Onde cada padrão é utilizado:

* Builder:
  + Classes:
    - ClienteBuilder
  + Implementação:
    - Cliente
  + Test:
* Prototype
  + Classe:
    - Nenhuma
  + Implementação:
    - Pedido, o método pedirNovamente().
  + Test:
* Flyweight:
  + Classes:
    - Endereco, enderecoFactory
  + Implementado em:
    - Cliente
  + Test:
    - deveRetornarTotalEnderecos em clientTest
* Command
  + Classes:
    - ChamadoSuporte, AbrirChamadoSuporte, FecharChamadoSuporte, Tarefa
  + Implementado em:
    - Cliente, executarTarefa e cancelarTarefa
  + Test:
    - ClienteTest, deveAbrirChamadoSuporte, deveFecharChamadoSuporte, deveCancelarFecharChamadoSuporte
* Memento
  + Classes:
    - Nenhuma
  + Implementado em:
    - Cliente, List<Tarefa> tarefas
  + Test:
    - deveCancelarFecharChamadoSuporte
* Iterator
  + Classes:
    - HistoricoDePedidos
  + Implementado em:
    - Cliente, List<Pedido> pedidos, iterator
  + Test:
    - deveContarTotalPedidosHistoricoDePedidos
* TemplateMethod:
  + Classes:
    - Avaliacao, AvaliacaoComImagem, AvaliacaoSemImagem
  + Implementado em:
    - Cliente, avaliarPedido
    - Pedido, receberAvaliacao
  + Test:
    - deveAvaliarPedidoSemImagemESemPassarPelaGerencia
* Mediator
  + Classes:
    - Gerencia
  + Implementado em:
    - Avaliacao , AvaliacaoComImagem, AvaliacaoSemImagem, notificarGerencia.
  + Test:
    - deveAvaliarPedidoComImagem, deveAvaliarPedidoSemImagemPassandoPorGerencia

Estrutural

Falta colocar:

1. Proxy (deixar esse pro final, ele vai ter o esquema do lazy load)
2. Facade (antes de fazer o pedido, verifica se o cliente não tem algum pagamento pendente)
3. Composite (se não implementar esse, implementa o bridge...ou nenhum dos dois fodasse)
4. Adapter (Fingir que existe uma integração externa, por exemplo, o pedido foi feito através do app de parceiro, ou ubereats sei lá)
5. Visitor (talvez esse não seja implementado)
6. Decorator (talvez fazer um esquema tipo colocar um adicional no pedido)
7. Bridge (talvez esse não, ou esse ou o composite)

Já colocados:

* Observer;
* Chain of Responsability;
* State;
* Singleton;
* Strategy;
* Factory
* Prototype
* Memento
* Command
* iterator
* Mediator
* Template method