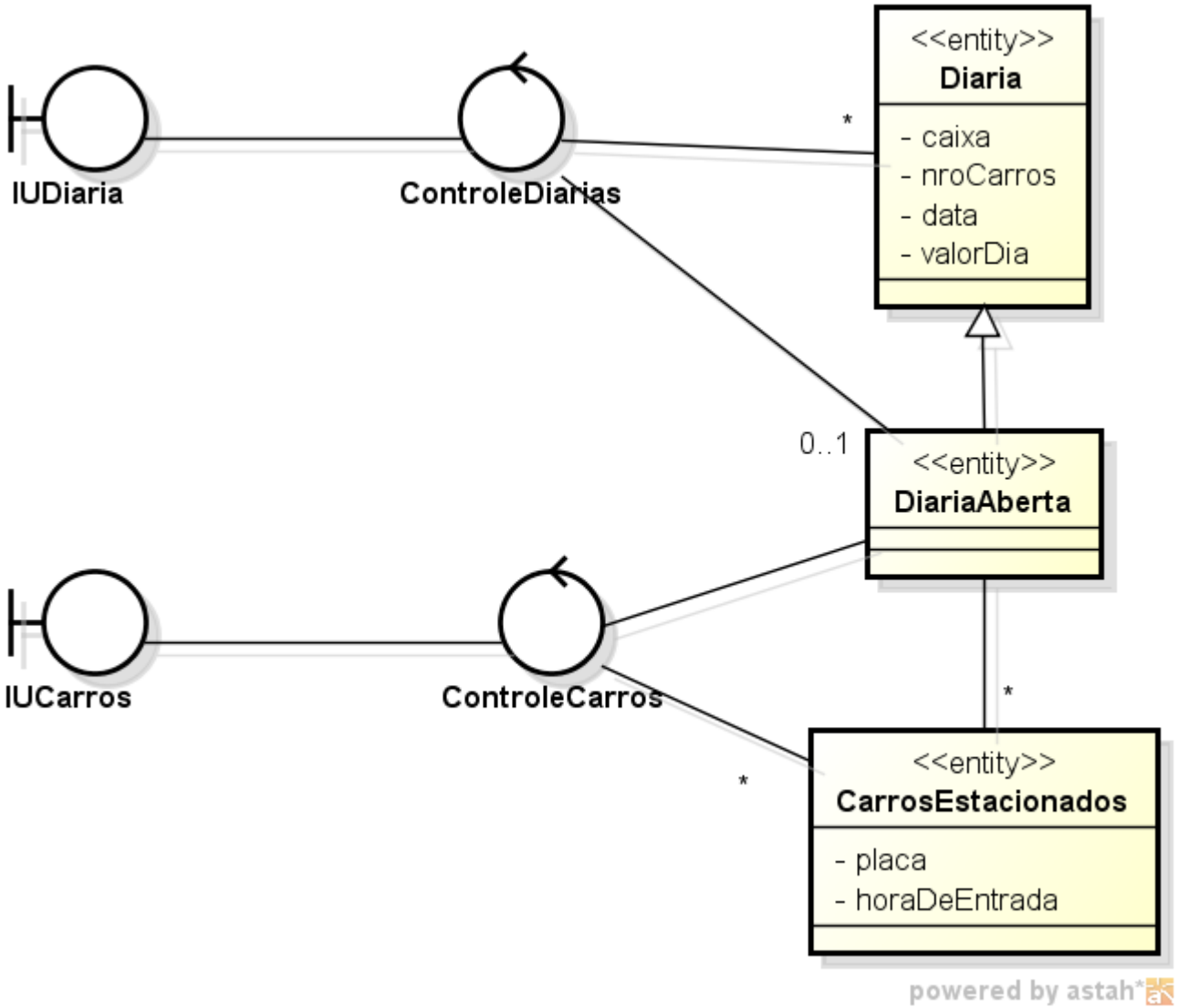
Estudo de Caso: Exercício Estacionamento

Profa. Karin Becker

Engenharia de Software N

Diagrama de Classes de Análise



Antes de mais nada …

- É bom tomar decisões arquiteturais
  - Camadas
- Divisão de responsabilidades
  - Interface e semântica
  - Por quê? Foco em dividir responsabilidades de menor escala em nível de distribuição de métodos
- Na prática
  - Provavelmente mais camadas seria uma melhor escolha
    - Separação lógica do negócio e objetos do domínio
    - Persistência (não será tratada)

Antes de mais nada …

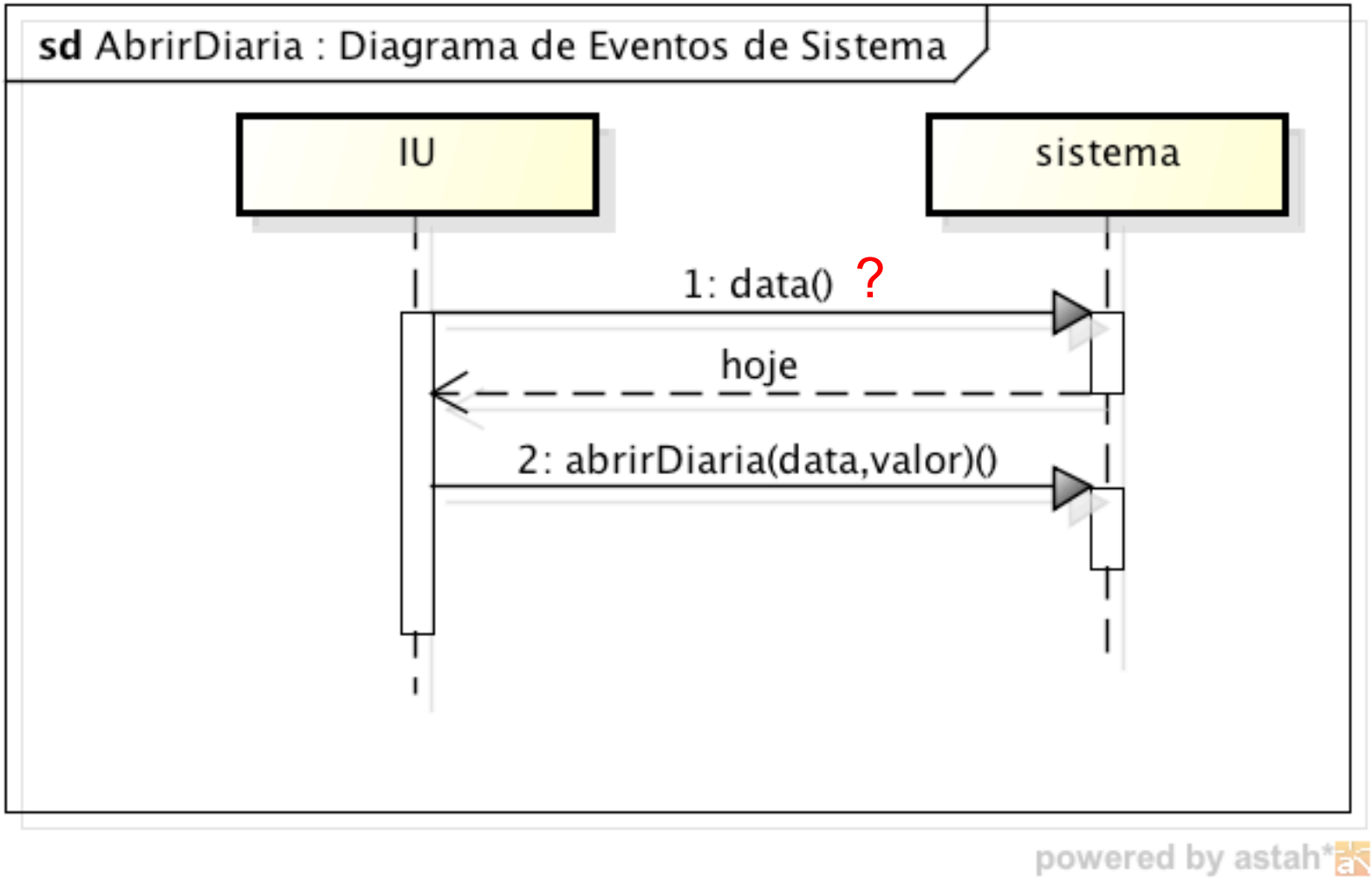
- Escolhas tecnológicas
  - OO
  - Java (ou similar)
- Projeto de interface precede ou caminha lado a lado
- Disclaim
  - Vou seguir método do Larmann
  - Vou trabalhar por realização de caso de uso
    - Novos casos de uso podem me fazer revisitar meu projeto
  - São minhas decisões: vou justificar porque as tomei, você deve justificar as suas !!

Caso de Uso: AbrirDiária

Pré-condição: não há diária aberta
Pós-condição: é criado um controle para o dia (Diaria) com a data atual e valor/hora cobrado, sendo que não há carros estacionados, e o caixa está vazio.
Seqüência típica de eventos:
O caso de uso inicia quando o funcionário solicita a criação de uma diária.
O sistema informa o dia atual.
O funcionário informa o valor/hora a ser cobrado, confirma a data (ou fornece outra data).
O sistema cria um registro de estacionamento para o dia (diária), com a data e valor hora, sendo que o caixa é inicializado com 0 e nenhum carro encontra-se estacionado.
Fluxo Alternativo:
4.a Já existe registro para o dia fornecido
Cancelar a operação.

- Quais são os eventos de sistema??
- O que tem que ser garantido?
- Quais objetos afetados (CRUD)?

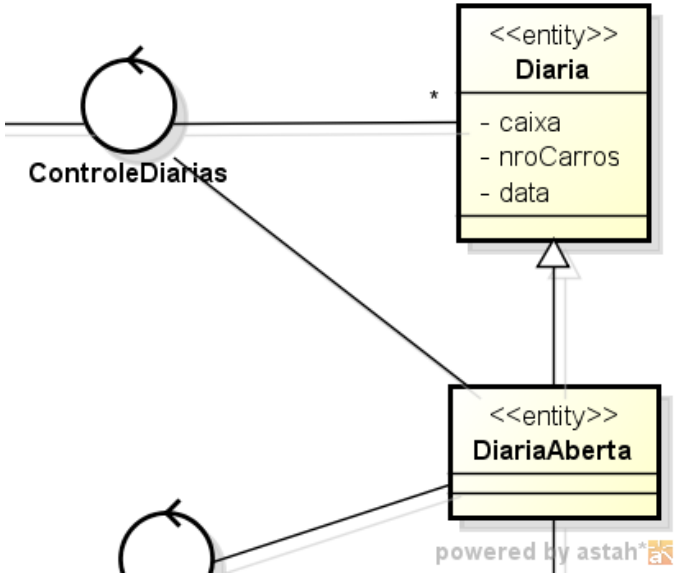
Abrir Diárias

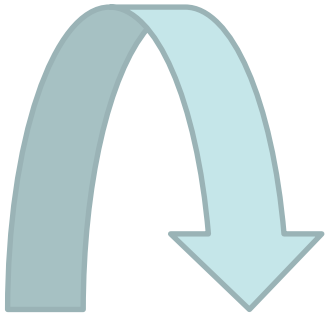


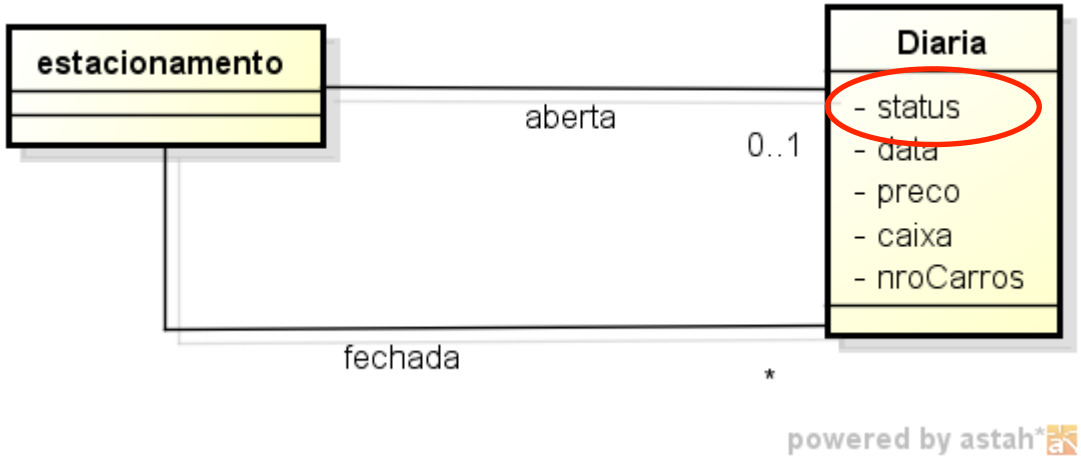
abrirDiaria(data, valor)

- Quem trata deste evento?
  - Padrão controlador
    - Opções:
      - Controladores de caso de uso : não
      - Fachada de caso de uso : talvez ….
      - Fachada : talvez ….
    - Aplicativo tem poucos eventos ➔ fachada
      - Qual classe será a fachada?
      - Diária: tem responsabilidade de gerenciar a movimentação de carros de um único dia
      - Nova Classe: Estacionamento: tem responsabilidade de saber qual diária está aberta

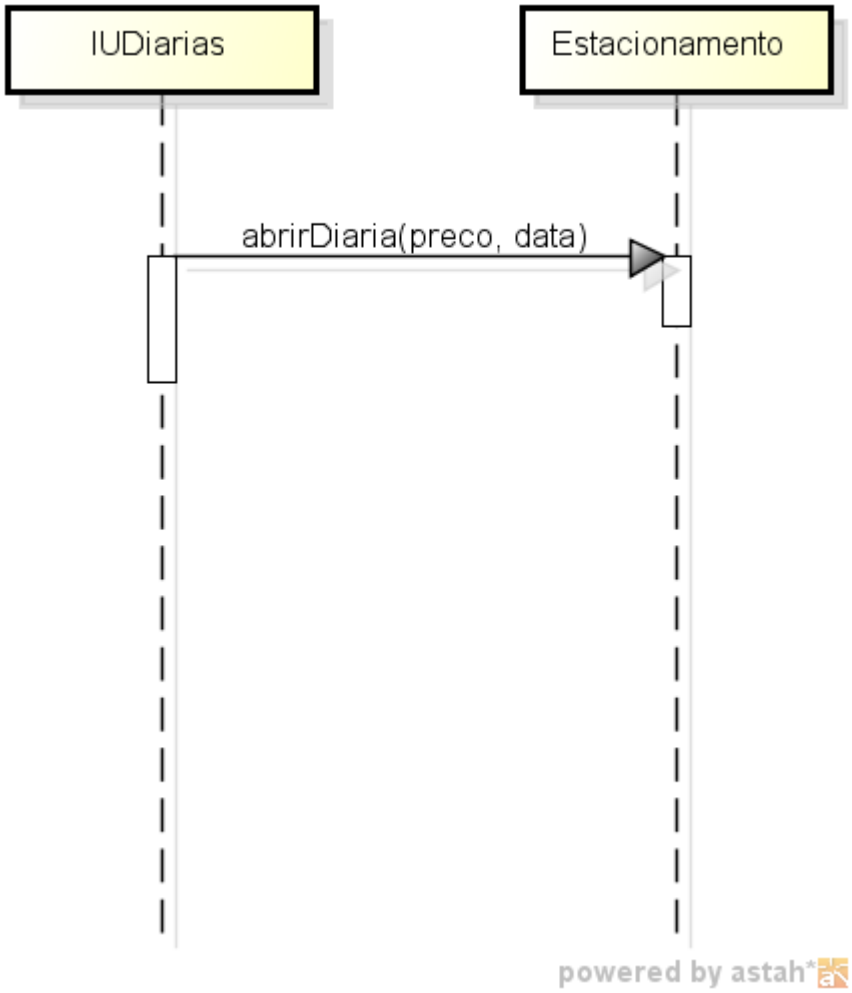
abrirDiaria(data, valor)

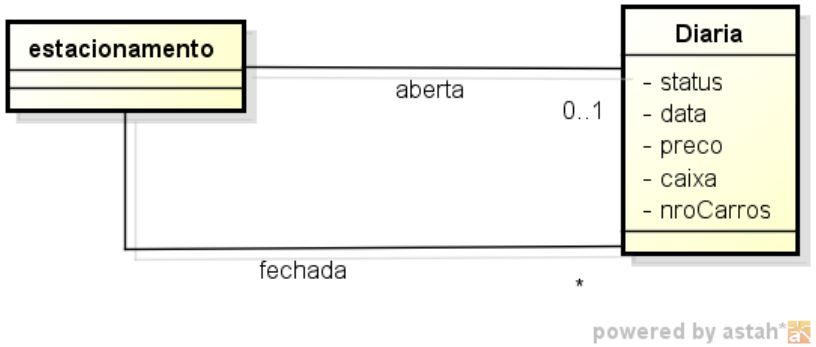






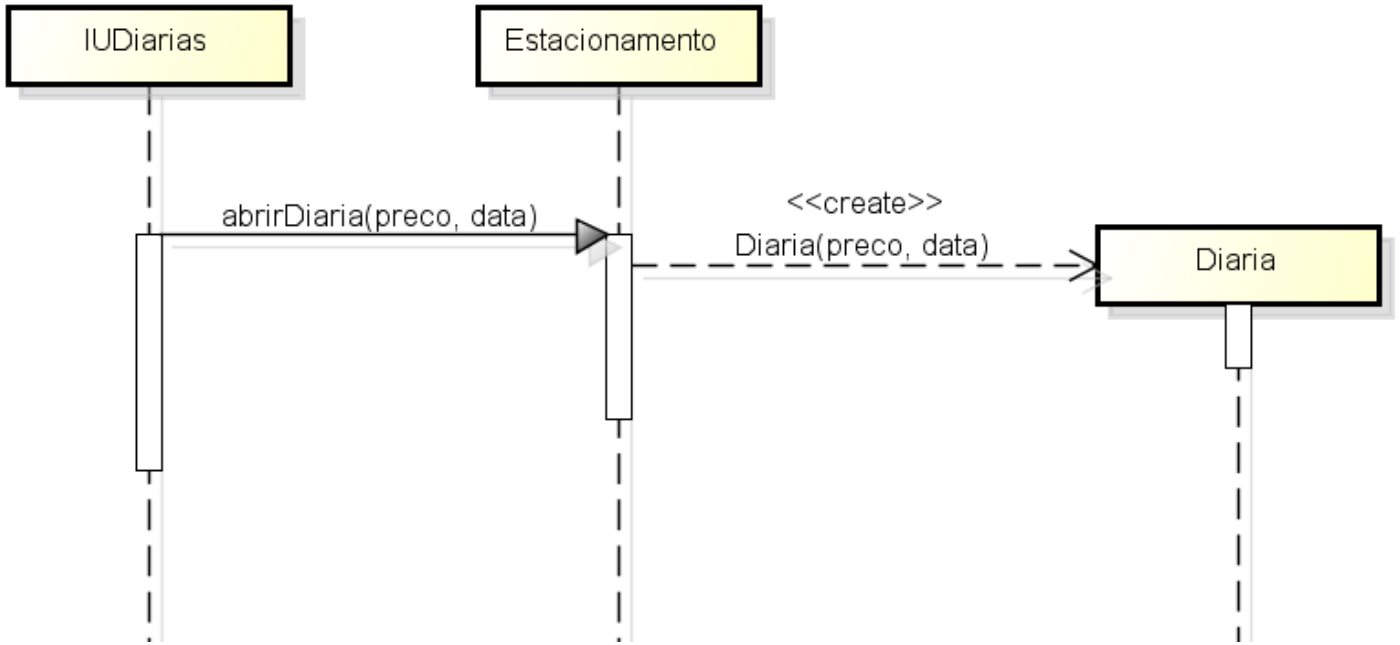
abrirDiaria(data, valor)





abrirDiaria(data, valor)

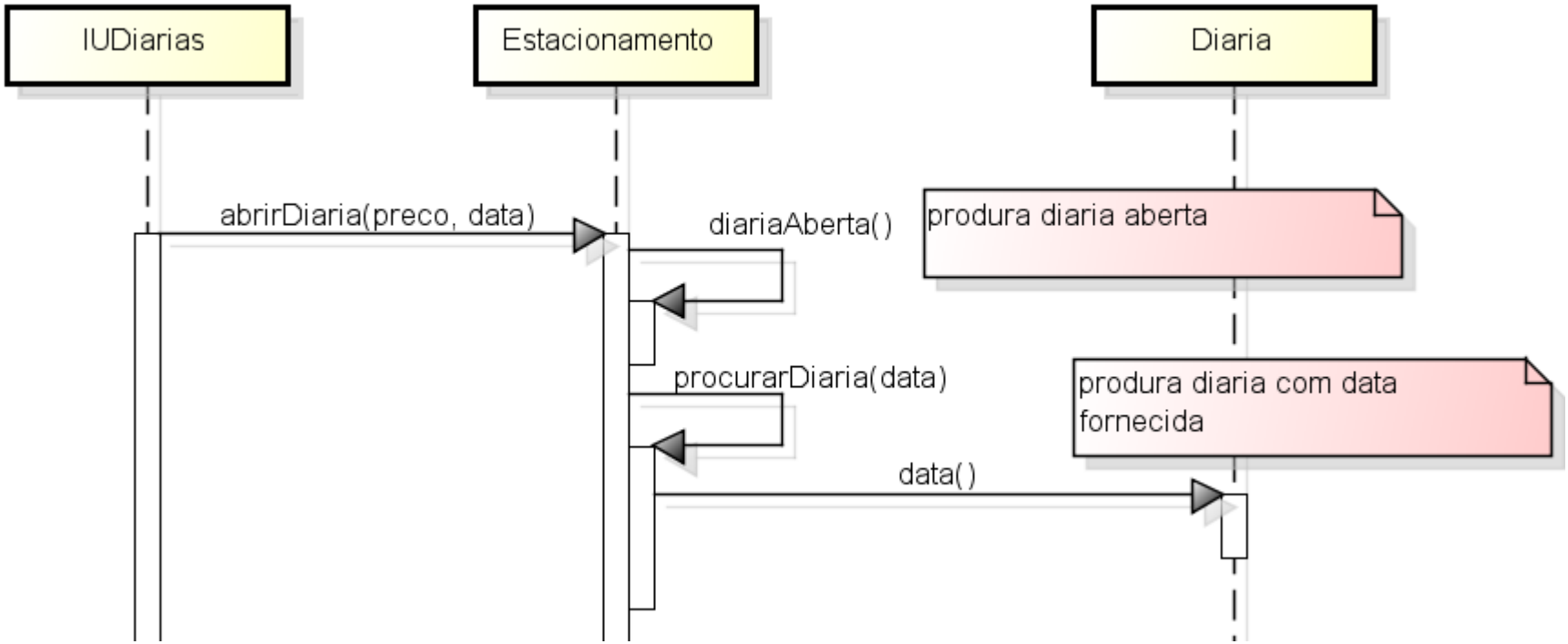
- Quem cria o objeto Diária?
  - Padrão criador
    - Estacionamento agrega/contém objetos Diária
    - Estacionamento registra instâncias de Diaria



abrirDiaria(data, valor)

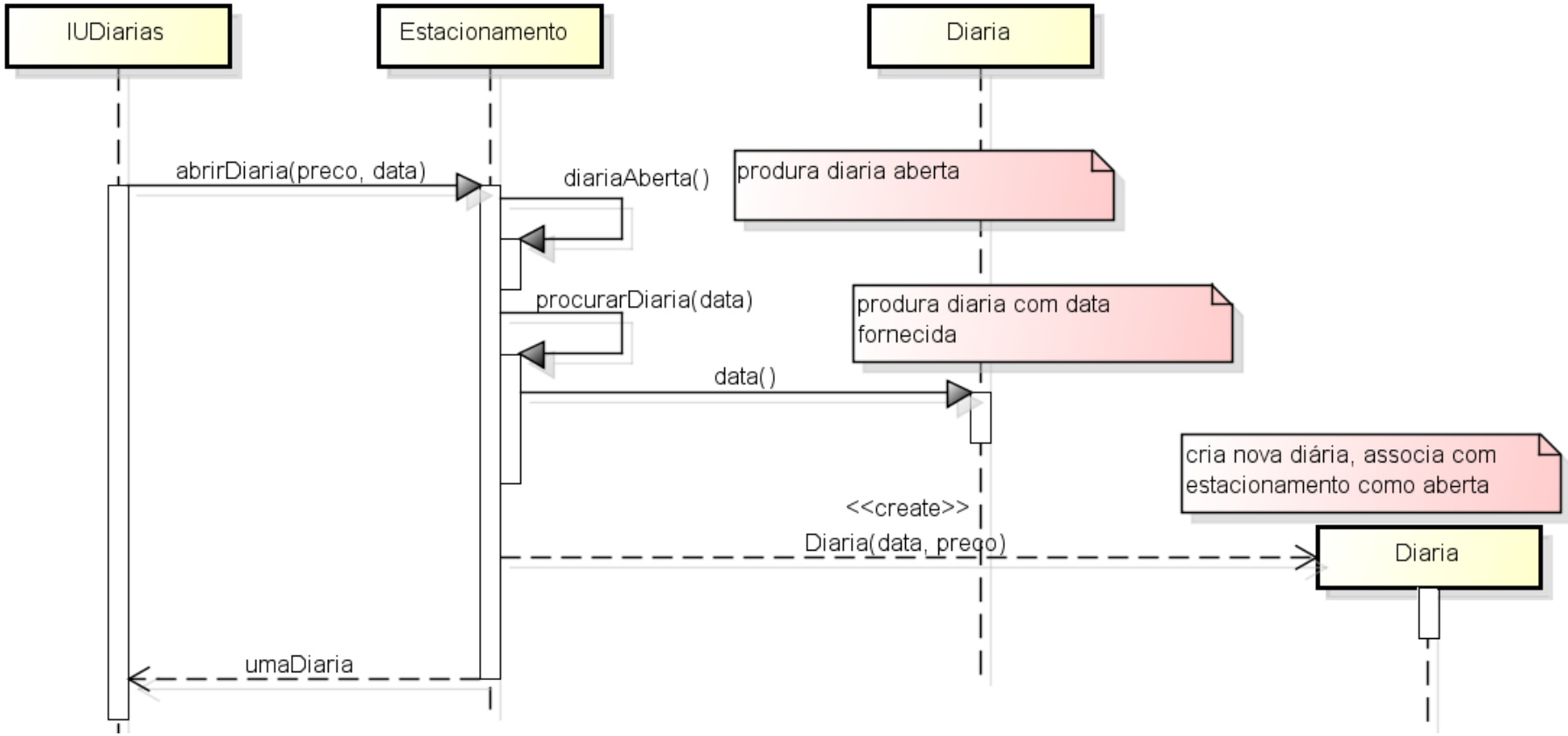
- mas antes de criar uma nova diária, é necessário verificar algumas situações
  - Não há diária aberta
  - Não há diária criada para a data fornecida
- Quem é o especialista?
  - Estacionamento: tem responsabilidade de saber qual diária está aberta
  - Estacionamento: tem responsabilidade de conhecer todas as diárias (fechadas)
  - Diária: responsabilidade de saber a data à qual se refere

verificações



Precisa método diariaAberta??
Precisa método procurarDiária ??
Precisa interagir com Diária?

abrirDiaria(data, valor)



Estacionar Carro

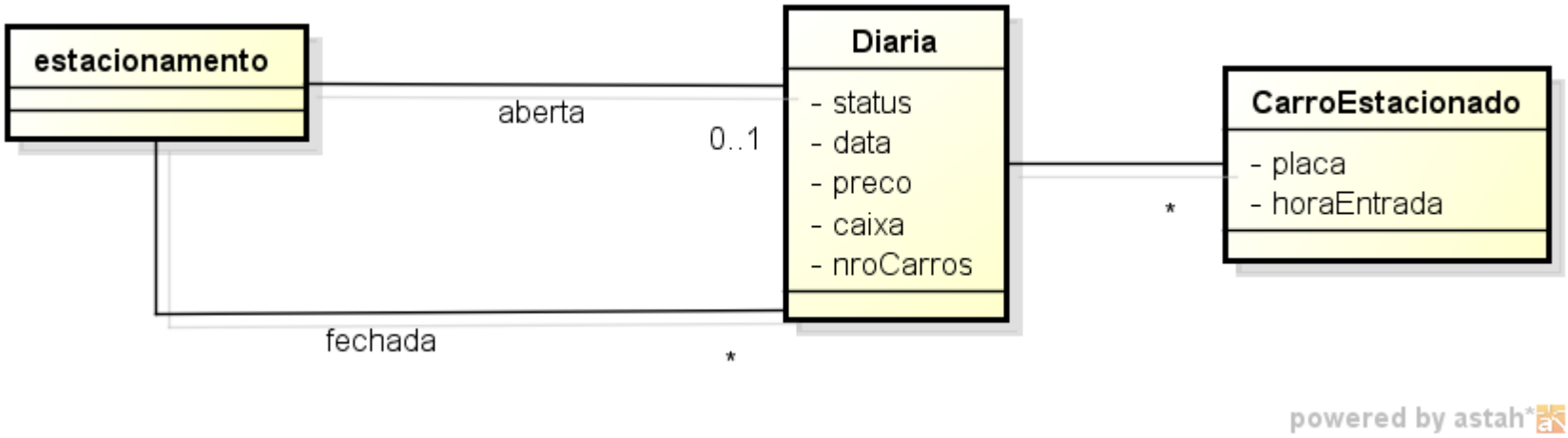
Pré-condição: existe uma Diária aberta
Pós-condição: o carro é incluído no registro da Diária, sendo registrada sua placa e hora de entrada no estacionamento.
Seqüência típica de eventos:
1.O caso de uso inicia quando o funcionário informa a placa do carro.
2. O sistema registra a placa do carro, junto com a hora atual.
Fluxo Alternativo:
2.a Um carro com a placa fornecida já está localizado no estacionamento
Cancelar a operação.

- Quais são os eventos de sistema??
- O que tem que ser garantido?
- Quais objetos afetados (CRUD)?

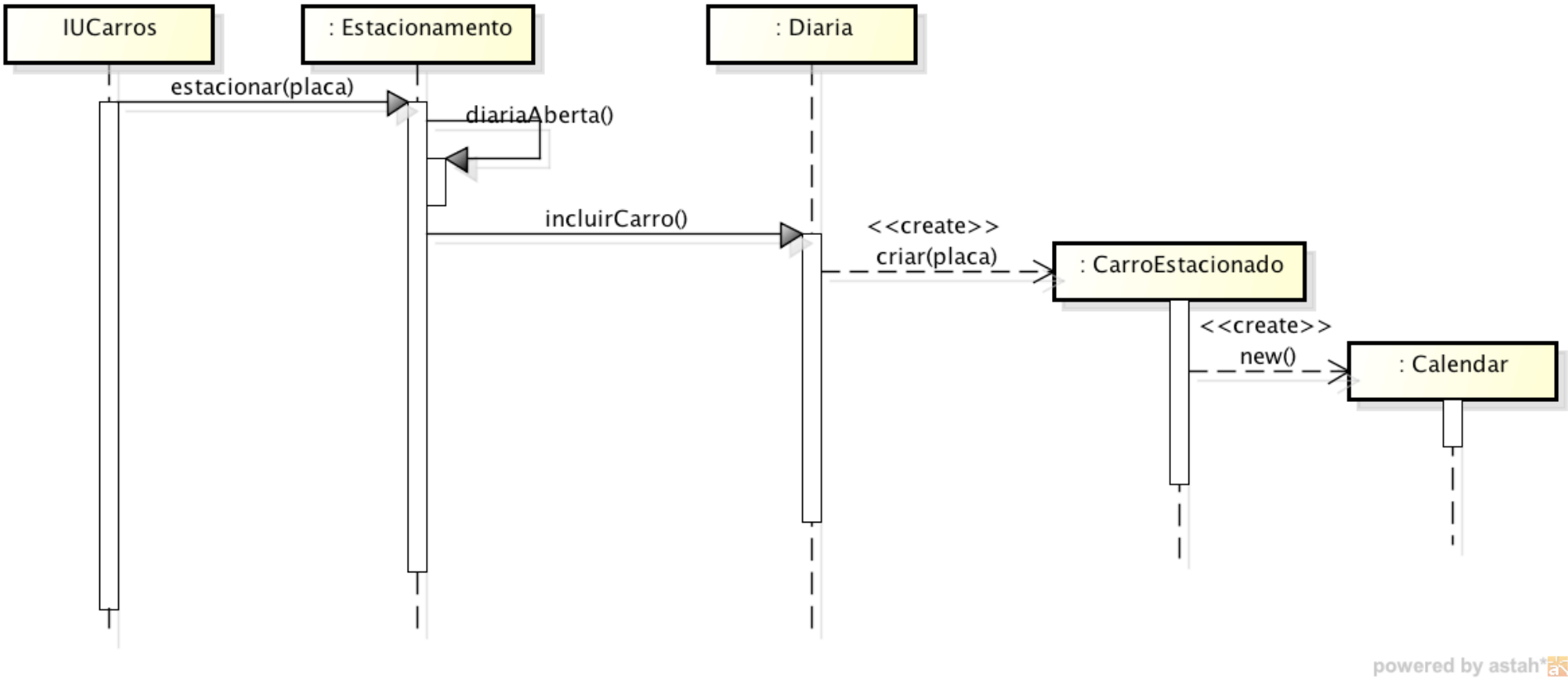
estacionarCarro(placa)

- Quem trata deste evento?
  - Padrão controlador: Estacionamento
    - Aplicativo tem poucos eventos ➔ fachada
      - Estacionamento: tem responsabilidade de saber qual diária está aberta
      - Vale a pena voltar à decisão de fachada de caso de uso? Talvez ….
- Quem cria o carro?
      - Diária agrega/contém objetos Carro
      - Diária registra instâncias de Carro
- Quem verifica se pode ou não estacionar o carro?
      - Diária: tem responsabilidade de gerenciar a movimentação de carros de um único dia (conhece todos os carros estacionados)
      - Carro: conhece a sua placa

Estacionar Carro



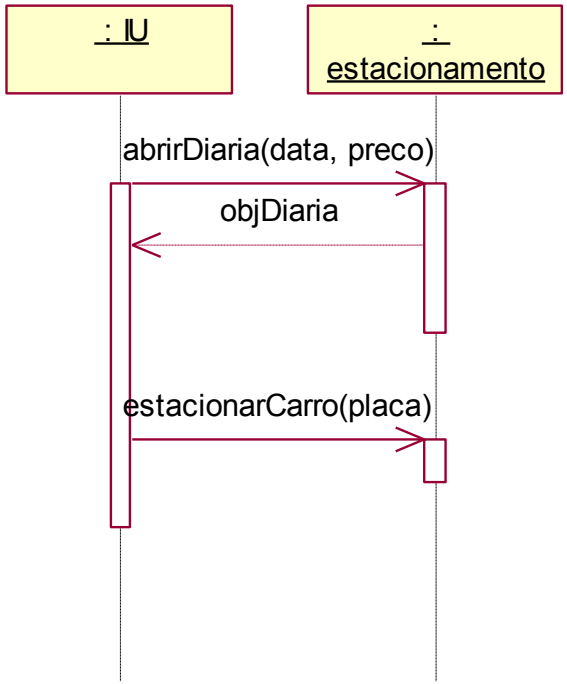
Estacionar Carro

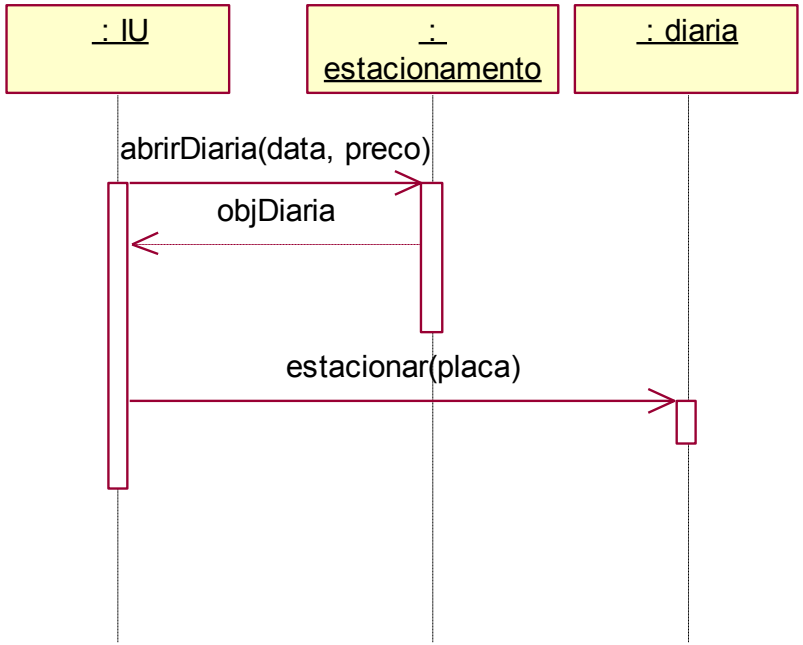


Atenção: criar vs incluir na coleção

Estacionar Carro

- Se eu juntasse estes dois diagramas, não dava para ter mandado a mensagem “estacionarCarro” direto para a Diária? Afinal, após eu abrir a diária, recebo como resposta o objeto objDiaria ... Ficaria mais eficiente, não?





Estacionar Carro: Controlador

- Sempre para mandar uma mensagem deve-se conhecer o receptor.
  - Lembre-se que entre abrir uma diária e estacionar um carro na diária aberta, o usuário pode pedir para fechar a diária … você estaria duplicando as classes que conhecem a diária aberta (Estacionamento e IU)!!!!
  - Se você jogar a responsabilidade de conhecer a diária aberta (também) na interface do usuário (IU), vai ter de controlar duas vezes a mesma coisa
- Conhecer os objetos que podem tratar um evento do sistema é um dos principais, e desacoplar a interface da semântica da aplicação, são os dois principais papéis do controlador
- Decisão deste projeto: Estacionamento trata este evento!

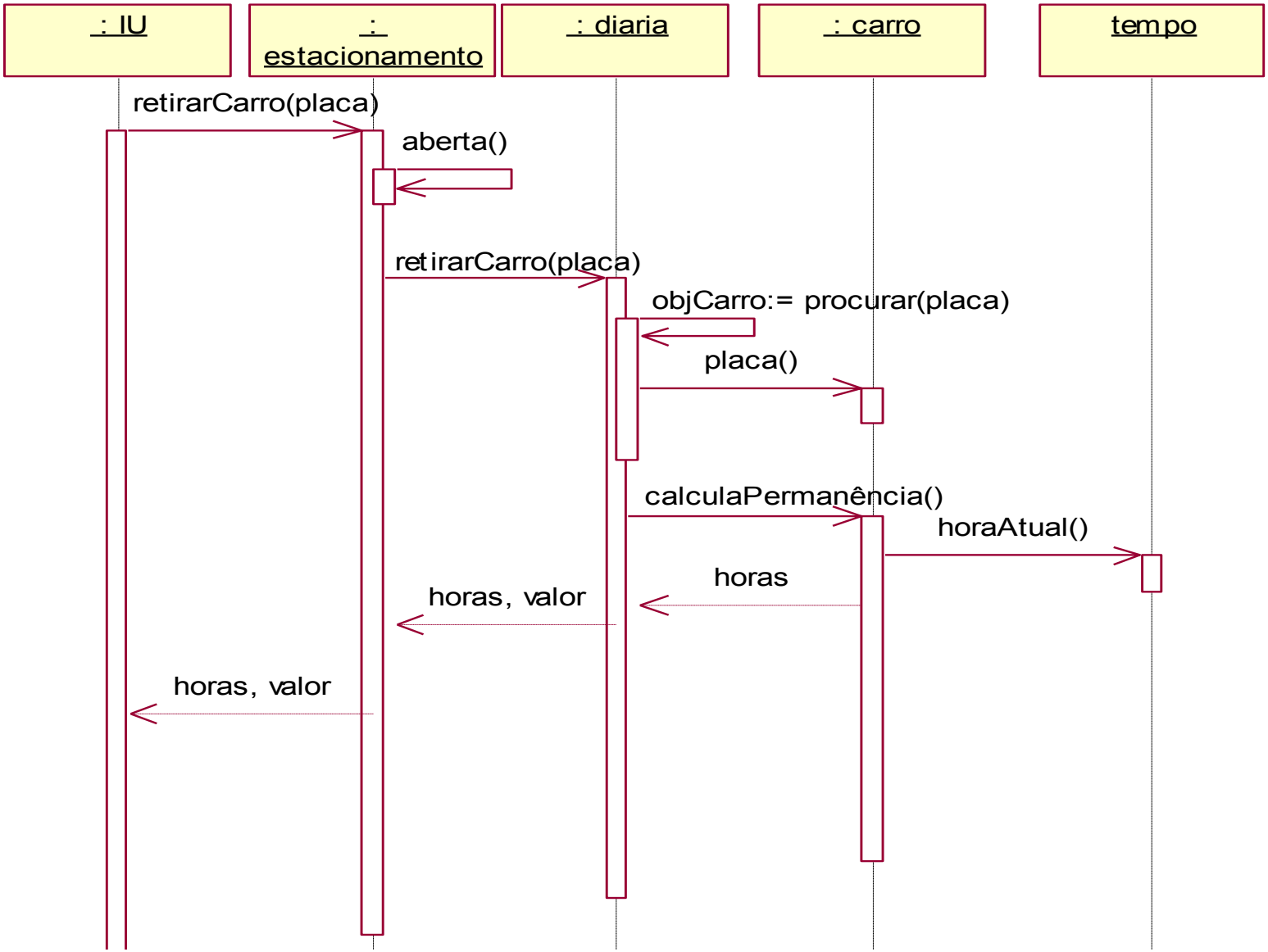
Retirar o carro

Pré-condição: existe uma Diária aberta
Pós-condição: é excluído o registro de estacionamento do carro, é informada a duração de sua estadia, bem como o preço a pagar. As informações da diária são atualizadas: o caixa é acrescido do valor cobrado (pago), e a informação do número de carros que já deixaram o estacionamento é atualizada.
Seqüência típica de eventos:
O caso de uso inicia quando o funcionário informa a placa do carro a ser retirado
O sistema calcula quantas horas o carro permaneceu estacionado (é cobrada hora cheia), o preço da estadia (dependente do valor cobrado na diária), atualizando as informações de movimentações de carro do dia (número de carros estacionados correntemente, e número de carros que já saíram), acrescentando o preço pago ao caixa diário.
Fluxo Alternativo:
2.a Um carro com a placa fornecida não está localizado no estacionamento
Cancelar a operação.

Retirar o Carro

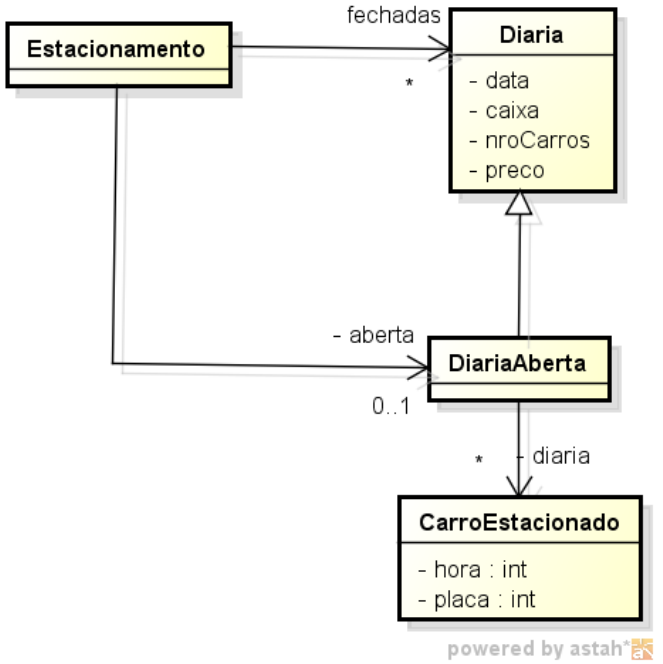
- Estacionamento controla o evento
  - Será que ele não está controlando muitos eventos?
- Especialista
  - Quem sabe a informação necessária para calcular o valor a pagar?
    - Carro: sabe a hora quando entrou;
    - Diaria: sabe o valor da hora naquela diária, sabe se o carro está estacionado
    - Estacionamento: sabe a diária aberta

Retirar Carro

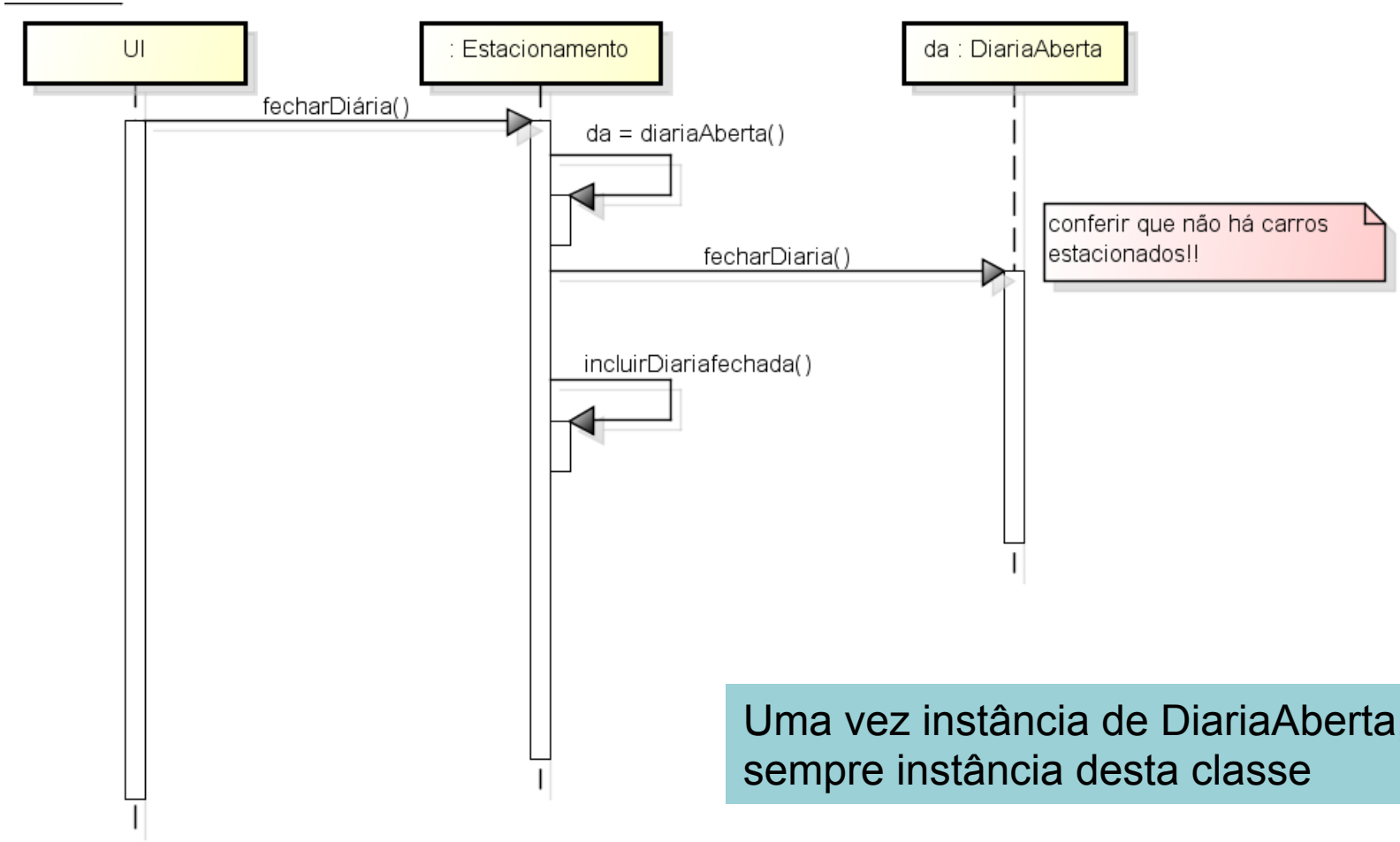


Diaria Aberta

- estacionarCarro é uma operação que só uma diária aberta pode realizar
  - Será que vale a pena rever a decisão de não modelá-la como uma subclasse?
- retirarCarro também ….



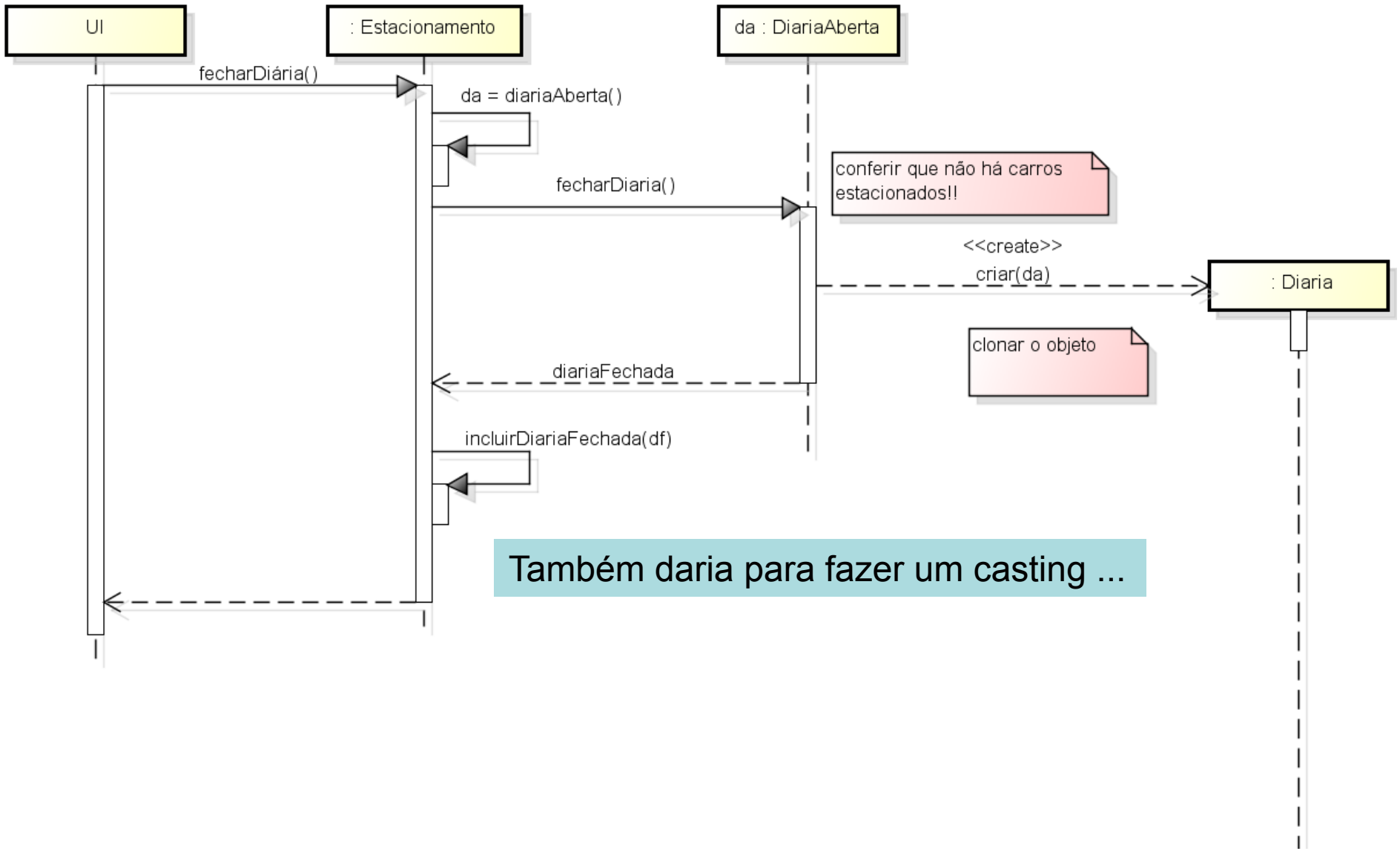
Fechar Diária



Uma vez instância de DiariaAberta, sempre instância desta classe

powered by astah*

FecharDiaria



Também daria para fazer um casting ...

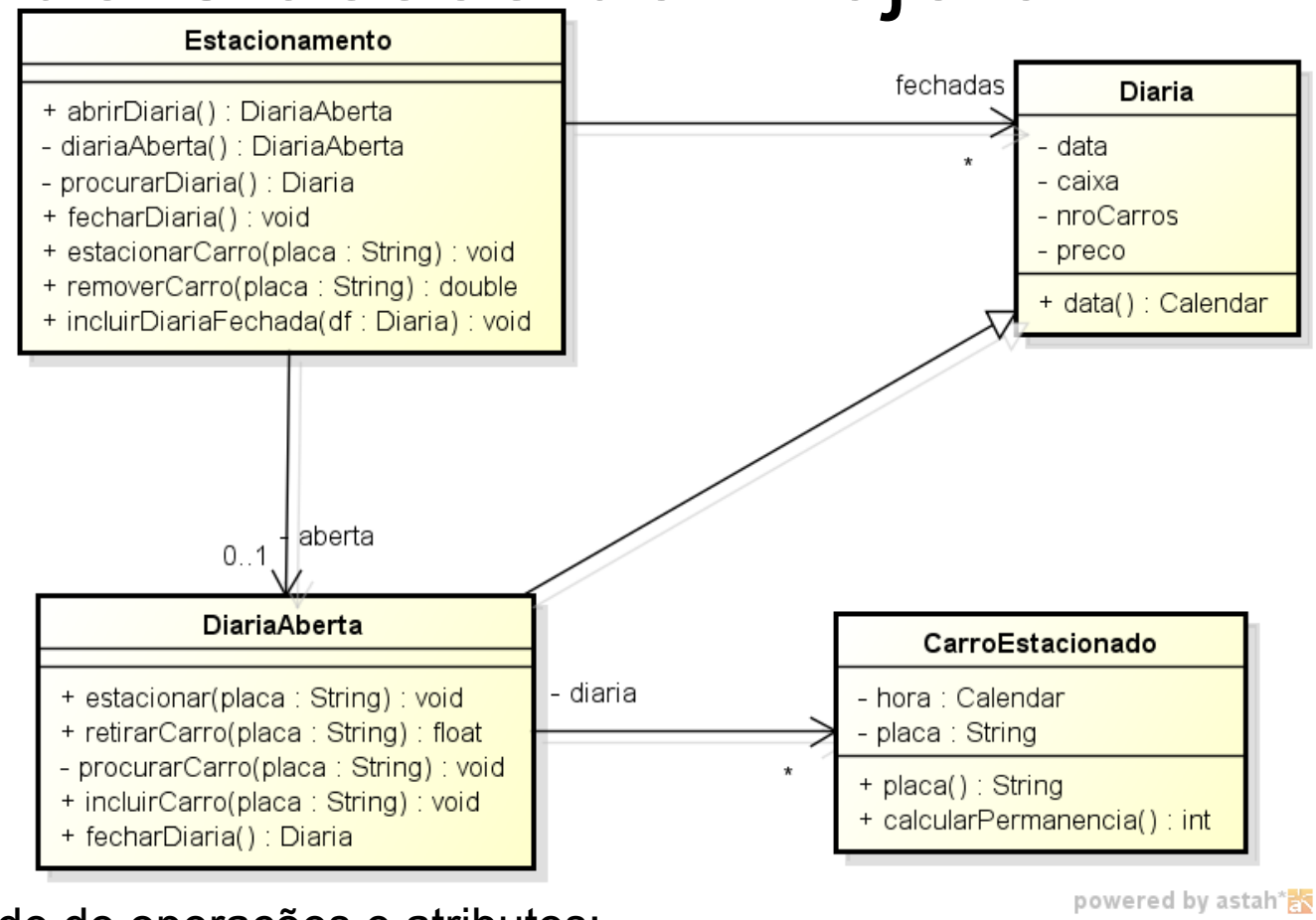
Lembretes

- Controlador: tem como responsabilidade saber tratar os eventos do sistema
  - Quem?
  - Quais informações são necessárias, além dos parâmetros fornecidos pelo usuário?
- Diferença entre objeto e valor
  - Dois objetos com o mesmo número de placa não são o mesmo objeto!!!
- Distribuição uniforme de responsabilidade

Diagrama de Classes

- São feitos em paralelo com os diagramas de interação
- Cada mensagem trocada corresponde a uma operação
  - Decidir se é pública ou privada
- Novos atributos em relação ao conceitual podem ser necessários
- Novas classes e novos atributos (em relação ao conceitual) podem ser necessários
- Definir sentido da navegação nas associações

Diagrama de Classes de Projeto



- ainda em construção
- decisões sobre visibilidade de operações e atributos;
- Associações têm sentido de navegação definido
- poderia ter incluido relações de dependência (Calendar) não foi acrescentada
- ainda falta ….
    - verificar coesão e acoplamento?
    - outros casos de uso
    - persistência?