

Singleton

Elementos Essenciais

- **Nome:** *Singleton*
- **Problema:** Perder informações acerca de objetos criados por conta da perda de referência a esses objetos.
- **Solução:** Garantir que uma classe tenha somente uma instância e fornece um ponto global de acesso para a mesma.
- **Consequências:** Permite o controle de como e quando os clientes acessam a instância.

Problema

- Imaginemos uma classe de acumulação implementada como abaixo:


Note que o programador acabou criando uma nova instância de `Acumuladora` (bug mesmo). Isso poderia ter sido evitado se `Acumuladora` só pudesse ser criada uma única vez, ou seja, se fosse um objeto único.

```
public class Acumuladora {  
    private int valor;  
  
    public Acumuladora() {  
        valor = 0;  
    }  
  
    public void incrementarValor(int acrescimo){  
        valor += acrescimo;  
    }  
  
    public int getValor(){  
        return valor;  
    }  
}
```

```
package singleton;  
public class Singleton {  
  
    public static void outroMetodo(Acumuladora a){  
        a = new Acumuladora();  
        a.incrementarValor(50);  
        a.incrementarValor(10);  
    }  
  
    public static void main(String[] args) {  
        Acumuladora a = new Acumuladora();  
        a.incrementarValor(10);  
        a.incrementarValor(20);  
        System.out.println("O valor é " + a.getValor());  
  
        // Suponha aqui diversos outros códigos de outras classes ...  
  
        Singleton.outroMetodo(a);  
        System.out.println("O valor é " + a.getValor());  
    }  
}
```

Acumuladora vai virar Singleton

```
public class Acumuladora {  
    private int valor;  
  
    private static Acumuladora instancia;  
    public static Acumuladora getInstancia() {  
        if (instancia == null)  
            instancia = new Acumuladora();  
        return instancia;  
    }  
    private Acumuladora() {  
        valor = 0;  
    }  
    public void incrementarValor(int acrescimo) {  
        valor += acrescimo;  
    }  
    public int getValor() {  
        return valor;  
    }  
}
```



```
public class Acumuladora {  
    private int valor;  
  
    public Acumuladora() {  
        valor = 0;  
    }  
  
    public void incrementarValor(int acrescimo) {  
        valor += acrescimo;  
    }  
  
    public int getValor() {  
        return valor;  
    }  
}
```

Erros de compilação

```
package singleton;
public class Singleton {

    public static void outroMetodo(Acumuladora a){
        a = new Acumuladora();
        a.incrementarValor(50);
        a.incrementarValor(10);
    }

    public static void main(String[] args) {
        Acumuladora a = new Acumuladora();
        a.incrementarValor(10);
        a.incrementarValor(20);
        System.out.println("O valor é " + a.getValor());

        // Suponha aqui diversos outros códigos de outras classes ...

        Singleton.outroMetodo(a);
        System.out.println("O valor é " + a.getValor());
    }
}
```

Resolvendo o problema de compilação – mudando o comportamento

```
package singleton;
public class Singleton {

    public static void outroMetodo(Acumuladora a){
        a = Acumuladora.getInstancia();
        a.incrementarValor(50);
        a.incrementarValor(10);
    }

    public static void main(String[] args) {
        Acumuladora a = Acumuladora.getInstancia();
        a.incrementarValor(10);
        a.incrementarValor(20);
        System.out.println("O valor é " + a.getValor());

        // Suponha aqui diversos outros códigos de outras classes ...

        Singleton.outroMetodo(a);
        System.out.println("O valor é " + a.getValor());
    }
}
```

Resultado:

```
run:
O valor é 30
O valor é 90
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Antes estava:

```
run:
O valor é 30
O valor é 30
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Dúvidas?

