

Tarefa03.py > ...

```
1  # Nome: Victor Mariano dos Santos Gomes
2  # RA: 2171392221041
3
4  # Nome: Susana Jesus Reis Alves
5  # RA: 2171392211029
6
7
8  # Implementação do padrão Bridge para tipos de massas de pizza
9
10 # Classe base para massas de pizza
11 class MassaPizza:
12     def preparar(self):
13         pass
14
15     def assar(self):
16         pass
17
18     def cortar(self):
19         pass
20
21     def embalar(self):
22         pass
23
24 # Subclasse representando massa fina
25 class MassaFina(MassaPizza):
26     def preparar(self):
27         return "Preparando massa fina"
28
29     def assar(self):
30         return "Assando em alta temperatura"
31
32     def cortar(self):
33         return "Cortando em pedaços triangulares"
34
35     def embalar(self):
36         return "Embalando na caixa tradicional"
37
38 # Subclasse representando massa grossa
39 class MassaGrossa(MassaPizza):
40     def preparar(self):
41         return "Preparando massa grossa"
```

```

41         return "Preparando massa grossa"
42
43     def assar(self):
44         return "Assando em temperatura média"
45
46     def cortar(self):
47         return "Cortando em pedaços quadrados"
48
49     def embalar(self):
50         return "Embalando na caixa especial"
51
52 # Implementação do padrão Composite para criar pedidos de pizza compostos
53
54 # Classe para criar pedidos de pizza compostos
55 class PedidoPizza:
56     def __init__(self):
57         self.pizzas = [] # Lista para armazenar as pizzas no pedido
58
59     def adicionar_pizza(self, pizza):
60         self.pizzas.append(pizza) # Adiciona uma pizza ao pedido
61
62     def preparar(self):
63         return "\n".join([pizza.preparar() for pizza in self.pizzas]) # Preparação de todas as pizzas
64
65     def assar(self):
66         return "\n".join([pizza.assar() for pizza in self.pizzas]) # Cozimento de todas as pizzas
67
68     def cortar(self):
69         return "\n".join([pizza.cortar() for pizza in self.pizzas]) # Corte de todas as pizzas
70
71     def embalar(self):
72         return "\n".join([pizza.embalar() for pizza in self.pizzas]) # Embalagem de todas as pizzas
73
74 # Implementação do padrão Factory para criar objetos de pizza
75 class FabricaMassaFina(FabricaPizza):
76     def criar_pizza(self):
77         return MassaFina()
78
79 class FabricaMassaGrossa(FabricaPizza):
80     def criar_pizza(self):
81         return MassaGrossa()

```

```
83 # Exemplo de uso com as fábricas específicas
84 fabrica_massa_fina = FabricaMassaFina()
85 fabrica_massa_grossa = FabricaMassaGrossa()
86 pedido = PedidoPizza()
87
88 # Cria pizzas usando as fábricas específicas
89 massa_fina_pizza = fabrica_massa_fina.criar_pizza()
90 massa_grossa_pizza = fabrica_massa_grossa.criar_pizza()
91
92 # Adiciona as pizzas ao pedido
93 pedido.adicionar_pizza(massa_fina_pizza)
94 pedido.adicionar_pizza(massa_grossa_pizza)
95
96 # Imprime as etapas para cada pizza no pedido
97 print("Preparação:")
98 print(pedido.preparar())
99
100 print("\nCozimento:")
101 print(pedido.assar())
102
103 print("\nCorte:")
104 print(pedido.cortar())
105
106 print("\nEmbalagem:")
107 print(pedido.embalar())
108
109
```