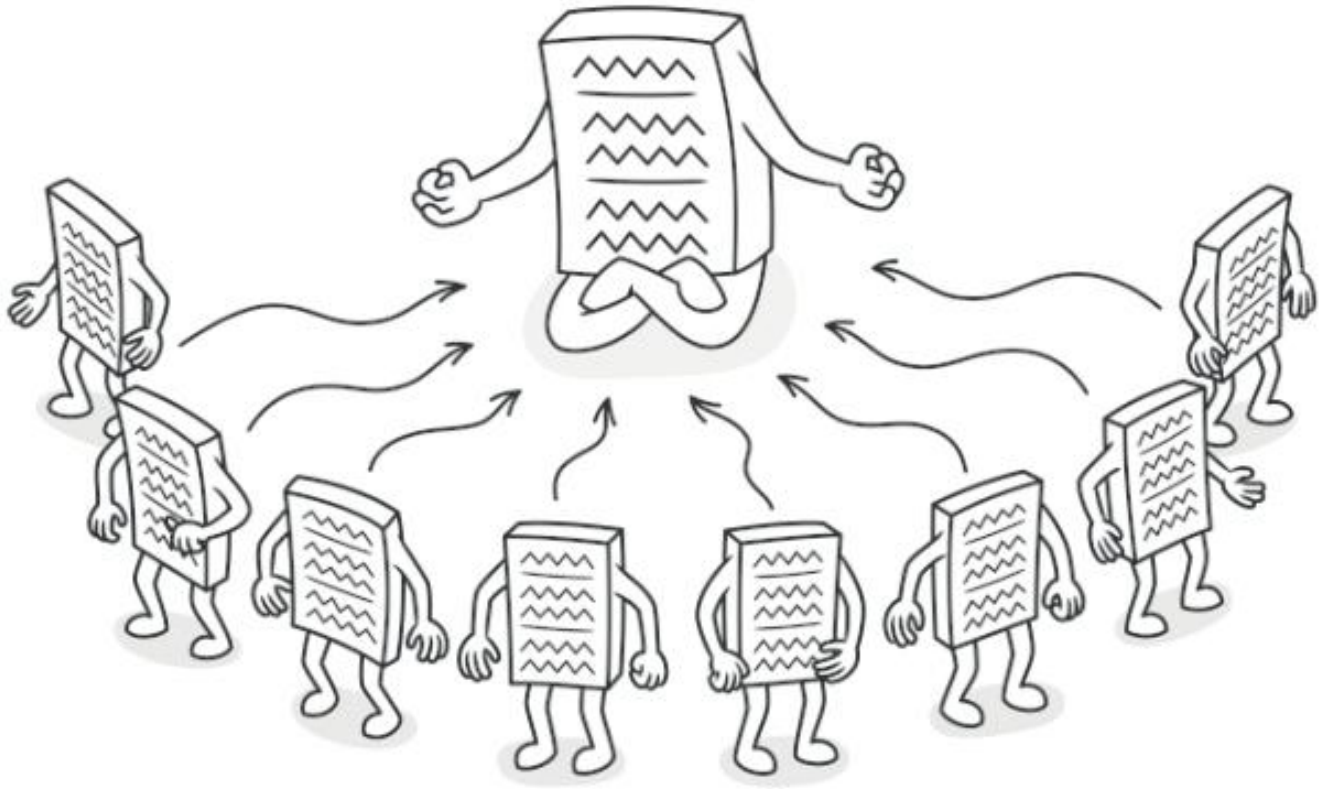
Roteiro

- Introdução aos Padrões de Projeto (Design Patterns) - Singleton:
  - Propósito;
  - Problema;
  - Solução;
  - Analogia com o Mundo Real;
  - Estrutura;
  - Como implementar;
  - Código de Exemplo;
  - Aplicabilidade;
  - Prós e Contras;
  - Relações com Outros Padrões.

Singleton – Propósito

O **Singleton** é um padrão de projeto criacional que permite garantir que uma classe tenha apenas uma instância (objeto), enquanto provê um ponto de acesso global para essa instância.

Singleton – Propósito



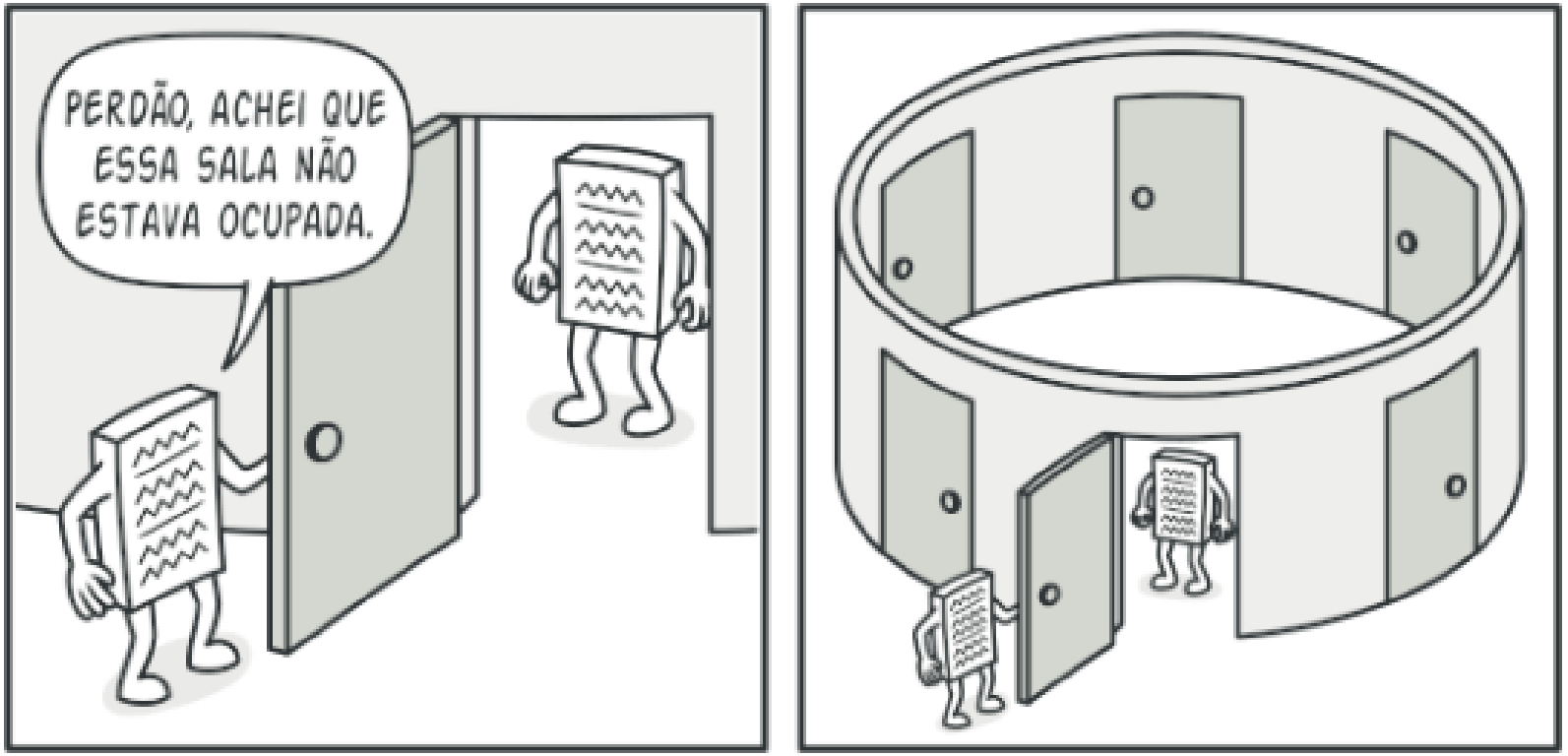
Singleton – Problema

O padrão **Singleton** resolve dois problemas de uma só vez, violando o *princípio de responsabilidade única*:

Singleton – Problema

- **Garantir que uma classe tenha apenas uma única instância:** Serve para **controlar** o **acesso** a algum **recurso compartilhado** (base de dados, arquivos...).

Singleton – Problema



Clientes podem não se dar conta que estão lidando com o mesmo objeto a todo momento.

Singleton – Problema

- **Fornece um ponto de acesso global para aquela instância:** Permite que você acesse algum objeto de qualquer lugar no programa. Contudo, ele também **protege** aquela **instância** de ser **sobrescrita** por outro código.

Singleton – Solução

Todas as implementações do **Singleton** tem esses dois passos em comum:

Singleton – Solução

- Fazer o **construtor padrão privado**, para **prevenir** que outros objetos **usem** o **operador new** com a **classe singleton**.

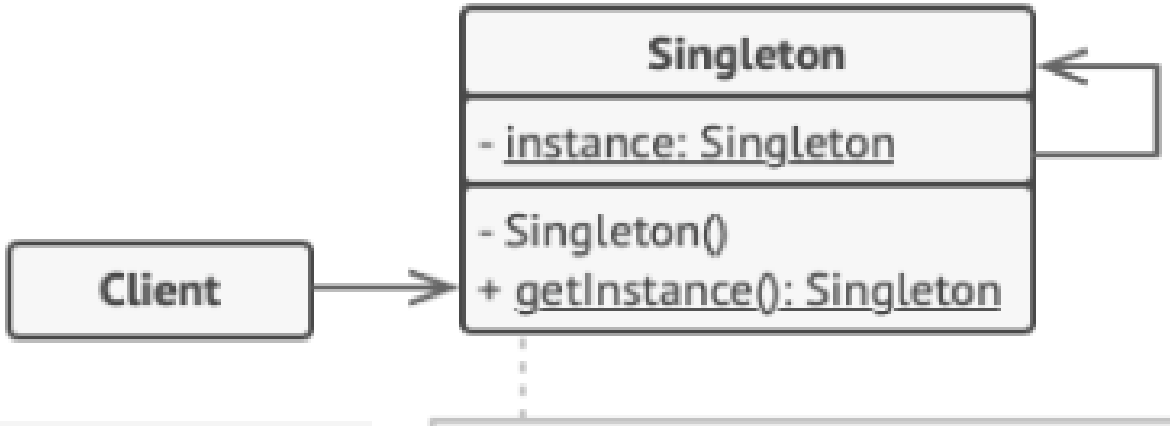
Singleton – Solução

- Criar um **método estático** de **criação** que **age como um construtor**. **Esse método chama o construtor privado** para criar um objeto **e** o **salva em um campo** (atributo) **estático**. As chamadas seguintes para esse método estático retornam o objeto em "cache".

Singleton – Analogia

O Singleton é como ter uma única chave mestra para acessar um cofre compartilhado por todos em uma loja. Mesmo que muitas pessoas peçam acesso, só existe uma chave para abrir o cofre, garantindo que todos acessem o mesmo lugar. Da mesma forma, o Singleton cria uma única instância de uma classe para ser compartilhada por todo o programa.

Singleton – Estrutura



1 A classe **Singleton** declara o método estático `getInstance` que retorna a mesma instância de sua própria classe.

O construtor da singleton deve ser escondido do código cliente. Chamando o método `getInstance` deve ser o único modo de obter o objeto singleton.

Singleton – Como Implementar

1. Adicione um campo (atributo) privado estático na classe para o armazenamento da instância singleton.

Singleton – Como Implementar

2. Declare um método de criação público estático para obter a instância singleton. Esse método deve criar um novo objeto na sua primeira chamada e colocá-lo no campo (atributo) estático. O método deve sempre retornar aquela instância em todas as chamadas subsequentes.

Singleton – Como Implementar

3. Faça o construtor da classe ser privado. O método estático da classe vai ainda ser capaz de chamar o construtor, mas não os demais objetos.

Singleton – Como Implementar

4. Vá para o código cliente e substitua todas as chamadas diretas para o construtor do singleton com chamadas para seu método de criação estático.

Singleton – Código de Exemplo

```java
package SingletonConceitual;

public class Singleton {
    private static Singleton instance;

    private Singleton() {}

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

    //Outros...
    //...
}
```

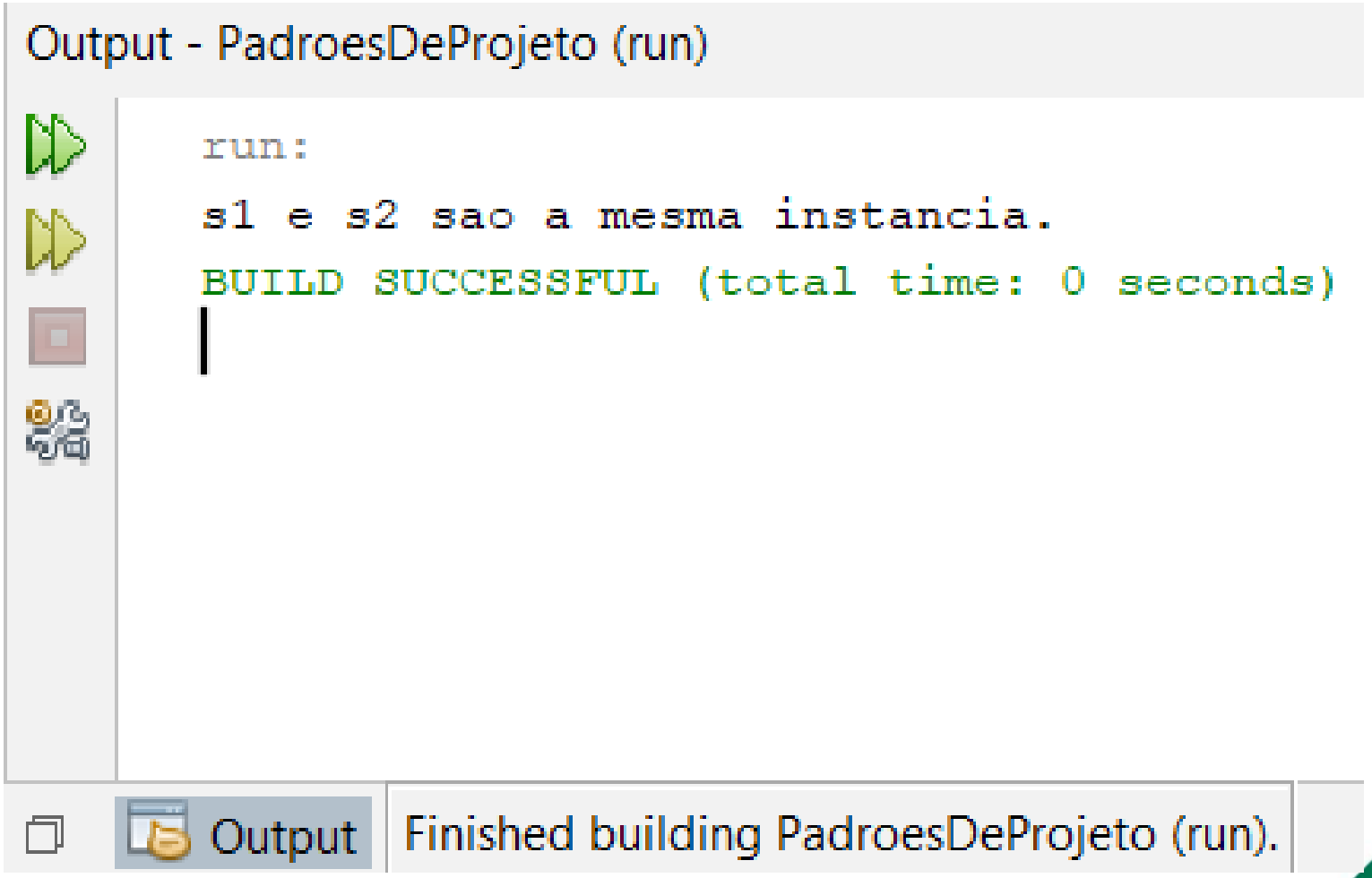
```java
package SingletonConceitual;

public class Singleton {
    private static Singleton instance;

    private Singleton() {}

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

    //Outros...
    //...
}
```

Singleton – Código de Exemplo

```
1   package SingletonConceitual;
2
3   public class Programa {
4
5       public static void main(String[] args) {
6
7           Singleton s1 = Singleton.getInstance();
8           Singleton s2 = Singleton.getInstance();
9
10          if (s1 == s2) {
11              System.out.println("s1 e s2 sao a mesma instancia.");
12          } else {
13              System.out.println("s1 e s2 sao instâncias diferentes.");
14          }
15      }
16  }
```

```java
package SingletonConceitual;

public class Programa {

    public static void main(String[] args) {

        Singleton s1 = Singleton.getInstance();
        Singleton s2 = Singleton.getInstance();

        if (s1 == s2) {
            System.out.println("s1 e s2 sao a mesma instancia.");
        } else {
            System.out.println("s1 e s2 sao instâncias diferentes.");
        }
    }
}
```

Singleton – Código de Exemplo – Exec.



Singleton – Código de Exemplo

```
1   package SingletonReal;
2
3   public class Arquivo {
4       private static Arquivo arquivo;
5
6       private Arquivo() {}
7
8       public static Arquivo obterArquivo() {
9           if (arquivo == null) {
10              arquivo = new Arquivo();
11          }
12          return arquivo;
13      }
14
15      //Outros...
16      //...
17  }
```

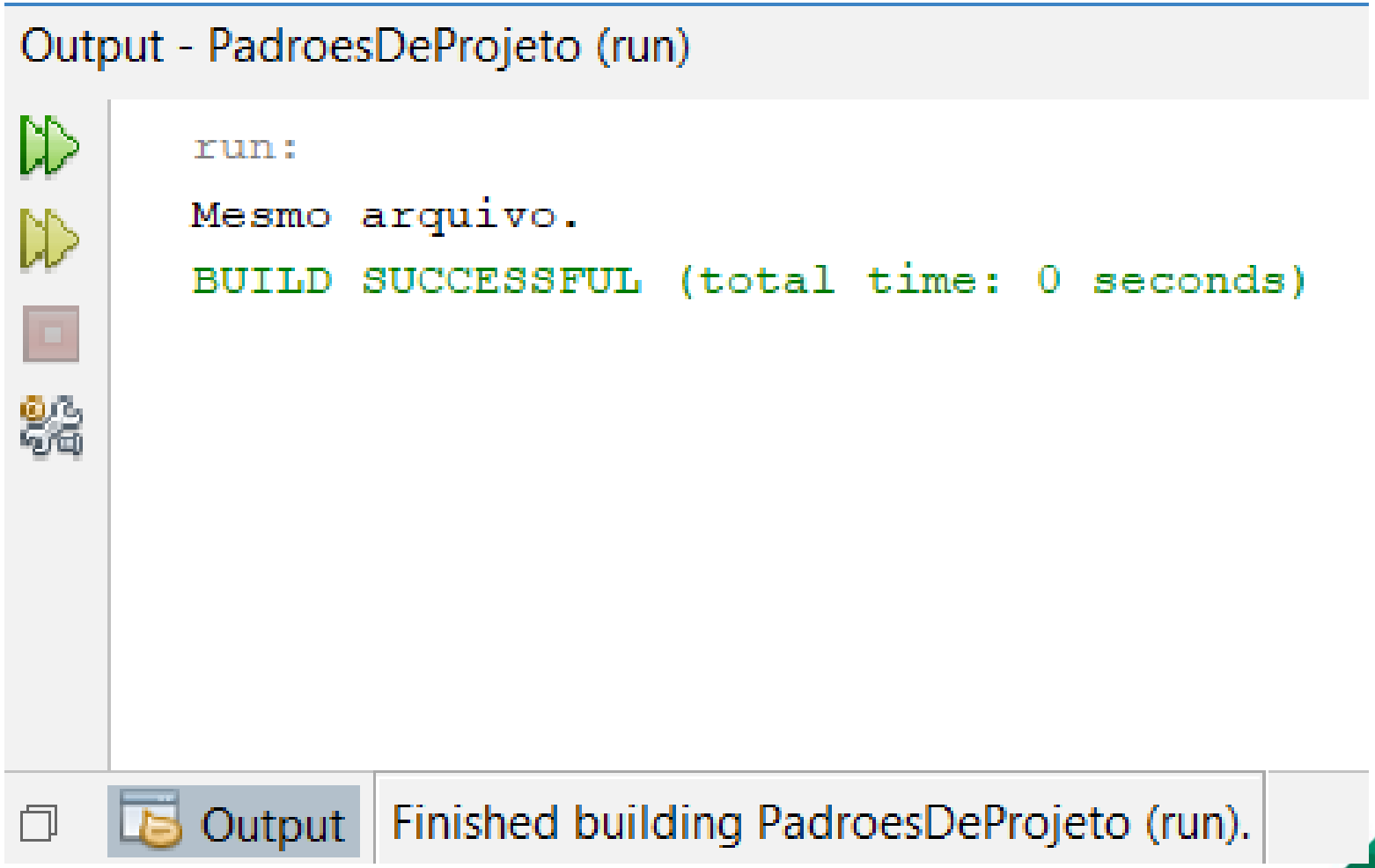
```java
package SingletonReal;

public class Arquivo {
    private static Arquivo arquivo;

    private Arquivo() {}

    public static Arquivo obterArquivo() {
        if (arquivo == null) {
            arquivo = new Arquivo();
        }
        return arquivo;
    }

    //Outros...
    //...
}
```

Singleton – Código de Exemplo

```
1   package SingletonReal;
2
3   public class Programa {
4
5       public static void main(String[] args) {
6
7           Arquivo a1 = Arquivo.obterArquivo();
8           Arquivo a2 = Arquivo.obterArquivo();
9
10          if (a1 == a2) {
11              System.out.println("Mesmo arquivo.");
12          } else {
13              System.out.println("Arquivos diferentes.");
14          }
15
16      }
17  }
```

```java
package SingletonReal;

public class Programa {

    public static void main(String[] args) {

        Arquivo a1 = Arquivo.obterArquivo();
        Arquivo a2 = Arquivo.obterArquivo();

        if (a1 == a2) {
            System.out.println("Mesmo arquivo.");
        } else {
            System.out.println("Arquivos diferentes.");
        }

    }
}
```

Singleton – Código de Exemplo – Exec.



Singleton – Aplicabilidade

- **Utilize** o padrão **Singleton quando** uma **classe** em seu programa **deve ter apenas uma instância** disponível **para todos** seus **clientes**; por exemplo, **um objeto de base de dados único compartilhado** por diferentes partes do programa.

Singleton – Aplicabilidade

- **Utilize** o padrão **Singleton quando** você **precisa** de **um controle mais estrito sobre** as **variáveis globais** do seu programa.

Singleton – Prós

- Certeza de que uma classe só possuirá uma instância;
- Criação de um ponto de acesso global para essa instância;
- A instância só é gerada quando for solicitada pela primeira vez.

Singleton – Contras

- Viola o Princípio da Responsabilidade Única;
- O padrão requer um tratamento especial para programas que utilizam várias threads;
- Dificultam a utilização de testes unitários.

Singleton – Relação com Outros Padrões

Uma classe **fachada** (**Facade**) **pode** frequentemente **ser transformada** em uma **singleton** já que um **único objeto fachada é suficiente** na maioria dos casos.

Exercícios – Pesquise e Responda

Exercícios:

1) **Implemente** o padrão de projeto **Singleton** em uma **classe** chamada **Impressora** para que só seja possível criar uma instância (objeto) a partir dessa classe. Adicionalmente, **crie** uma **classe** chamada **Programa** para **demonstrar que** ao tentar criar duas instâncias (impressoras) na verdade **a classe com Singleton retorna sempre a mesma instância**.

Referências

- VALENTE, Marco Tulio. Engenharia de software moderna. Princípios e Práticas para Desenvolvimento de Software com Produtividade, v. 1, 2020.

- https://www.alura.com.br/artigos/design-patterns-introducao-padroes-projeto

- https://refactoring.guru/pt-br/design-patterns

Obrigado!

- Canais de Comunicação;
- Horário de Atendimento.