



# Cachaça Home

Alunos: Anael Jonas Alencar Andrade  
Felipe Correia Duarte Batista  
Francisco Luan Ferreira Brito



01

# DESCRIÇÃO DO PROBLEMA E USO DOS PADRÕES

# Descrição do problema

**Cachaça Home** é uma famosa empresa que gerencia uma enorme franquia de bares, seus bares dominam em cada região que são instalados, e o que os levaram a esse grande sucesso? Esse sucesso foi alcançado graças ao diferencial da empresa que tem a proposta de criar bares temáticos de acordo com estilo musical, assim atraindo pessoas que se identificam com um estilo musical específico.

A empresa criou um certo padrão na construção dos seus bares, onde eles têm a seguinte base de construção: Ela possui construtoras especializadas para cada tipo de bar, construtora x, apenas constrói bares do tipo x, construtora y, constrói bares do tipo y.

Os bares possuem uma estrutura básica, mas podem se expandir para ter mais opções acrescidas a ele, como palcos, +Seguranças (conjunto: câmeras, alarmes, seguranças) , sinuca, fliperama, Telão entre outros.

Para gerenciar as informações sobre dias de fechamentos, promoções ou eventos, o gerente geral envia uma mensagem que pode ser vista por todos os bares da franquia evitando a necessidade de avisar uma por uma.

# Compreensão do uso dos padrões

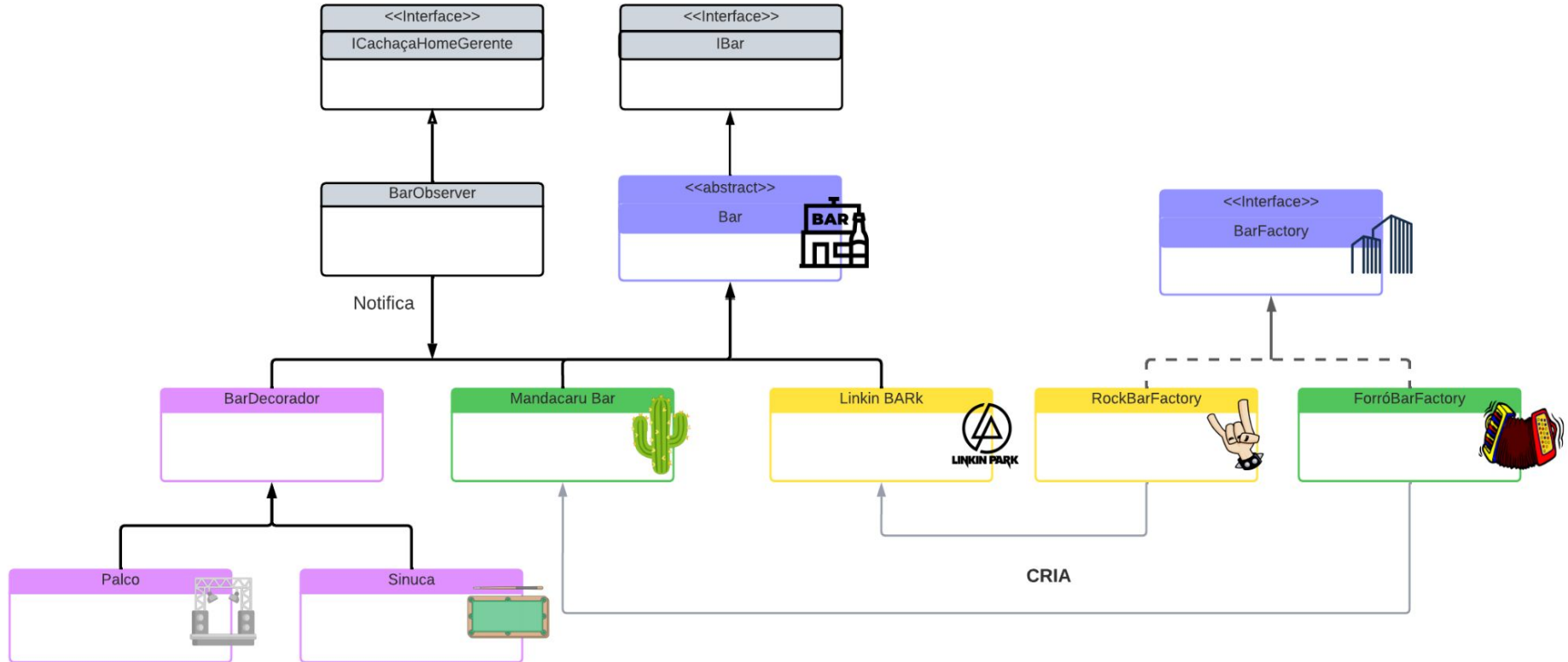
- **Factory Method:** Criar a estrutura padrão do bar (Bares de Rock, Forró, Reggae, etc).
- **Decorator:** Decorações extras do bar (seguranças, palco, alarmes, ar-condicionado, TV 's, etc).
- **Observer:** O Gerente avisa aos bares os feriados(dias que o os bares não devem funcionar) como por exemplo Natal(25).



02

DIAGRAMA

# DIAGRAMA





03

# ORGANIZAÇÃO E CÓDIGO

# Organização & Código

01

Bar  
(estrutura base)

03

Factory Method

02






Decorator

04

Observer

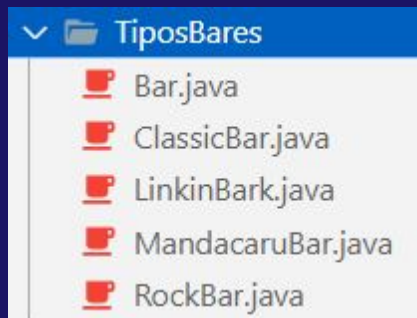
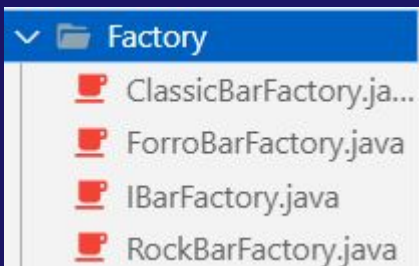
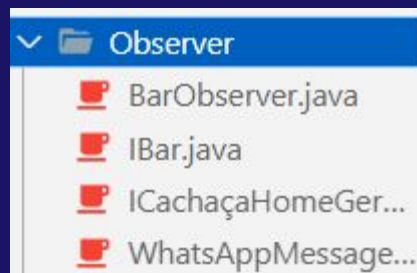
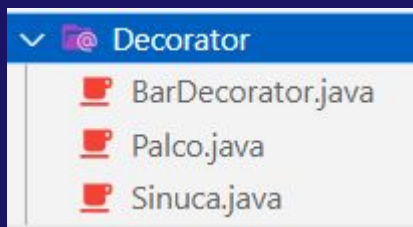


# Organização

- >  Decorator
- >  Factory
- >  Observer
- >  TiposBares
-  CachaçaHome.java



# Organização



CachaçaHome.java

# Organização

```
9 public class CachaçaHome {  
    Run | Debug  
10     public static void main(String[] args) {  
11         // Factory.  
12         RockBarFactory rock_bar_factory = new RockBarFactory();  
13         ClassicBarFactory classic_bar_factory = new ClassicBarFactory();  
14         Bar rocks_bar = rock_bar_factory.create(c: RockBar.class);  
15         Bar classic_bar = classic_bar_factory.create(c: ClassicBar.class);  
16  
17         // Decorator.  
18         classic_bar = new Palco(classic_bar);  
19         classic_bar = new Sinuca(classic_bar);  
20         rocks_bar = new Sinuca(rocks_bar);  
21  
22         // Observer.  
23         BarObserver bar_observer = new BarObserver();  
24         bar_observer.register(classic_bar);  
25         bar_observer.register(rocks_bar);  
26         bar_observer.remove(rocks_bar);  
27         WhatsAppMessage message = new WhatsAppMessage();  
28         message.setMessage(message: "Feriado dia 14 de dezembro!");  
29         bar_observer.setMessage(message);  
30         bar_observer.sendMessage();  
31  
32         // Informações.  
33         System.out.println(x: "\n");  
34         System.out.println(classic_bar.getBarInformations());  
35         System.out.println(rocks_bar.getBarInformations());  
36         System.out.println(x: "\n");  
37     }  
38 }
```

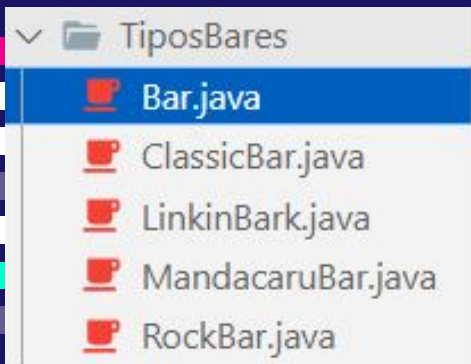


CachaçaHome.java

# DECORATOR

The background is a solid dark blue. It is decorated with several horizontal rectangles of different colors and sizes. In the top right, there is a bright cyan rectangle and a white rectangle. On the left side, there is a magenta rectangle, a white rectangle, and a dark blue rectangle. On the right side, there is a small cyan rectangle, a dark blue rectangle, and a white rectangle. At the bottom, there is a white rectangle, a dark blue rectangle, a magenta rectangle, and a large cyan rectangle.

# ESTRUTURA BASE



```
package TiposBares;

import Observer.IBar;
import Observer.WhatsAppMessage;

public abstract class Bar implements IBar {
    protected String name;
    protected String type;
    protected double budget;
    protected String decorations;
    protected String whatsapp;

    public Bar() {
        decorations = "";
        whatsapp = "";
    }
}
```

# ESTRUTURA BASE

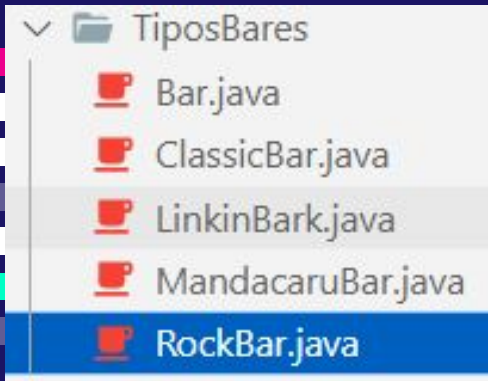
```
public String getName() {  
    return name;  
}  
  
public void setName(String name) {  
    this.name = name;  
}  
  
public String getType() {  
    return type;  
}  
  
public void setType(String type) {  
    this.type = type;  
}
```

```
public double getBudget() {  
    return budget;  
}  
  
public void setBudget(double budget) {  
    this.budget = budget;  
}  
  
public String getDecorations() {  
    return decorations;  
}  
  
public void setDecorations(String decorations) {  
    this.decorations = decorations;  
}
```

# ESTRUTURA BASE

```
public String getBarInformations() {  
    String retorno = String.format(format: "%s | tipo %s | decorações: %s | orçamento: R$  
%.2f | whatsapp: %s \n ", getName(), getType(), getDecorations(), getBudget(),  
    whatsapp);  
    return retorno;  
}  
  
public void update(WhatsAppMessage message) {  
    this.whatsapp = this.whatsapp + message.getMessage() + ", ";  
}  
}
```

# ESTRUTURA BASE

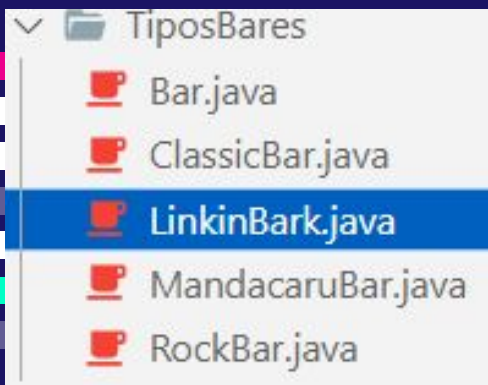


```
package TiposBares;

public class RockBar extends Bar {
    public RockBar() {
        setName(name: "Bar de Rock");
        setType(type: "RockBar");
        setBudget(budget: 50000.00);
    }
}
```



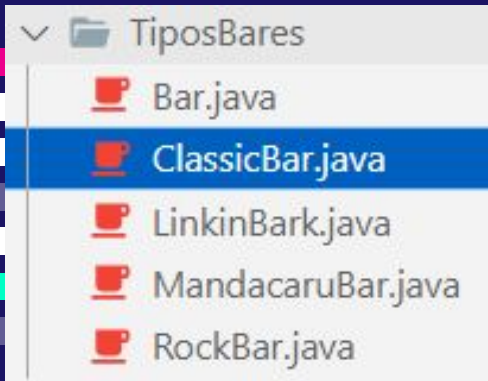
# ESTRUTURA BASE



```
package TiposBares;

public class LinkinBark extends Bar {
    public LinkinBark() {
        setName(name: "Linkin Bark");
        setType(type: "RockBar");
        setBudget(budget: 60000.00);
    }
}
```

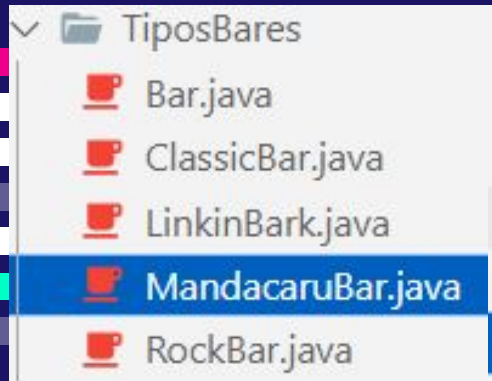
# ESTRUTURA BASE



```
package TiposBares;

public class ClassicBar extends Bar {
    public ClassicBar() {
        setName(name: "Bar Clássico");
        setType(type: "ClassicBar");
        setBudget(budget: 200000.98);
    }
}
```

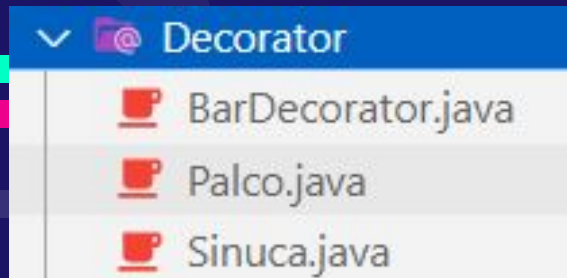
# ESTRUTURA BASE



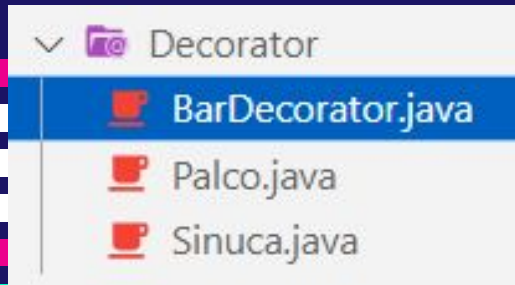
```
package TiposBares;
```

```
public class MandacaruBar extends Bar{  
    public MandacaruBar() {  
        setName(name: "Mandacaru Bar");  
        setType(type: "Forró");  
        setBudget(budget: 40000.00);  
    }  
}
```

# Decorator



# DECORATOR



```
package Decorator;

import TiposBares.Bar;

public abstract class BarDecorator extends Bar {
    protected Bar bar_decorado;

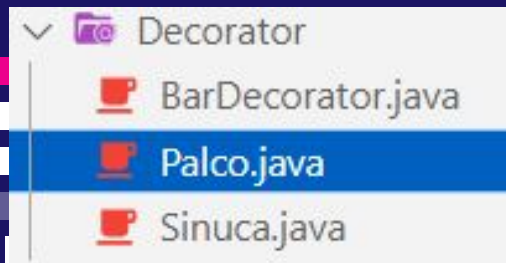
    public BarDecorator(Bar bar_decorado) {
        this.bar_decorado = bar_decorado;
    }

    public Bar getBar() {
        return bar_decorado;
    }

    @Override
    public double getBudget() {
        return this.budget + bar_decorado.getBudget();
    }

    @Override
    public String getDecorations() {
        return bar_decorado.getDecorations() + this.decorations;
    }
}
```

# DECORATOR

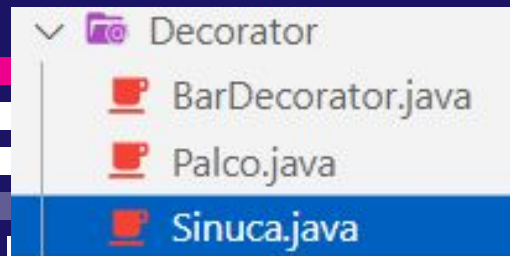


```
package Decorator;

import TiposBares.Bar;

public class Palco extends BarDecorator {
    public Palco(Bar bar) {
        super(bar);
        setName(bar.getName());
        setType(bar.getType());
        setDecorations(decorations: "Palco, ");
        setBudget(budget: 12538.92);
    }
}
```

# DECORATOR



```
package Decorator;

import TiposBares.Bar;

public class Sinuca extends BarDecorator {
    public Sinuca(Bar bar) {
        super(bar);
        setName(bar.getName());
        setType(bar.getType());
        setDecorations(decorations: "Sinuca, ");
        setBudget(budget: 2000);
    }
}
```

# FACTORY METHOD



# FACTORY METHOD

Factory

ClassicBarFactory.java...

ForroBarFactory.java

IBarFactory.java

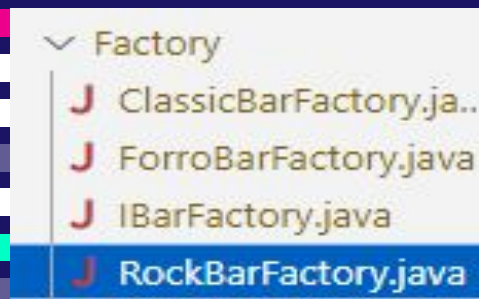
RockBarFactory.java

IBarFactory.java 1 ●

Factory > IBarFactory.java > ...

```
1 package Factory;
2
3 import TiposBares.Bar;
4
5 public interface IBarFactory {
6     Bar create(Class c);
7 }
8
```

# FACTORY METHOD



```
J IBarFactory.java 1 ● J RockBarFactory.java 1 X
Factory > J RockBarFactory.java > {} Factory
1 package Factory;
2
3 import TiposBares.*;
4
5 public class RockBarFactory implements IBarFactory {
6     public Bar create(Class c) {
7         if (c == RockBar.class)
8             return new RockBar();
9         else if (c == LinkinBark.class)
10            return new LinkinBark();
11        return null;
12    }
13 }
```

# FACTORY METHOD

## ForróBar

```
IBarFactory.java 1 • J ForróBarFactory.java 1 •
factory > J ForróBarFactory.java > ...
1  package Factory;
2
3  import TiposBares.Bar;
4  import TiposBares.MandacaruBar;
5
6  public class ForróBarFactory implements IBarFactory {
7      public Bar create(Class c) {
8          if (c == MandacaruBar.class)
9              return new MandacaruBar();
10         return null;
11     }
12 }
```

## ClassicBar

```
ClassicBarFactory.java 1 •
factory > J ClassicBarFactory.java > ...
1  package Factory;
2
3  import TiposBares.*;
4
5  public class ClassicBarFactory implements IBarFactory {
6      public Bar create(Class c) {
7          if (c == ClassicBar.class) return new ClassicBar();
8          return null;
9      }
10 }
```



OBSERVER

# OBSERVER

## Observer

BarObserver.java

IBar.java

ICachaçaHomeGerente.java

WhatsAppMessage.java

ICachaçaHomeGerente.java

Observer > ICachaçaHomeGerente.java > {} Observer

```
1 package observer;
2
3 public interface ICachaçaHomeGerente {
4     void register(IBar bar);
5     void remove(IBar bar);
6     void sendMessage();
7 }
8
```

# OBSERVER

## Observer

BarObserver.java

IBar.java

ICachaçaHomeGerente.java

WhatsAppMessage.java

IBar.java

Observer > IBar.java > ...

```
1 package _observer;  
2  
3 public interface IBar {  
4     void update(WhatsAppMessage message);  
5 }  
6
```

# OBSERVER

## Observer

BarObserver.java

IBar.java

ICachaçaHomeGerente.java

WhatsAppMessage.java

BarObserver.java

```
Observer > J BarObserver.java
1  package _observer;
2
3  import java.util.ArrayList;
4  import java.util.List;
5
6  public class BarObserver implements ICachaçaHomeGerente {
7      private List<IBar> bares = new ArrayList<>();
8      private WhatsAppMessage message;
9
10     @Override
11     public void register(IBar bar) {
12         bares.add(bar);
13     }
14
15     @Override
16     public void remove(IBar bar) {
17         bares.remove(bar);
18     }
19
20     @Override
21     public void sendMessage() {
22         for (IBar b : bares) {
23             b.update(message);
24         }
25     }
26
27     public void setMessage(WhatsAppMessage message) {
28         this.message = message;
29     }
30 }
31
```

# OBSERVER

## Observer

- BarObserver.java
- IBar.java
- ICachaçaHomeGerente.java
- WhatsAppMessage.java

WhatsAppMessage.java

Observer > WhatsAppMessage.java > WhatsAppMessage

```
1 package _observer;
2
3 public class WhatsAppMessage {
4     private String message;
5
6     public void setMessage(String message) {
7         this.message = message;
8     }
9
10    public String getMessage() {
11        return this.message;
12    }
13 }
```





Obrigado!  
Thank You!