Onde cada padrão é utilizado:

Obs: os padrões da primeira entrega não serão listados pois já foram validados.

* 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
* Adapter:
  + Classes:
    - PedidoAdapter
  + Implementado em:
    - .
  + Test:
    - Pedido, deveTestarPedidoIntegracao
* Facade:
  + Classes:
    - Financeiro, ClienteFacade
  + Implementado em:
    - Cliente, fazerPedido()
  + Test:
    - ClientTest, deveLancarExcecaoQuandoClienteTemPendencia
* Composite:
  + Classes:
    - Funcionario, FuncionarioCozinha, Equipe
  + Implementado em:
    - .
  + Test:
    - EquipeTest, deveExibirInformacoesDaEquipe
    - FuncionarioCozinhaTest, deveExibirInformacoesDoFuncionarioDaCozinha
* Bridge:
  + Classes:
    - Notificacao, EnviadorNotificacao, EnviadorEmail
  + Implementado em:
    - Cliente, fazerPedido
  + Test:
    - ClientTest, deveConfirmarQueClienteRecebeuNotificacaoPorEmail
    - NotificacaoTest
* Visitor:
  + Classes:
    - Visitor, Usuario, AuditorVisitor, Entregador
  + Implementado em:
    - Cliente, aceitar e getAuditoria
  + Test
    - Clienttest, deveAuditarCliente
    - EntregadorTest

Estrutural

Falta colocar:

1. Proxy (deixar esse pro final, ele vai ter a parte de verificar se o aluno está logado)
2. Decorator (talvez fazer um esquema tipo colocar um adicional no pedido)

Já colocados:

* Observer;
* Chain of Responsability;
* State;
* Singleton;
* Strategy;
* Factory
* Prototype
* Memento
* Command
* iterator
* Mediator
* Template method
* adapter
* Facade
* Composite
* Bridge
* visitor