Instituto Federal Catarinense *(Campus Blumenau)*

Professor: Ricardo de La Rocha Ladeira

Matéria: Padrões de Projeto

Nomes: Gabrielli Danker

Turma: BCC 2025.1

Data de entrega: 06 de Março de 2025

Exercícios

1) Coloque as informações do padrão Singleton no formato visto na aula 1.

**Nome:** Singleton

**Descrição:** Criação de instâncias de uma classe a apenas uma única instância e fornece um ponto global de acesso a essa instância

**Descrição do problema:** Em alguns cenários, como gerenciadores de configuração ou conexões com banco de dados, precisamos garantir que apenas uma instância da classe seja criada para evitar inconsistências e desperdício de recursos.

**Descrição da solução:** O Singleton cria uma instância única da classe e impede que novas instâncias sejam criadas, geralmente utilizando um atributo estático privado e um método público para fornecer acesso a essa instância.

**Consequências:**

* Garante uma única instância
* Economiza memória
* Facilita o controle de acesso a recursos compartilhados.
* Pode introduzir um ponto único de falha, tornar o código mais difícil de testar e, em alguns casos, criar problemas em aplicações multithread.

2) Implemente o padrão Singleton em outra linguagem, utilizando um exemplo livre..

class Singleton:

\_instance = None #Atributo estático para armazenar a única iinstância

def \_\_new\_\_(cls):

if cls.\_instance is None:

cls.\_instance = super(Singleton, cls).\_\_new\_\_(cls)

return cls.\_instance

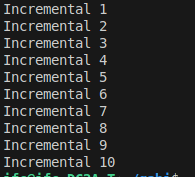
obj1 = Singleton()

obj2 = Singleton()

print(obj1 is obj2)

3) (adaptada de GOMES, s.d.) Compile e execute o programa a seguir. Depois, altere sua implementação para que a classe Incremental seja *Singleton*. Execute novamente e veja os resultados.

| class Incremental {  private static int count = 0;  private int numero;  public Incremental() {  numero = ++count;  }  public String toString() {  return "Incremental " + numero;  }  }  public class TesteIncremental {  public static void main(String[] a) {  for (int i = 0; i < 10; i++) {  Incremental inc = new Incremental();  System.out.println(inc);  }  }  } |
| --- |



class Incremental {

private static Incremental instance; // Única instancia da classe

private static int count = 0;

private int numero;

private Incremental() {

numero = ++count;

}

public static Incremental getInstance() {

if (instance == null){

instance = new Incremental();

}

return instance;

}

public String toString() {

return "Incremental " + numero;

}

}

public class TesteIncremental {

public static void main(String[] a) {

for (int i = 0; i < 10; i++) {

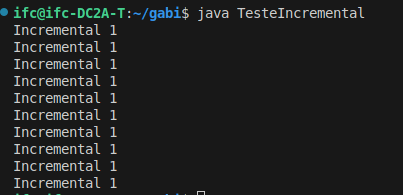
Incremental inc = Incremental.getInstance();

System.out.println(inc);

}

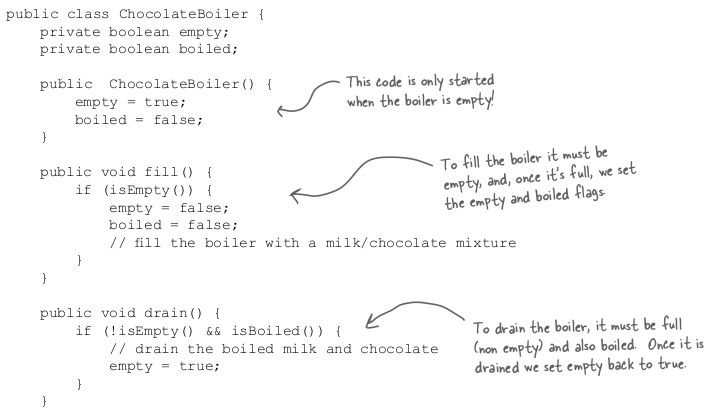
}

}



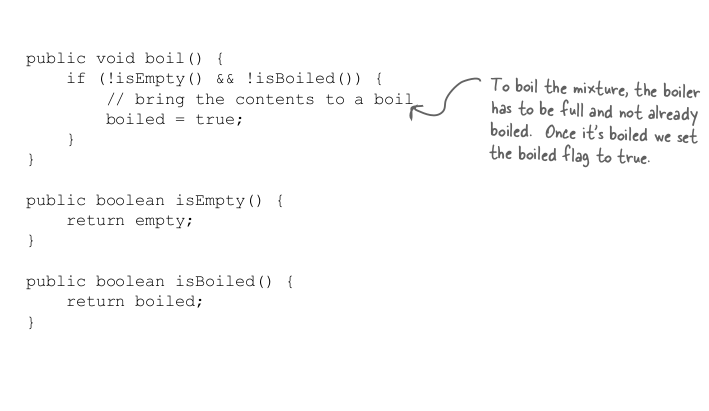
4) Observe as Figuras 2 e 3 (GOMES, s.d.) e preencha a Figura 4 (GOMES, s.d.).

**Figura 2**. Exemplo de classe ChocolateBoiler.



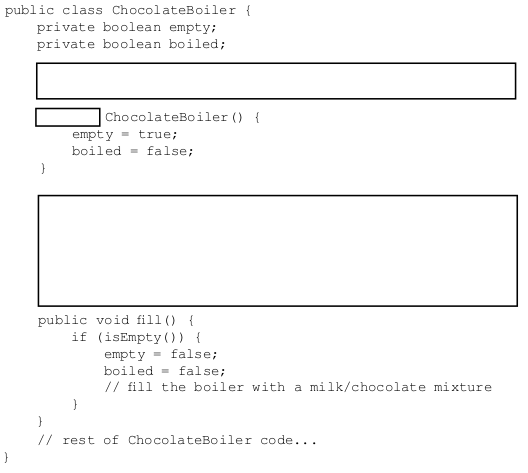
Fonte: GOMES, s.d.

**Figura 3**. Continuação do exemplo da classe ChocolateBoiler.



Fonte: GOMES, s.d.

**Figura 4**. Exercício.



Fonte: GOMES, s.d.

public class ChocolateBoiler {

private boolean empty;

private boolean boiled;

private static ChocolateBoiler instance;

private ChocolateBoiler(){

empty = true;

boiled = false;

}

public static ChocolateBoiler getInstance(){

if (instance == null){

instance = new ChocolateBoiler();

}

return instance;

}

public void fill(){

if (isEmpty()) {

empty = false;

boiled = false;

}

}

public void boil(){

if (!isEmpty( ) && !isBoiled()){

boiled = true;

}

}

public void drain(){

if (!isEmpty( ) && !isBoiled()) {

empty = true;

}

}

public boolean isEmpty(){

return empty;

}

public boolean isBoiled(){

return boiled;

}

}