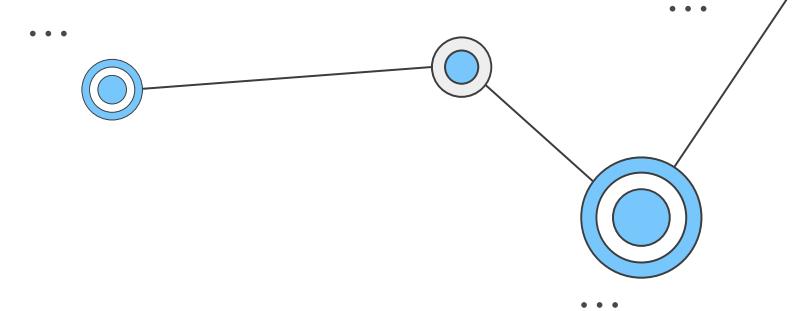
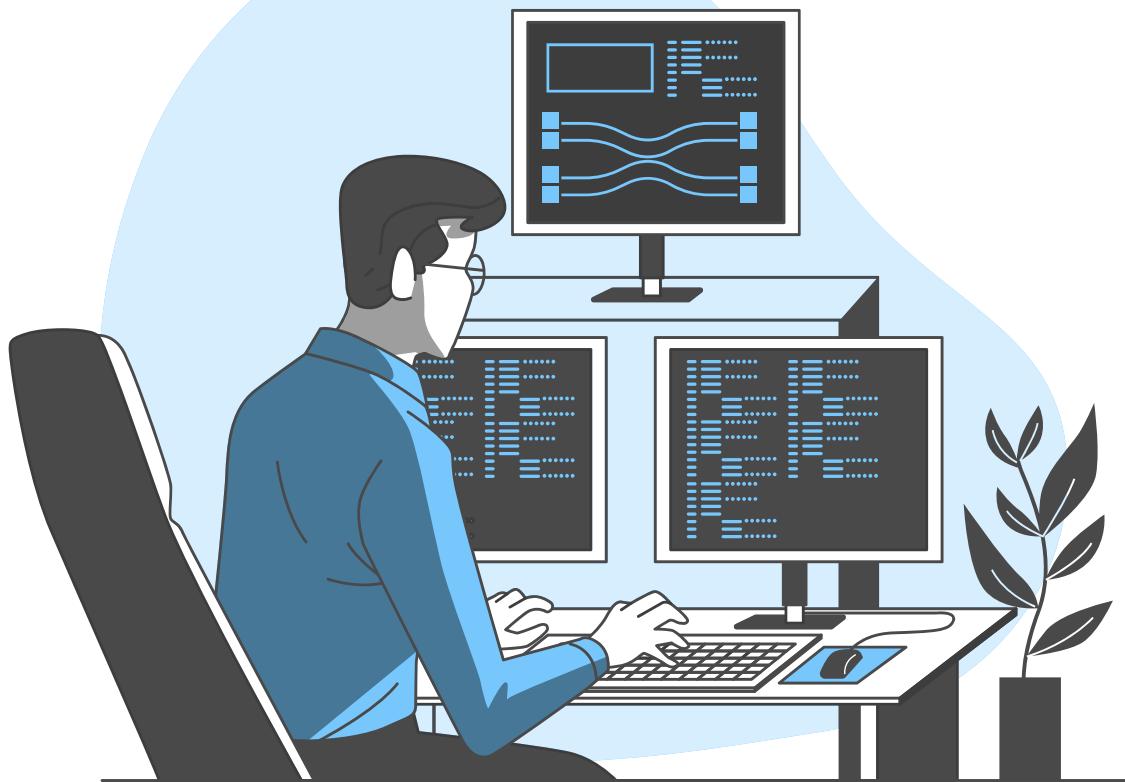




PUC Minas



Projeto de Software

Prof. Dr. João Paulo Aramuni

Unidade 5

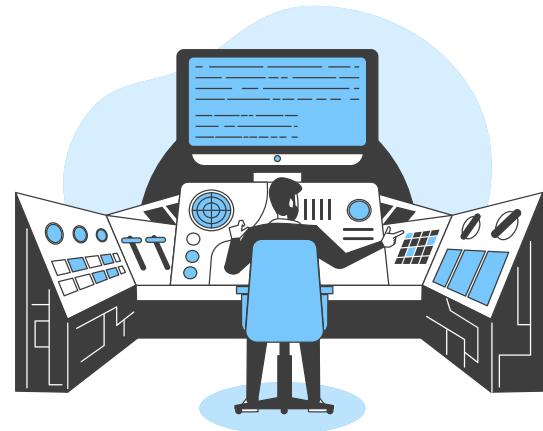
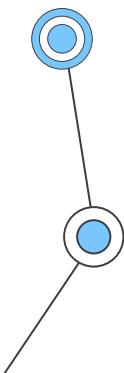
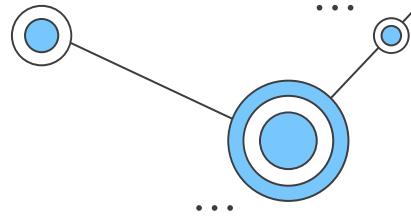
Projeto de Classes

PDS - Manhã

Projeto de Classes

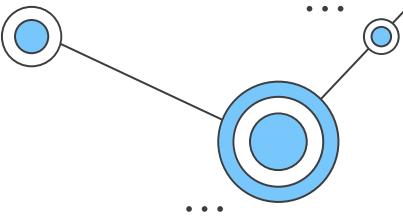
Sumário

- Introdução
- Diagramas de transição de estados
- Identificação dos elementos de um diagrama de estados
- Construção de diagramas de transição de estados
- Modelagem de estados no processo de desenvolvimento



Projeto de Classes

Contexto estudo dessa unidade: Classe de Projeto

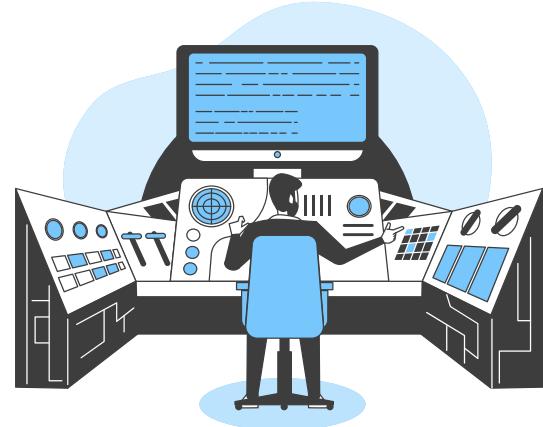
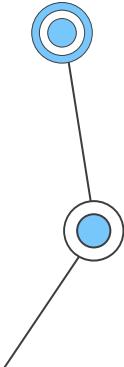


Alinhamento do diagrama de classe com outros artefatos.

- Adição de novos atributos.
- Adição de novas operações.
- Verificação dos caso de uso.

Definição do ciclo de vida dos objetos de uma classe.

- Entendimento da classe.



Projeto de Classes

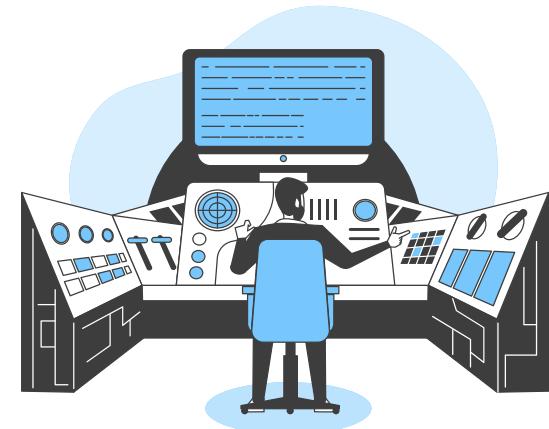
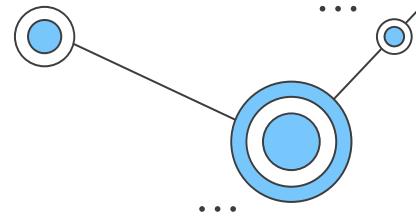
Introdução

Objetos do mundo real se encontram em estados particulares a cada momento.

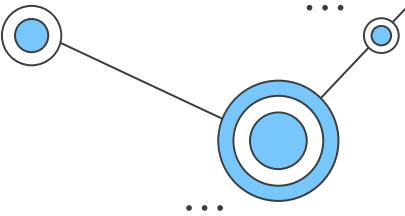
- uma jarra está cheia de líquido.
- uma pessoa está cansada.

Da mesma forma, cada objeto participante de um sistema de software orientado a objetos se encontra em um estado particular.

Um objeto muda de estado quando acontece algum evento interno ou externo ao sistema.



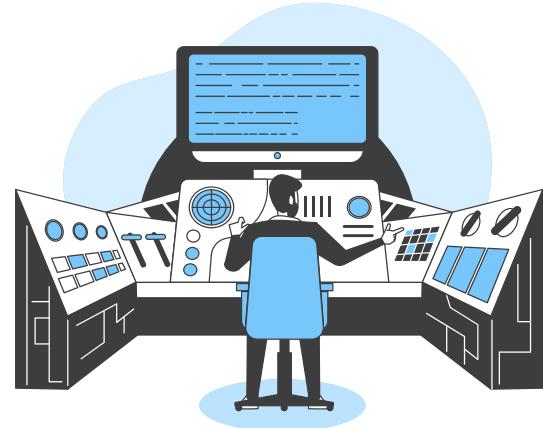
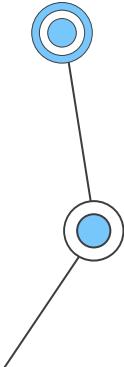
Projeto de Classes



Introdução

Durante a transição de um estado para outro, um objeto normalmente realiza determinadas ações dentro do sistema.

Quando um objeto transita de um estado para outro, significa que o sistema no qual ele está inserido também está mudando de estado.



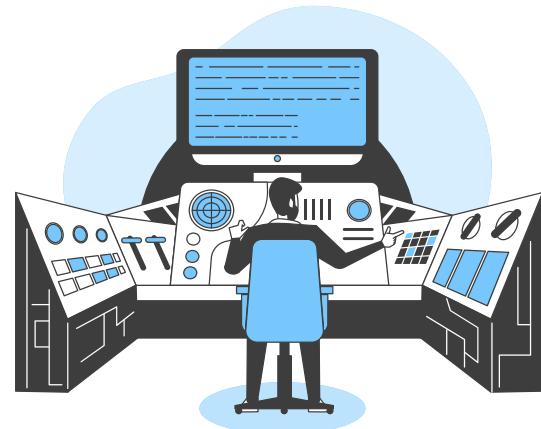
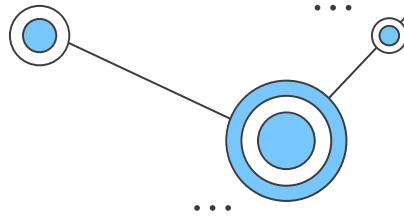
Projeto de Classes

Introdução

Por meio da análise das transições entre estados dos objetos de um sistema de software, pode-se prever:

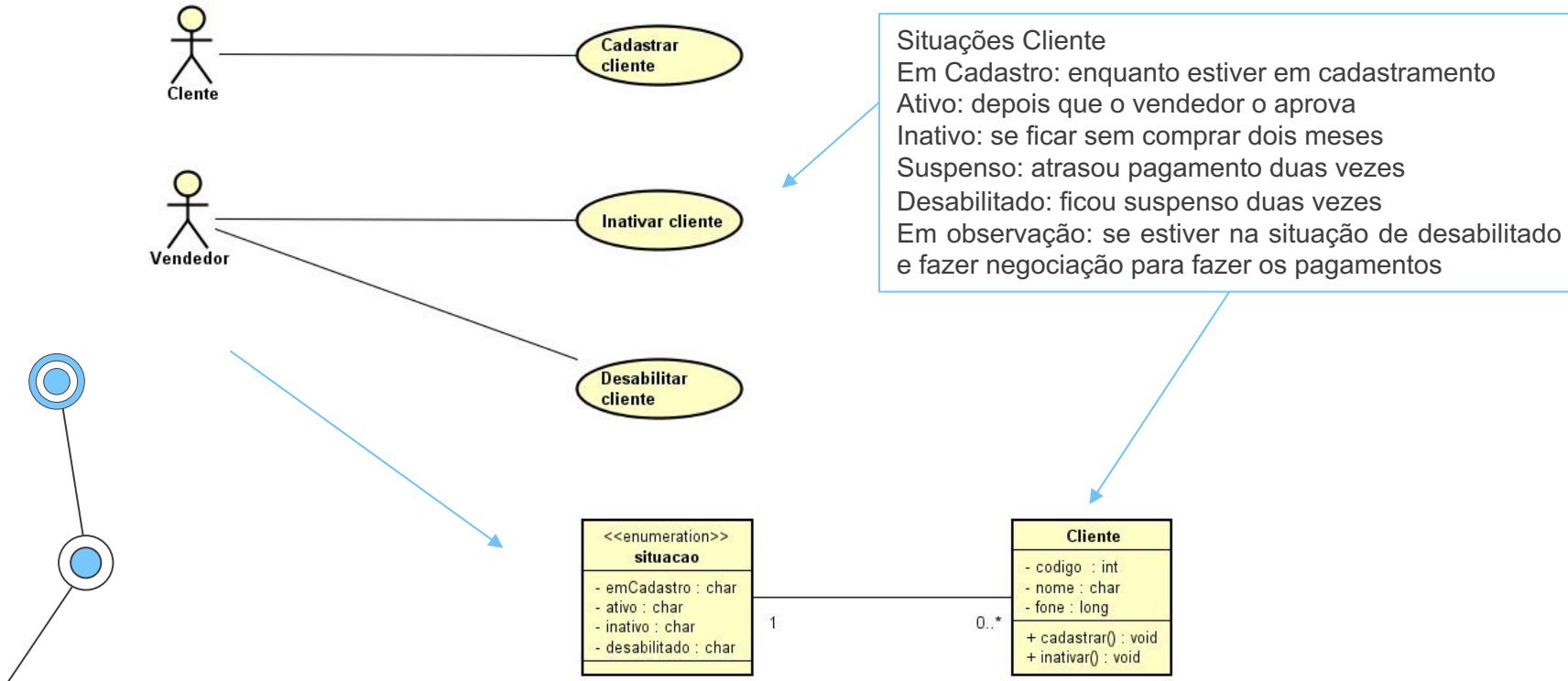
- todas as possíveis operações realizadas, em função de eventos que possam ocorrer.
- Todos atributos necessários para atender as mudanças de estados.
- Se todos os estados estão contemplados nos casos de uso.

O diagrama da UML que é utilizado para realizar esta análise é o **Diagrama de Transição de Estado**.



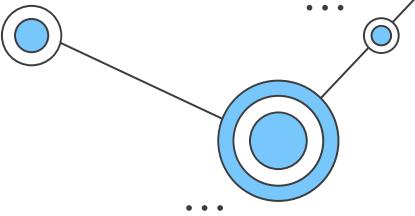
Projeto de Classes

Introdução



Projeto de Classes

O que é um diagrama de Transição de Estados?

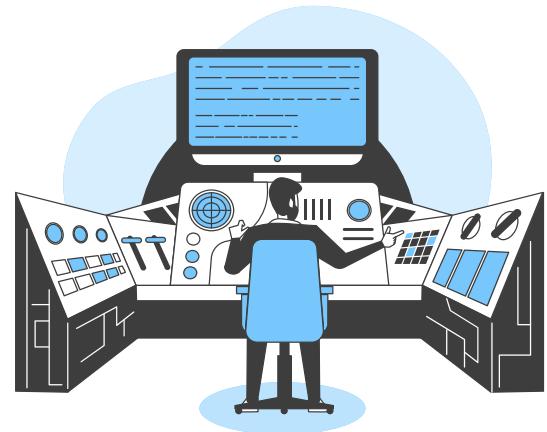


Usado para mostrar o ciclo de vida de uma determinada classe, os eventos que causam uma transição de um estado para outro e as ações que resultam de uma mudança de estado.

O domínio de estados de uma classe é o conjunto de todos os possíveis estados de um objeto.

O estado de um objeto é uma das possíveis condições nas quais um objeto pode existir.

- Incorpora todas as propriedades de um objeto - geralmente estático
- Mais os valores correntes de cada propriedade do objeto - geralmente dinâmico.

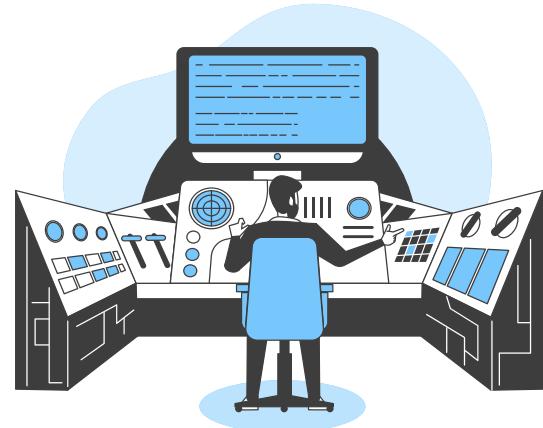
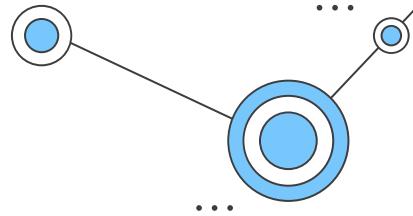


Projeto de Classes

Diagrama de transição de estado

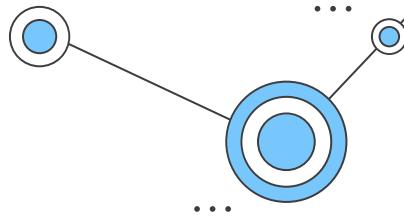
A UML tem um conjunto rico de notações para desenhar um DTE.

- Estados
- Transições
- Evento
- Ação
- Atividade
- Transições internas
- Estados aninhados
- Estados concorrentes



Projeto de Classes

Estado

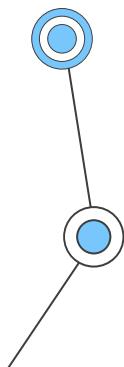


Situação na vida de um objeto em que ele satisfaz a alguma condição ou realiza alguma atividade.

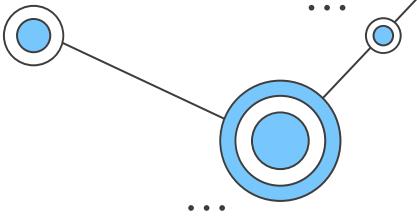
É função dos valores dos atributos e (ou) das ligações com outros objetos.

- O atributo reservado deste objeto livro tem valor verdadeiro.
- Uma conta bancária passa para o vermelho quando o seu saldo fica negativo.
- Um professor está licenciado quando não está ministrando curso algum durante o semestre.
- Um tanque está na reserva quando nível de óleo está abaixo de 20%.
- Um pedido está atendido quando todos os seus itens estão atendidos.

Estados podem ser vistos como uma abstração dos atributos e associações de um objeto.



Projeto de Classes

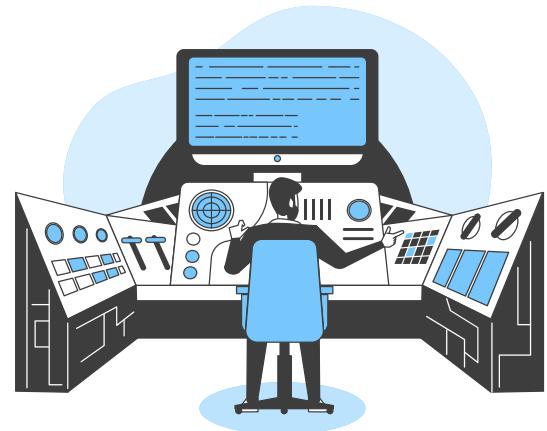
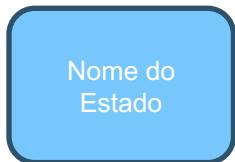


Estados - Representação Gráfica

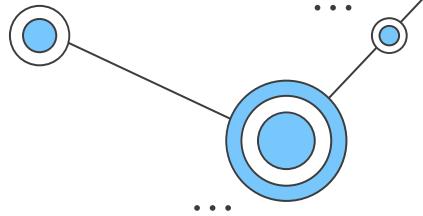
na UML é Representado por retângulo com as bordas arredondadas

Nome do estado:

- muitas vezes o nome é descrito no gerúndio
- No entanto em alguns casos o objeto pode estar esperando por um evento e portanto, estará em um estado estático ou de espera



Projeto de Classes

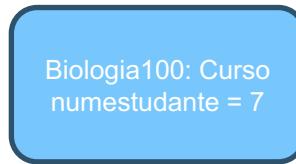


Estados - Representação Gráfica

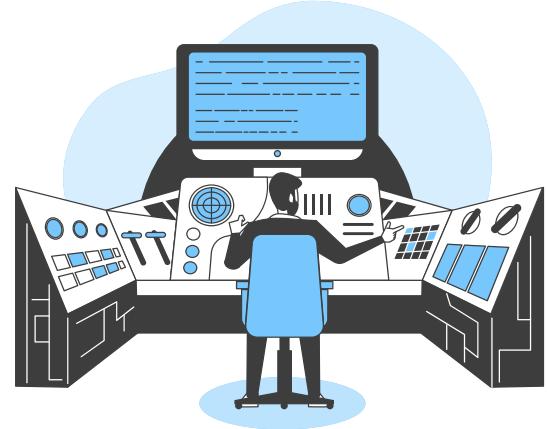
Os estados podem ser distinguidos pelos valores de certos atributos



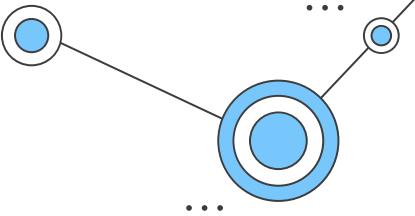
$\text{numestudantes} < 10$



$\text{numestudantes} \geq 10$

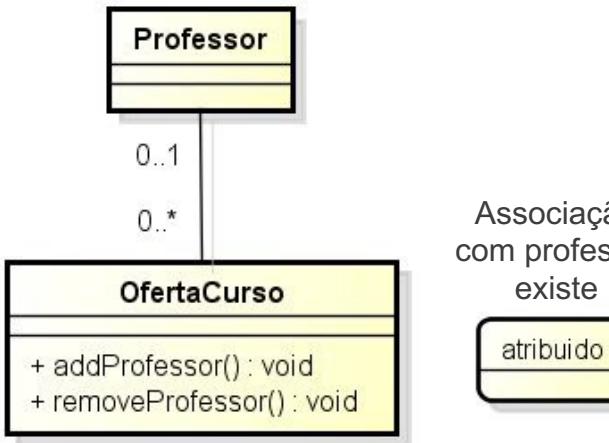


Projeto de Classes



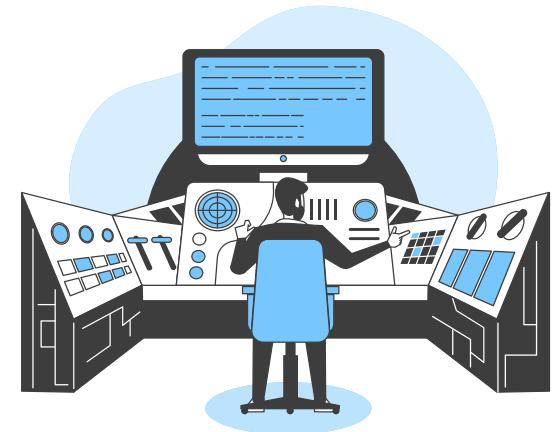
Estados - Representação Gráfica

Os estados podem ser distinguidos pelas associações



Associação
com professor
não existe

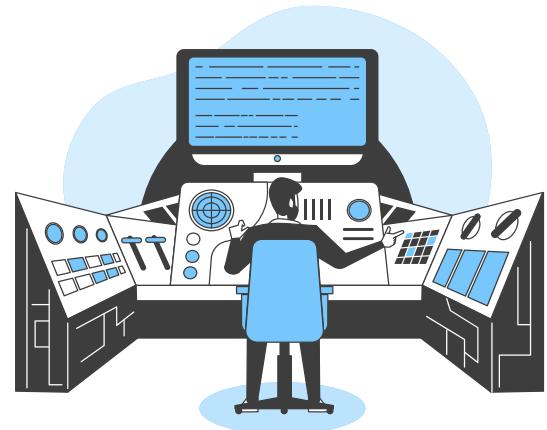
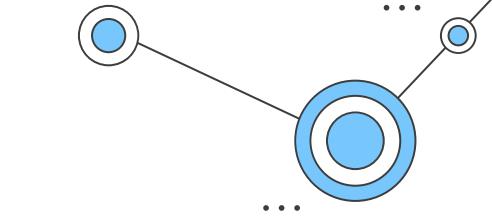
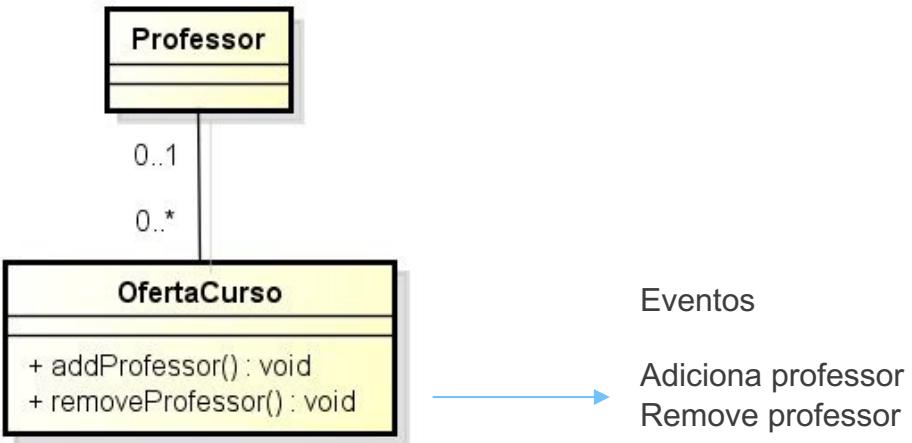
não atribuído



Projeto de Classes

Estados - Representação Gráfica

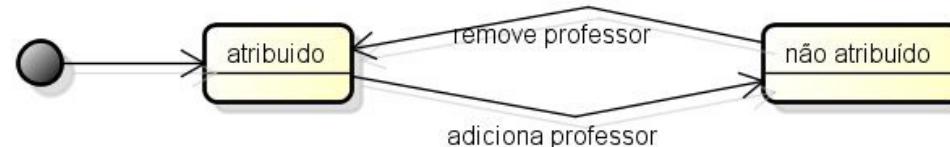
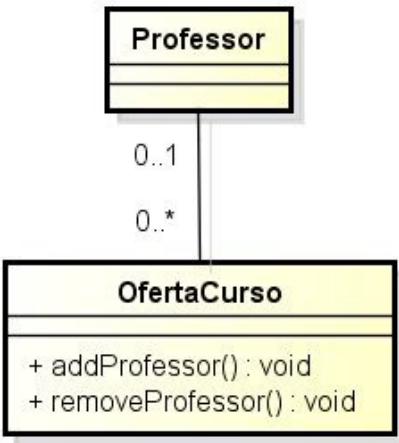
Os estados podem ser distinguidos pelas associações



Projeto de Classes

Estados - Representação Gráfica

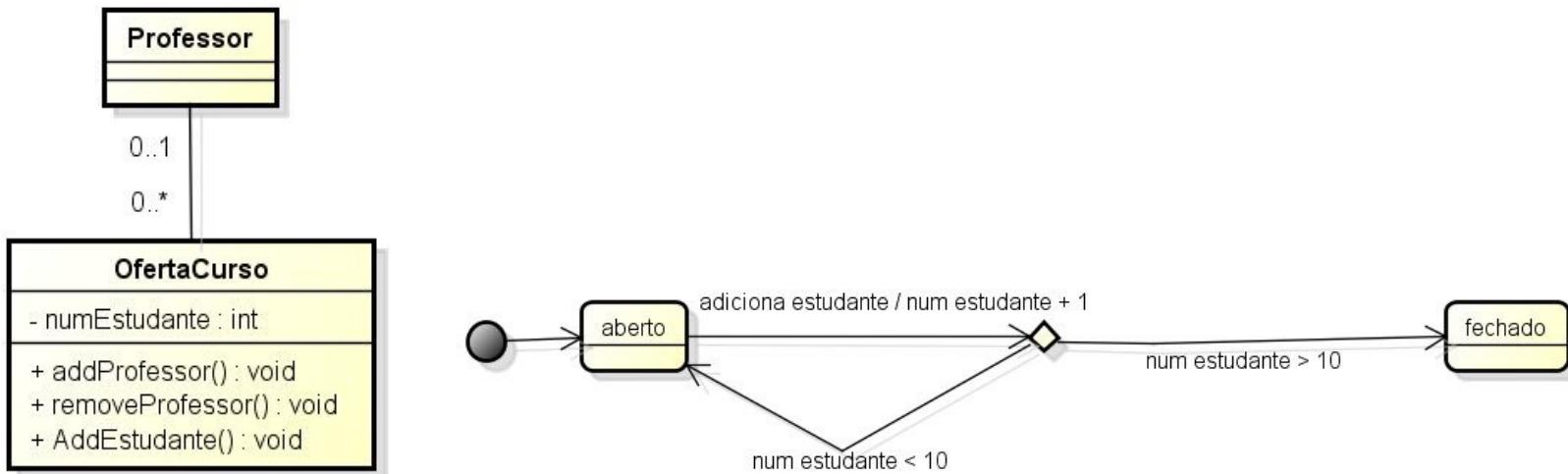
Para cada estado, determinar quais eventos causam transições para que estados, incluindo as condições de guarda, quando necessário



Projeto de Classes

Estados - Representação Gráfica

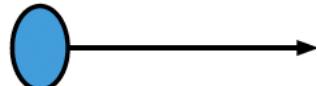
- Eventos podem ser mapeados para operações
- Estados são muitas vezes representados usando atributos



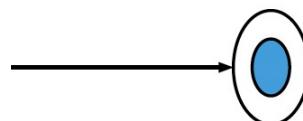
Projeto de Classes

Estados - Representação Gráfica - Estados inicial e final

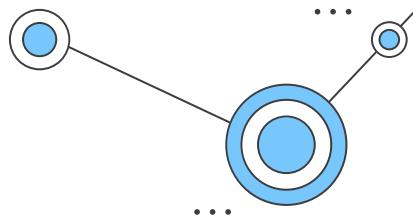
- O estado inicial indica o estado de um objeto quando ele é criado. Só pode haver um estado inicial em um DME.
 - Essa restrição serve para definir a partir de que ponto um DME deve começar a ser lido.
- O estado final é representado como um círculo “eclipsado” e indica o fim do ciclo de vida de um objeto.
 - Este estado é opcional e pode haver mais de um estado final em um DME.
- Representação



Estado inicial



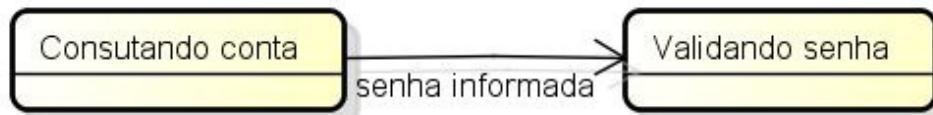
Estado final



Projeto de Classes

Transições

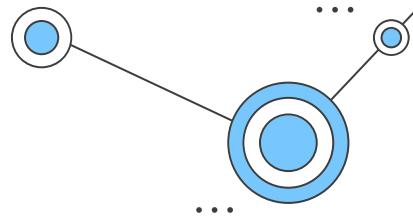
- Os estados estão associados a outros pelas transições.
- Uma transição é mostrada como uma linha conectando estados, com uma seta apontando para um dos estados.
- Quando uma transição entre estados ocorre, diz-se que a transição foi disparada.
- Uma transição pode ser rotulada com uma expressão da seguinte forma:
 - evento (lista-parâmetros)[guarda] / ação



Projeto de Classes

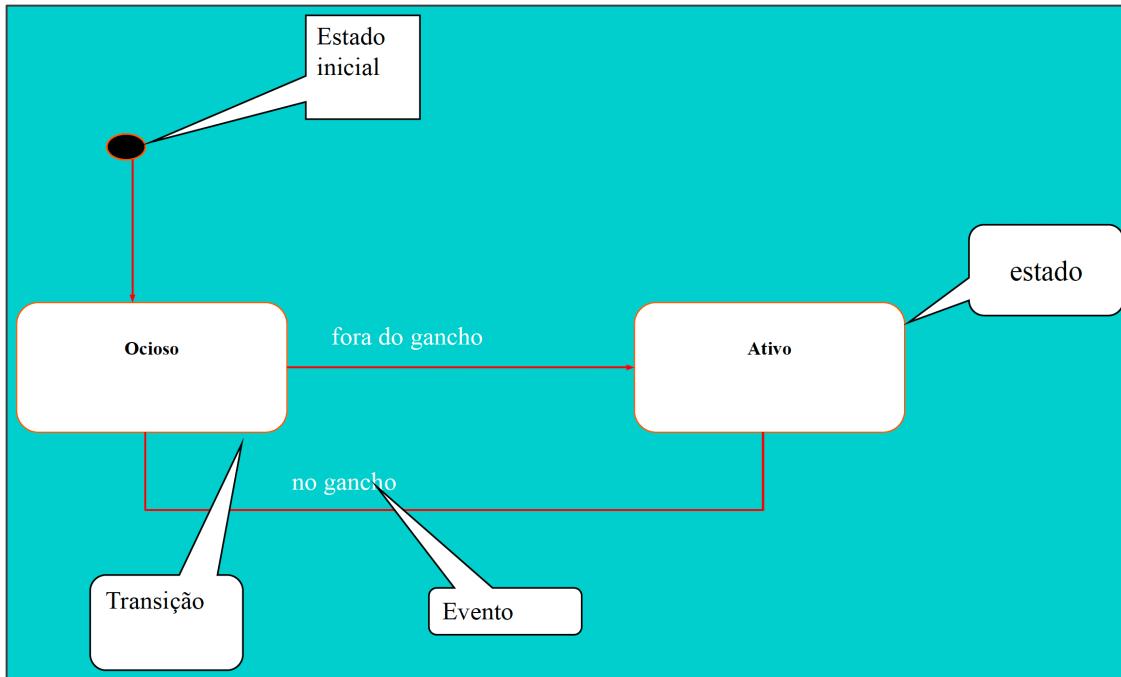
Eventos

- Uma transição possui um evento associado.
- Um evento é algo que acontece em algum ponto no tempo e que pode modificar o estado de um objeto:
 - Pedido realizado
 - Fatura paga
 - Cheque devolvido
- Os eventos relevantes a um sistema de software podem ser classificados em nos seguintes tipos.
 - 1. Evento de chamada: recebimento de uma mensagem de outro objeto.
 - 2. Evento de sinal: recebimento de um sinal.
 - 3. Evento temporal: passagem de um intervalo de tempo predefinido.
 - 4. Evento de mudança: uma condição que se torna verdadeira.



Projeto de Classes

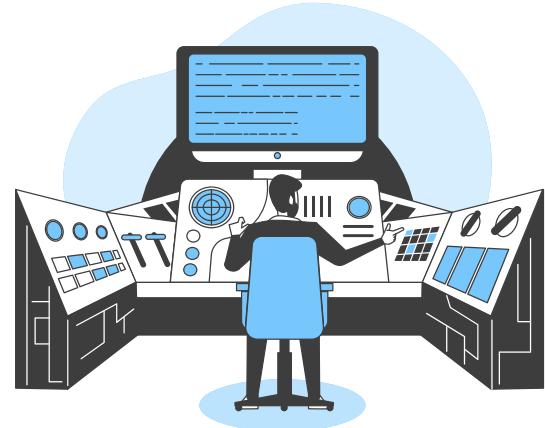
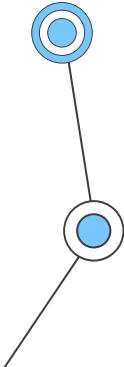
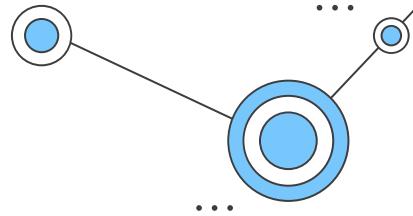
Diagrama de Estados - Exemplo 1



Projeto de Classes

Evento de chamada

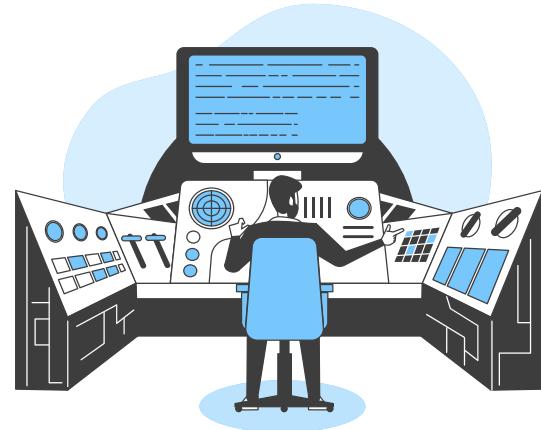
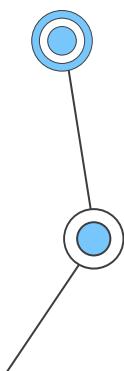
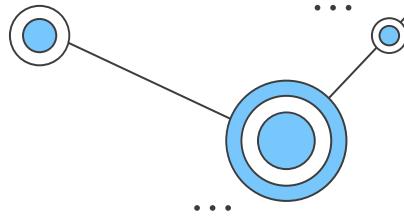
- Corresponde ao recebimento de uma mensagem de outro objeto.
- Pode-se pensar neste tipo de evento como uma solicitação de serviço de um objeto a outro.



Projeto de Classes

Evento de sinal

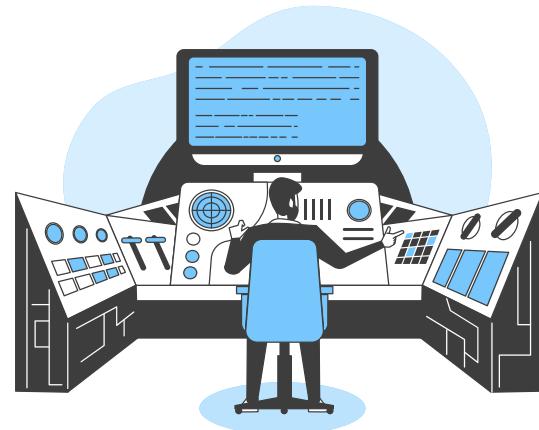
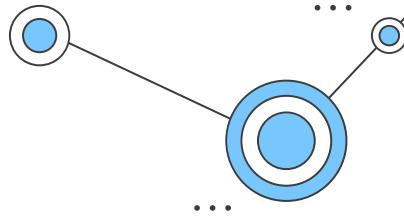
- Neste evento o objeto recebe um sinal de outro objeto que pode fazê-lo mudar de estado.
- A diferença básica entre o evento de sinal e o evento de chamada é que neste último o objeto que envia a mensagem fica esperando a execução da mesma.
 - No evento de sinal, o objeto remetente continua o seu processamento após ter enviado o sinal.



Projeto de Classes

Evento temporal

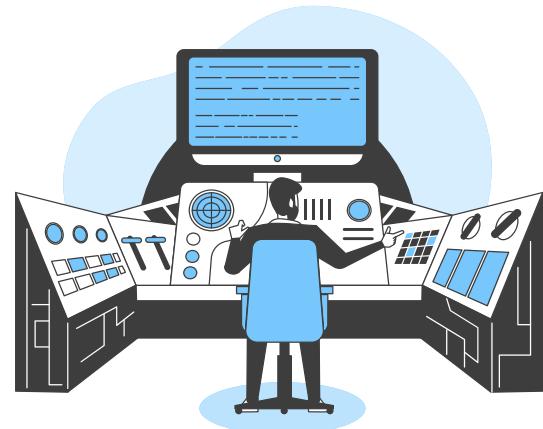
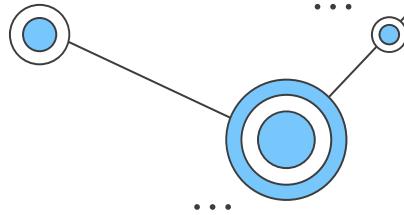
- Corresponde à passagem de um intervalo de tempo predefinido.
- Ao invés de receber uma mensagem específica, um objeto pode interpretar a passagem de um certo intervalo de tempo como sendo um evento.
- Um evento temporal é especificado com a cláusula after seguida de um parâmetro que especifica um intervalo de tempo.
 - `after(30 segundos)`: indica que a transição correspondente será disparada 30 segundos após o objeto ter entrado no estado atual.



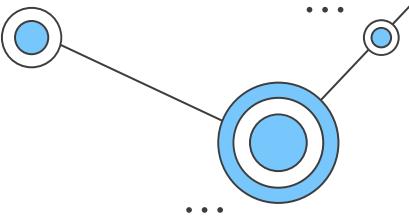
Projeto de Classes

Evento de mudança

- Corresponde a uma condição que se torna verdadeira.
- É representado por uma expressão de valor lógico (verdadeiro ou falso) e é especificado utilizando-se a cláusula when.
 - `when(saldo > 0)`: significa que a transição é disparada quando o valor do atributo saldo for positivo.
- Eventos temporais também podem ser definidos utilizando-se a cláusula when.
 - `when(data = 28/07/2024)`
 - `when(horário = 00:00h)`

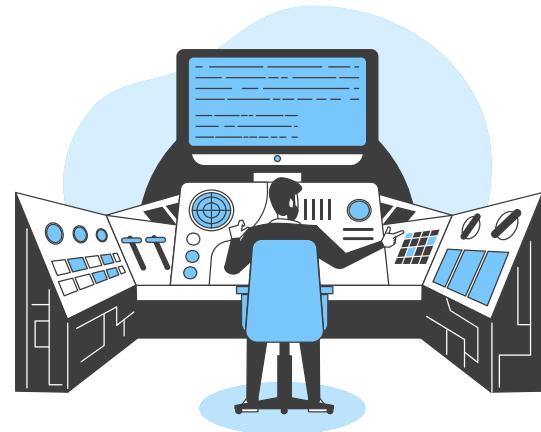


Projeto de Classes



Evento resultando em eventos

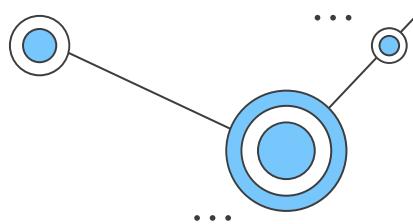
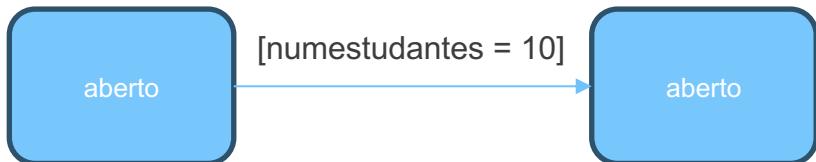
- A ocorrência de um evento relevante pode ocasionar a ocorrência de outro evento.
- No exemplo a seguir, além da transição de estados, o evento OutroEvento (relevante a objetoAlvo)



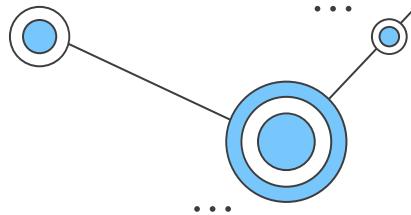
Projeto de Classes

Condição de guarda

- É uma expressão de valor lógico que condiciona o disparo de uma transição.
- A transição correspondente é disparada se e somente se o evento associado ocorre e a condição de guarda é verdadeira.
 - Uma transição que não possui condição de guarda é sempre disparada quando o evento ocorre.
- A condição de guarda pode ser definida utilizando-se parâmetros passados no evento e também atributos e referências a ligações da classe em questão.

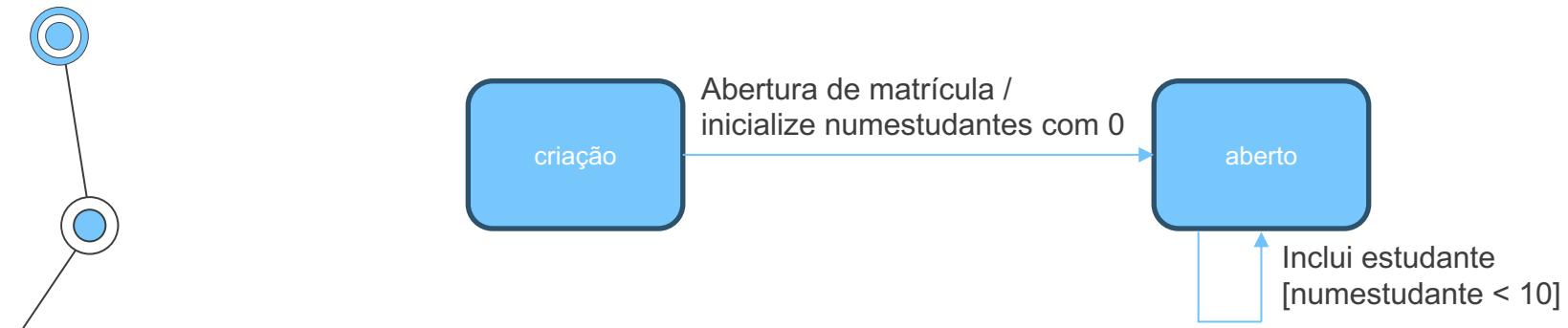


Projeto de Classes



Ações

- Ao transitar de um estado para outro, um objeto pode realizar uma ou mais ações.
- Uma ação é uma expressão definida em termo dos atributos, operações, associações da classe ou dos parâmetros do evento também podem ser utilizados.
- A ação associada a uma transição é executada se e somente se a transição for disparada.
- Leva um tempo pequeno para ser executada
- São mostradas na seta de transição precedidos por uma barra



Projeto de Classes

Atividades

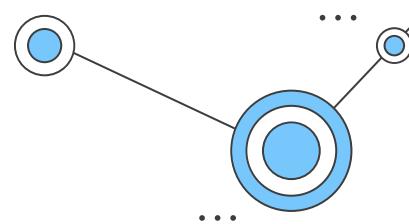
- Semelhantes a ações, atividades são algo que deve ser executado.
- No entanto, uma atividade pode ser interrompida (uma ação não pode).
 - Por exemplo, enquanto a atividade estiver em execução, pode acontecer um evento que a interrompa.
- Outra diferença: uma atividade sempre está associada a um estado (ao contrário, uma ação está associada a uma transição).

fechado
faça: emita e-mail para os alunos matriculados

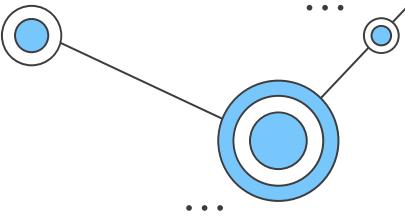
Projeto de Classes

Cláusulas

- No compartimento adicional de um retângulo de estado podem-se especificar ações ou atividades a serem executadas.
- Sintaxe geral: **evento / [ação | atividade]**
- Há três cláusulas predefinidas: entry,exit,do
- Cláusula **exit**
 - Pode ser usada para especificar uma ação a ser realizada no momento em que o objeto entra em um estado.
 - A ação desta cláusula é sempre executada, independentemente do estado do qual o objeto veio.
 - ✓ É como se a ação especificada estivesse associada a todas as transições de entrada no estado.

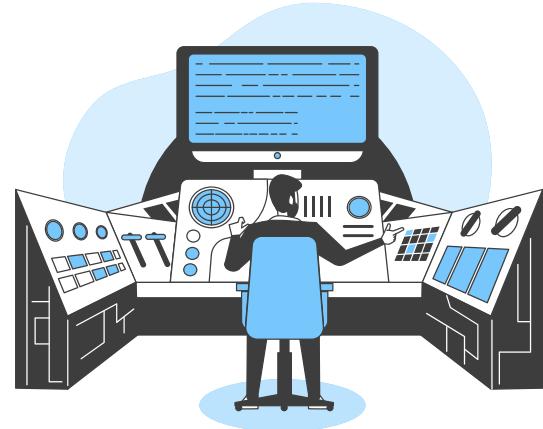
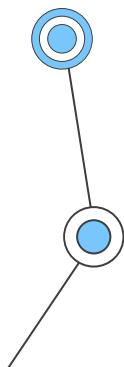


Projeto de Classes



Cláusula entry

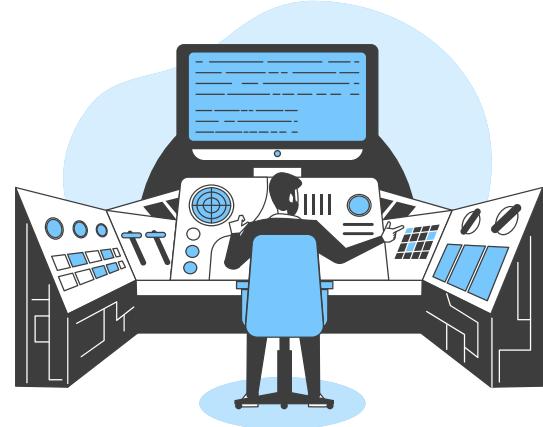
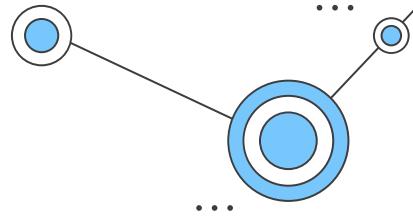
- Pode ser usada para especificar uma ação a ser realizada no momento em que o objeto entra em um estado.
- A ação desta cláusula é sempre executada, independentemente do estado do qual o objeto veio.
 - É como se a ação especificada estivesse associada a todas as transições de entrada no estado.



Projeto de Classes

Cláusula exit

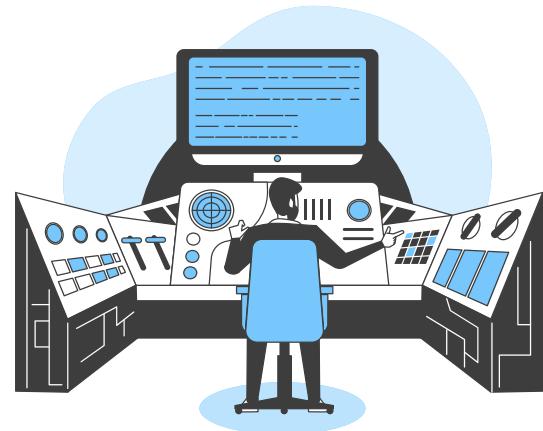
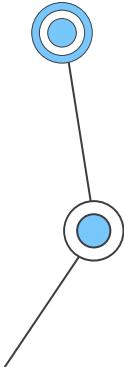
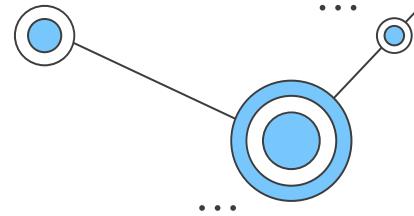
- Serve para declarar ações que são executadas sempre que o objeto sai de um estado.
- É sempre executada, independentemente do estado para o qual o objeto vai.
 - É como se a ação especificada estivesse associada a todas as transições de saída do estado.



Projeto de Classes

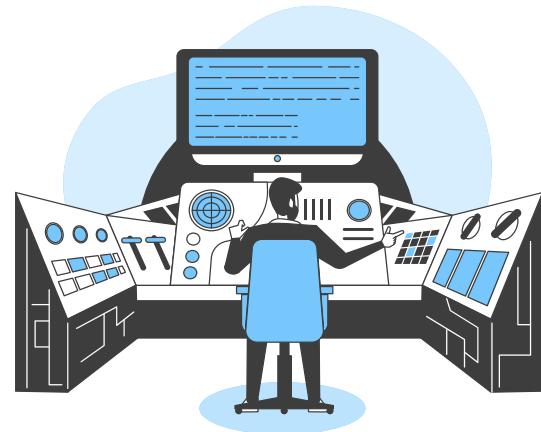
Cláusula do

- Usada para definir alguma atividade a ser executada quando o objeto passa para um determinado estado.
- Ao contrário da cláusula entry, serve para especificar uma atividade ao invés de uma ação.



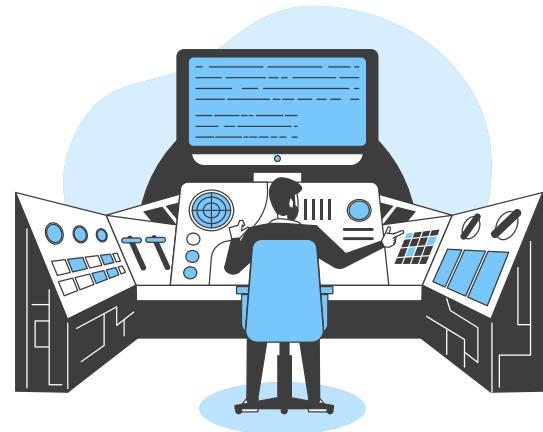
Projeto de Classes

Cláusula do - exemplo



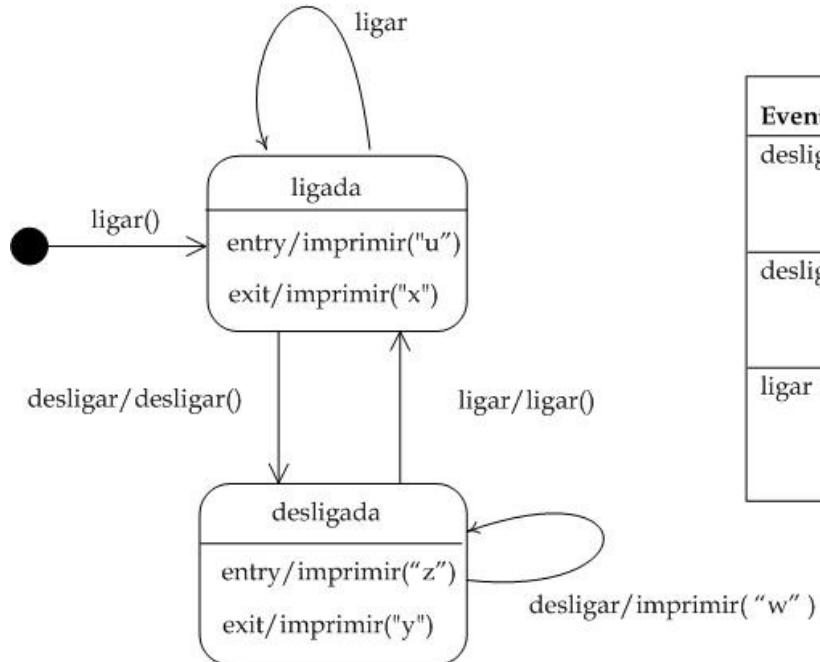
Projeto de Classes

Cláusula entry e exit - exemplo



Projeto de Classes

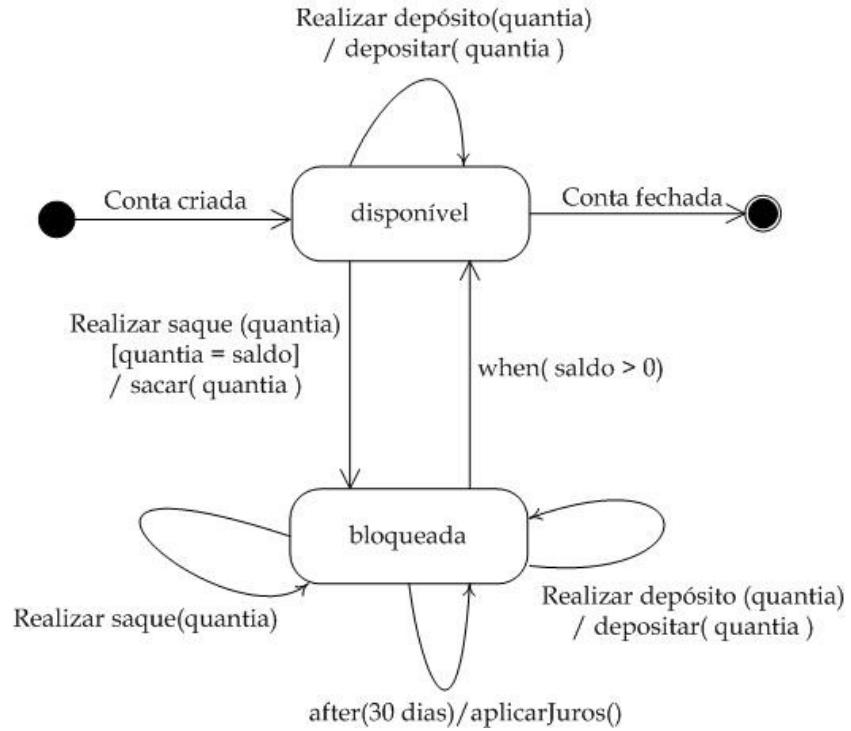
Cláusula do - exemplo



Evento	Ações executadas
desligar	imprimir("x") desligar() imprimir("z")
desligar	imprimir("y") imprimir("w") imprimir("z")
ligar	imprimir("y") ligar() imprimir("u")

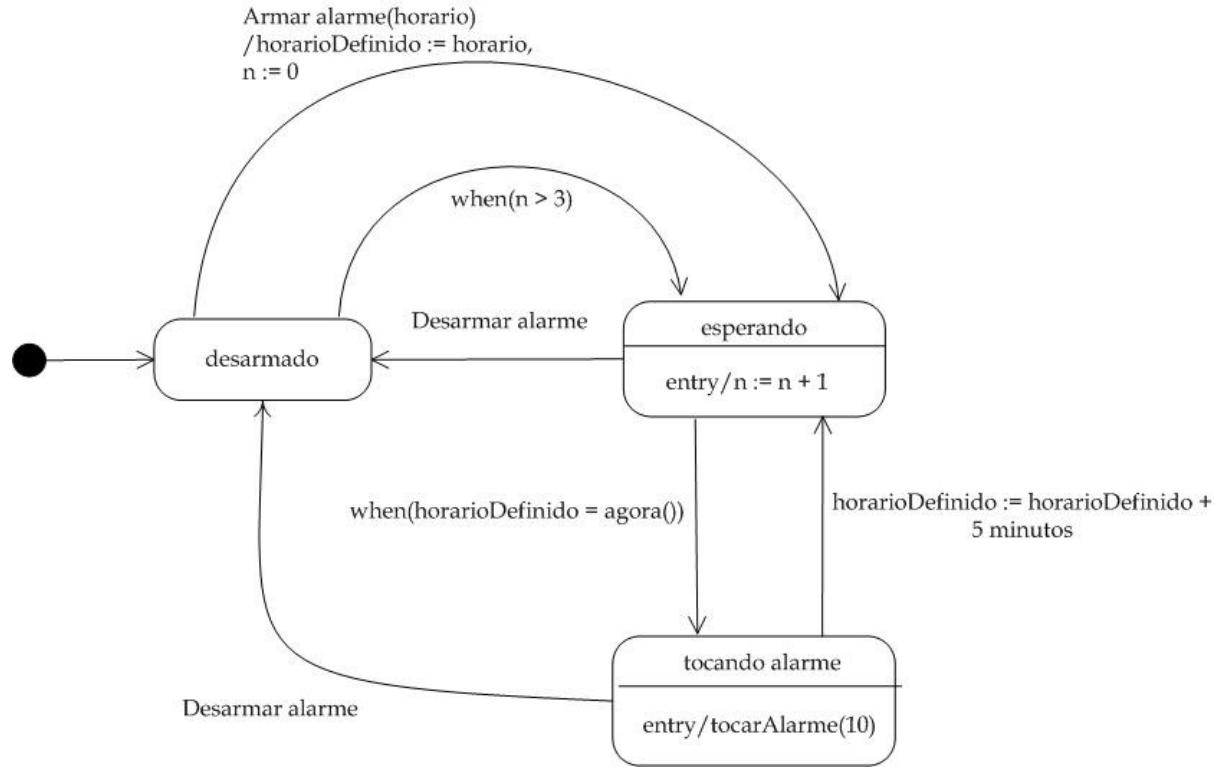
Projeto de Classes

Exemplo (ContaBancária)



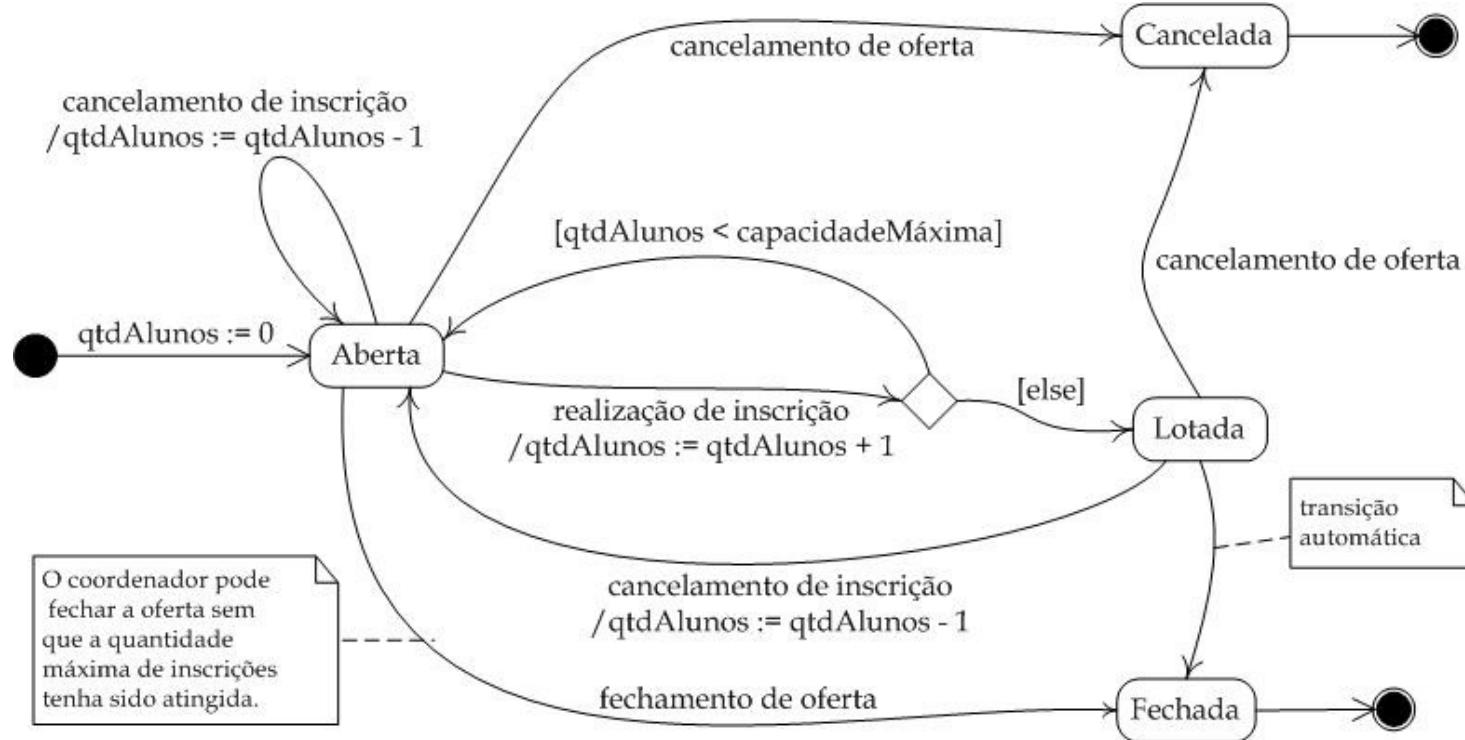
Projeto de Classes

Exemplo (Despertador)



Projeto de Classes

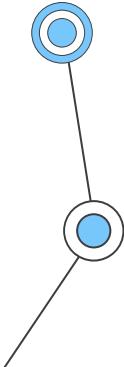
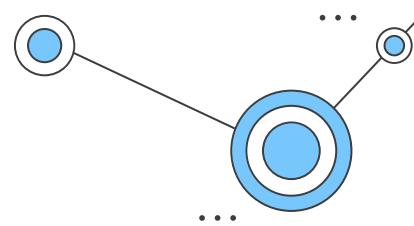
Exemplo (OfertaDisciplina)



Projeto de Classes

Ponto de junção

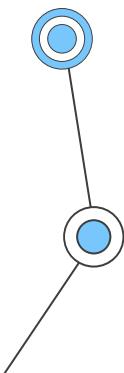
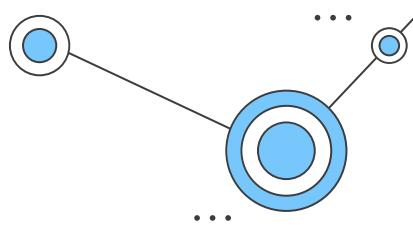
- Pode ser que o próximo estado de um objeto varie de acordo com uma condição.
 - Se o valor da condição for verdadeiro, o objeto vai para um estado E1; se o valor for falso, o objeto vai para outro estado E2.
 - É como se a transição tivesse bifurcações, e cada transição de saída da bifurcação tivesse uma condição de guarda.
- Essa situação pode ser representada em um DTE através de um **ponto de junção**.
- Pontos de junção permitem que duas ou mais transições compartilhem uma “trajetória de transições”.



Projeto de Classes

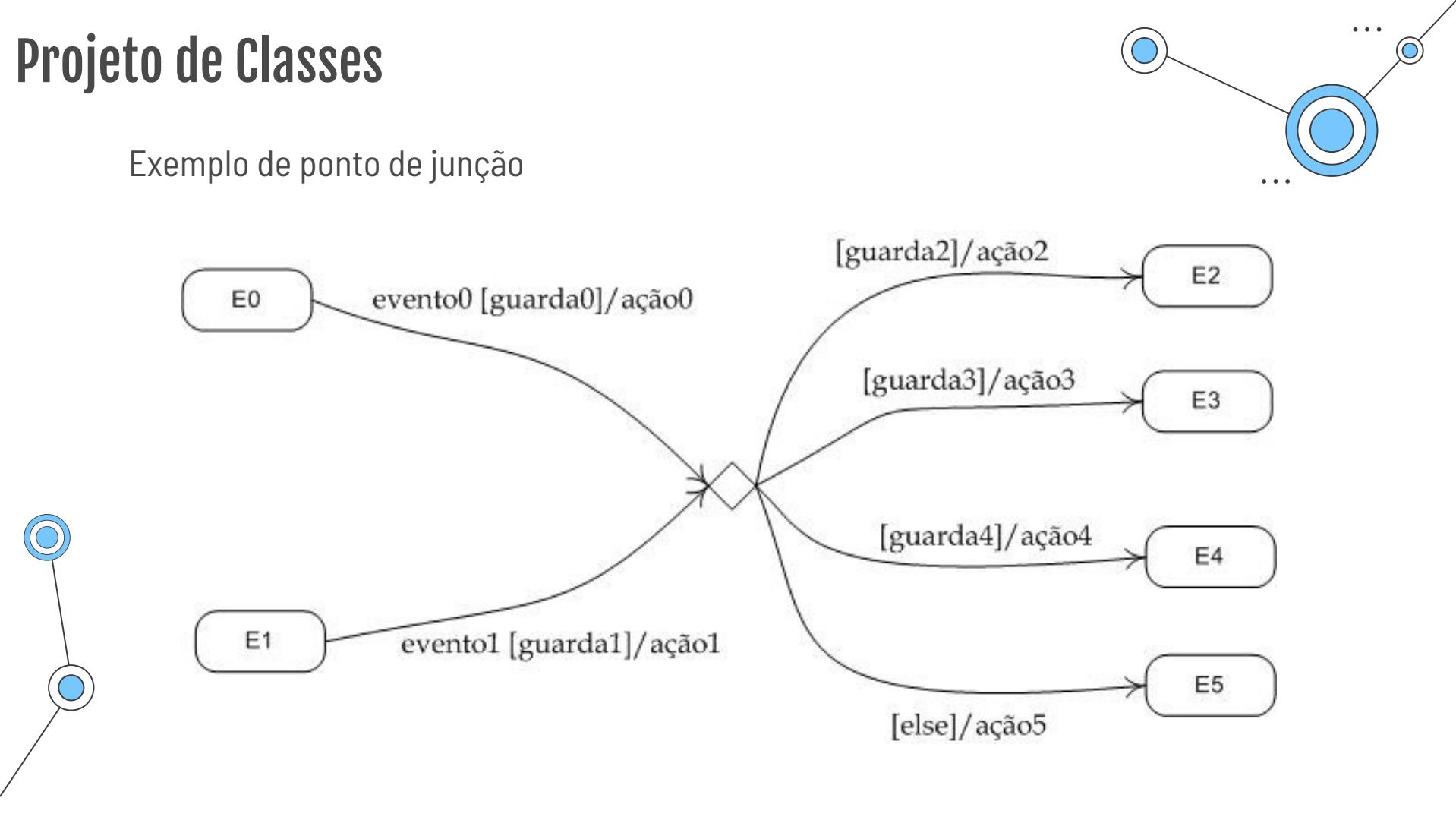
Ponto de junção

- De uma forma geral, pode haver um número ilimitado de transições saindo de um ponto de junção.
- Pode haver também uma transição de saída que esteja rotulada com a cláusula else.
 - Se as outras condições forem falsas, a transição da cláusula else é disparada.
- Pontos de junção permitem que duas ou mais transições compartilhem uma “trajetória de transições”.
- De uma forma geral, pode haver um número ilimitado de transições saindo de um ponto de junção.
- Pode haver também uma transição de saída que esteja rotulada com a cláusula else.
 - Se as outras condições forem falsas, a transição da cláusula else é disparada.



Projeto de Classes

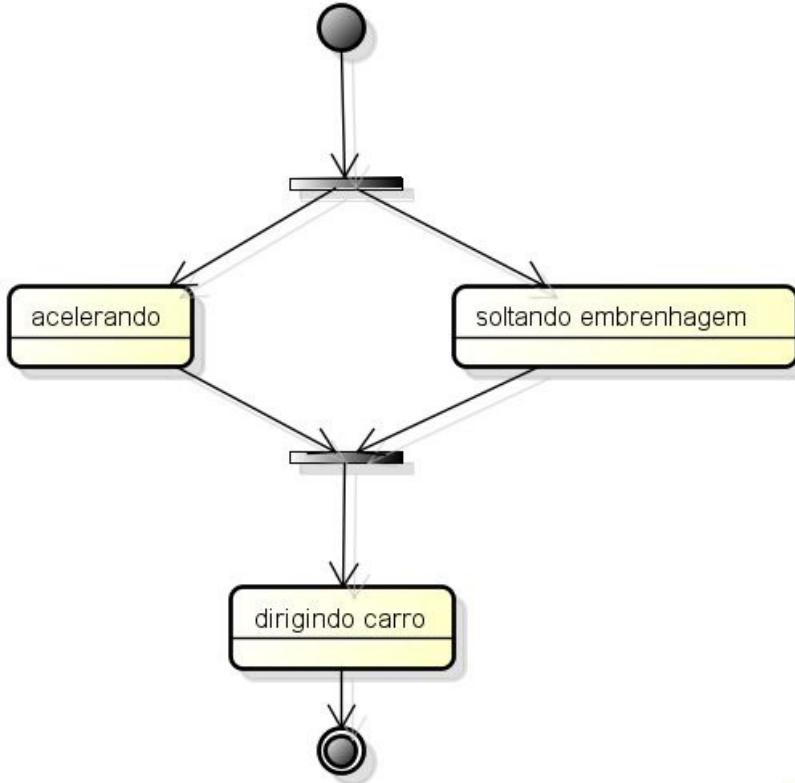
Exemplo de ponto de junção



Projeto de Classes

Barra de bifurcação/união

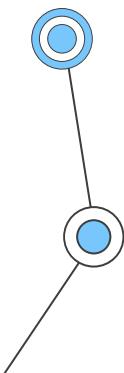
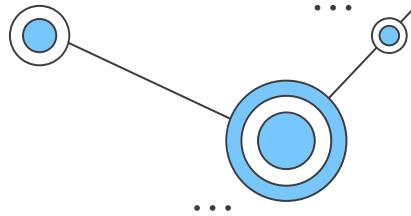
- Utilizada quando da ocorrência de estados paralelos.



Projeto de Classes

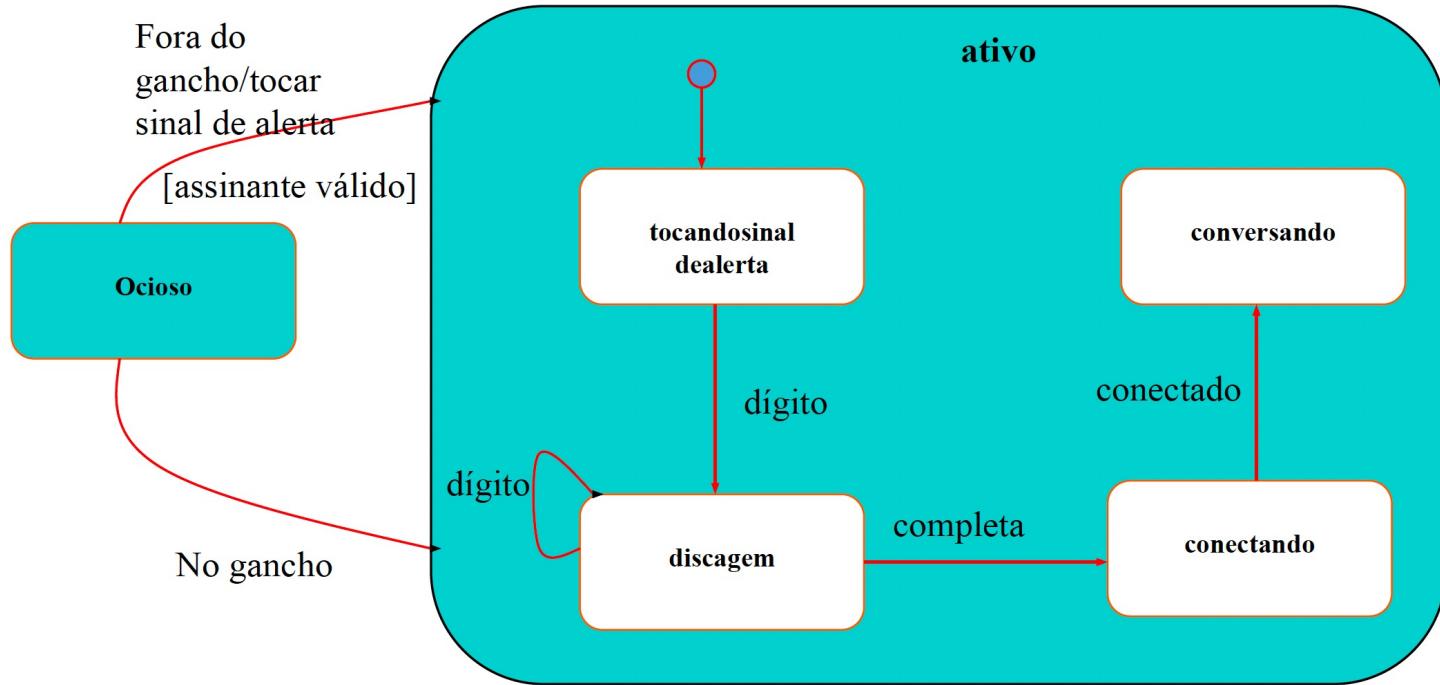
Diagrama de Estados - Estados Aninhados

- Diagramas de estado podem ficar complexos e difíceis de gerenciar.
- Estados podem ser usados para simplificar diagramas complexos.
- Um superestado é um estado que encerra estados aninhados chamados subestados.
- Transições comuns dos subestados são representadas no nível do superestado.
- É permitido qualquer número de níveis de aninhamento.
- Estados aninhados podem levar a uma redução substancial da complexidade gráfica, permitindo modelar problemas maiores e mais complexos.



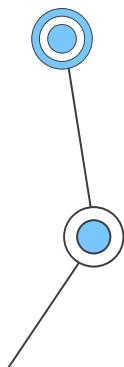
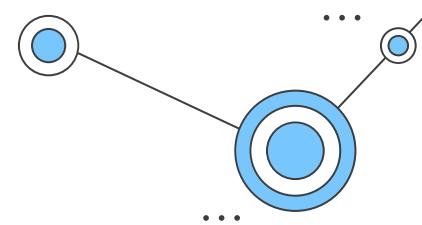
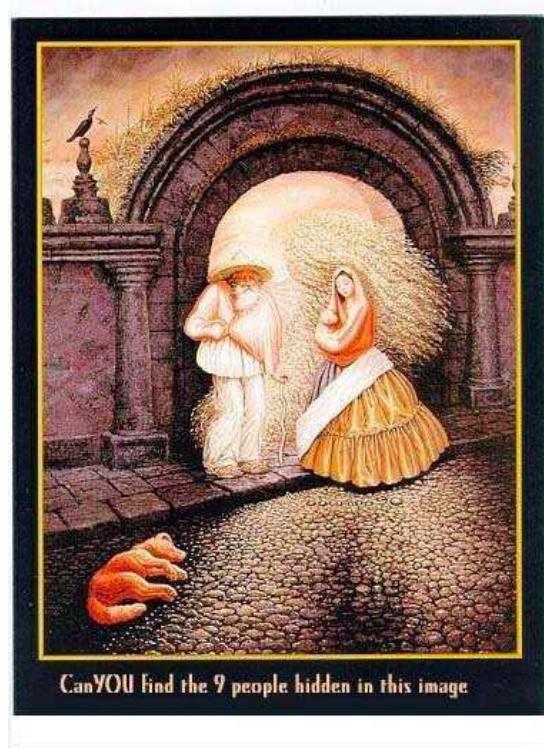
Projeto de Classes

Diagrama de Estados - Estados Aninhados



Projeto de Classes

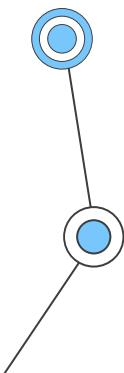
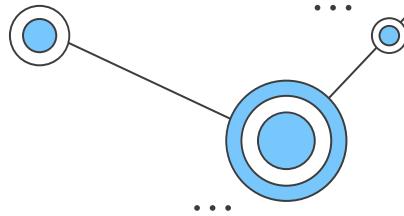
Identificação dos elementos de um diagrama de estados



Projeto de Classes

Identificação de elementos do DTE

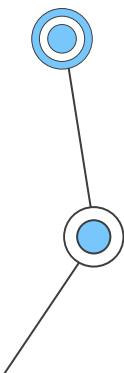
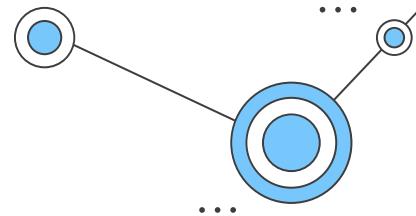
- Um bom ponto de partida para identificar estados é analisar os possíveis valores de seus atributos e as ligações que ele pode realizar com outros objetos.
- No entanto, a existência de atributos ou ligações não é suficiente para justificar a criação de um DTE.
 - O comportamento de objetos dessa classe deve depender de tais atributos ou ligações.



Projeto de Classes

Identificação de elementos do DTE

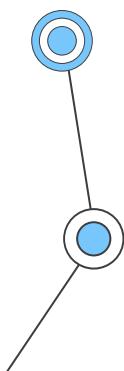
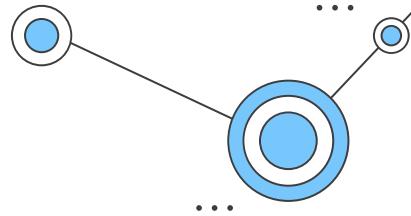
- Já que transições dependem de eventos para ocorrer, devem-se identificar estes eventos primeiramente.
- Além disso, deve-se examinar também se há algum fator que condicione o disparo da transição.
 - Se existir, este fator deve ser modelado como uma condição de guarda da transição.
- Um bom ponto de partida para identificar eventos é a descrição dos casos de uso.
- Os eventos encontrados na descrição dos casos de uso são externos ao sistema.
- Contudo, uma transição pode também ser disparada por um evento interno ao sistema.



Projeto de Classes

Identificação de elementos do DTE: transições

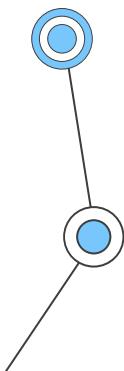
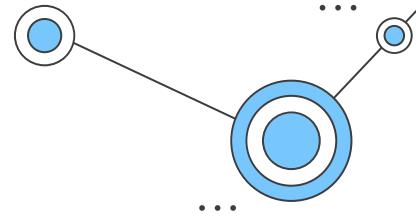
- Os eventos encontrados na descrição dos casos de uso são externos ao sistema.
- Contudo, uma transição pode também ser disparada por um evento interno ao sistema.
- Para identificar os eventos internos, analise os diagramas de interação.
 - mensagens podem ser vistas como eventos.
- De uma forma geral, cada operação com visibilidade pública de uma classe pode ser vista como um evento em potencial.



Projeto de Classes

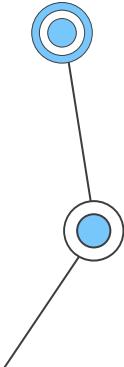
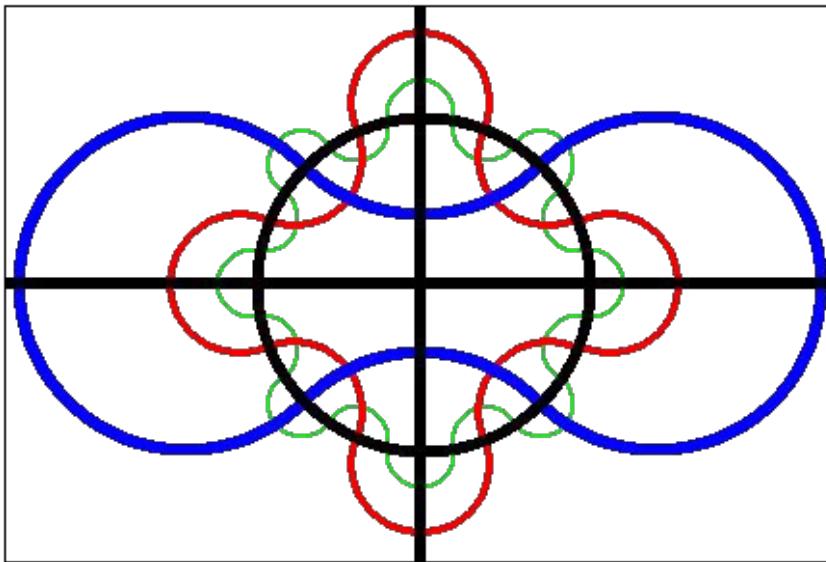
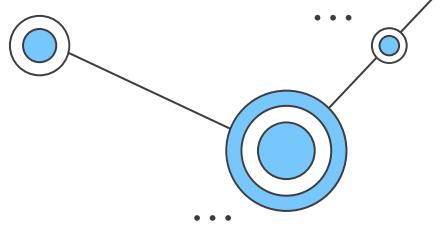
Identificação de elementos do DTE: transições

- Uma outra fonte para identificação de eventos associados a transições é analisar as regras de negócio.
 - “Um cliente do banco não pode retirar mais de R\$ 1.000 por dia de sua conta”.
 - “Os pedidos para um cliente não especial devem ser pagos antecipadamente”.
 - “O número máximo de alunos por curso é igual a 30”.



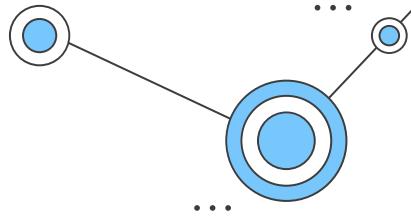
Projeto de Classes

Construção de diagramas de transição de estados



Projeto de Classes

Um DTE para uma classe

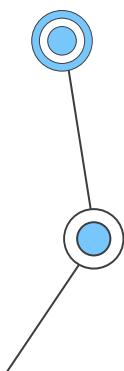


Os diagramas de estados são desenhados por classe.

- Desvantagem: dificuldade na visualização do estado do sistema como um todo.
- Essa desvantagem é parcialmente compensada pelos diagramas de interação.

Nem todas as classes de um sistema precisam de um DTE.

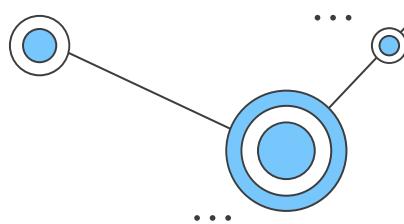
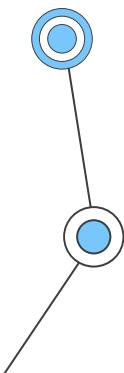
- Somente classes que exibem um comportamento dinâmico relevante.
- Objetos cujo histórico precisa ser rastreado pelo sistema são típicos para se construir um diagrama de estados.



Projeto de Classes

Procedimento para construção

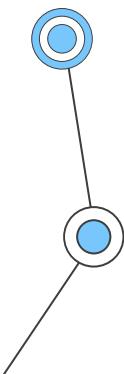
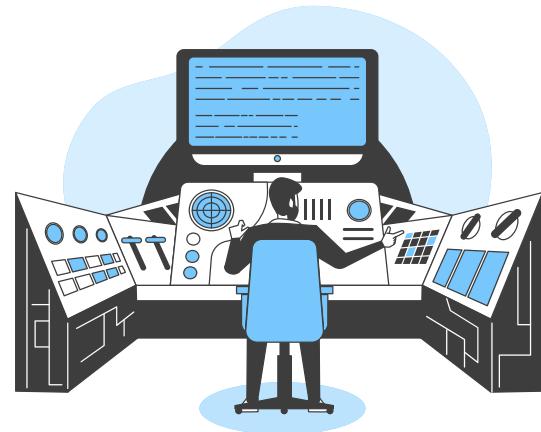
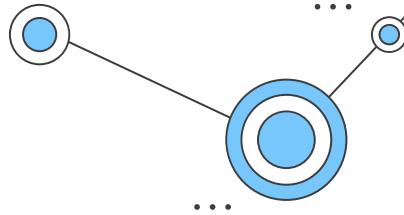
- Identifique os estados relevantes para a classe.
- Identifique os eventos relevantes. Para cada evento, identifique qual a transição que ele ocasiona.
- Para cada estado: identifique as transições possíveis quando um evento ocorre.
- Para cada estado, identifique os eventos internos e ações correspondentes.
- Para cada transição, verifique se há fatores que influenciam no seu disparo. (definição de condições de guarda e ações).
- Para cada condição de guarda e para cada ação, identifique os atributos e ligações que estão envolvidos.
- Defina o estado inicial e os eventuais estados finais.
- Desenhe o DTE.



Projeto de Classes

Informações para o modelo de classes

- A construção de um DTE frequentemente leva à descoberta de novos atributos para uma classe
 - principalmente atributos para servirem de abstrações para estados.
- Além disso, este processo de construção permite identificar novas operações na classe
 - pois os objetos precisam reagir aos eventos que eles recebem.



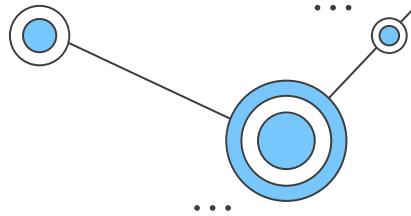
Projeto de Classes

Modelagem de estados no processo de desenvolvimento de software

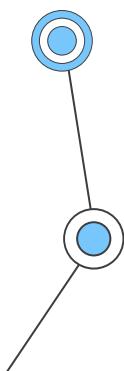
- Os DTEs podem ser construídos com base nos diagramas de interação e nos diagramas de classes.
- Durante a construção do DTE para uma classe, novos atributos e operações podem surgir.
 - Essas novas propriedades devem ser adicionadas ao modelo de classes.
- A construção de um DTE frequentemente leva à descoberta de novos atributos para uma classe
 - principalmente atributos para servirem de abstrações para estados.
- Além disso, este processo de construção permite identificar novas operações na classe
 - pois os objetos precisam reagir aos eventos que eles recebem.

Projeto de Classes

Modelagem de estados no processo de desenvolvimento de software



- O comportamento de um objeto varia em função do estado no qual ele se encontra.
- Pode ser necessária a atualização de uma ou mais operações de uma classe para refletir o comportamento do objetos em cada estado.
- Por exemplo, o comportamento da operação sacar() da classe ContaBancária varia em função do estado no qual esta classe se encontra
 - saques não podem ser realizados em uma conta que esteja no estado bloqueada.

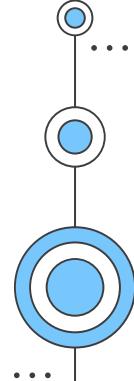




Projeto de Software

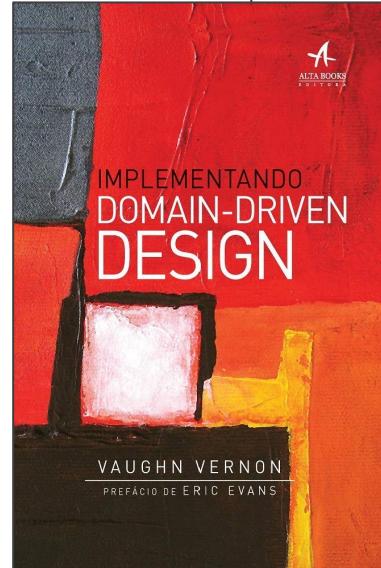
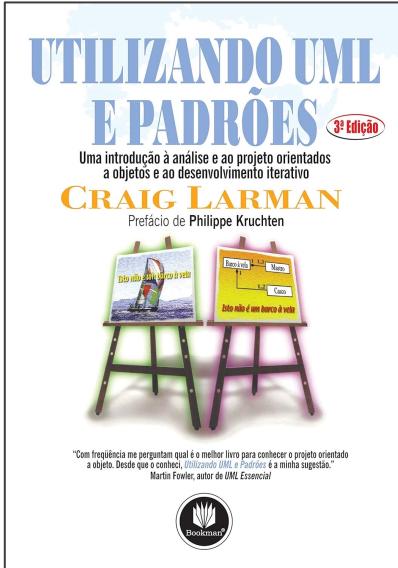
Referências básicas:

- **ACM TRANSACTIONS ON SOFTWARE ENGINEERING AND METHODOLOGY.** New York, N.Y., USA: Association for Computing Machinery, 1992-. Trimestral. ISSN 1049-331X. Disponível em: <https://dl.acm.org/toc/tosem/1992/1/2>. Acesso em: 19 jul. 2024. (Periódico On-line).
- LARMAN, Craig. **Utilizando UML e padrões:** uma introdução á análise e ao projeto orientados a objetos e desenvolvimento iterativo. 3. ed. Porto Alegre: Bookman, 2007. E-book. ISBN 9788577800476. (Livro Eletrônico).
- SILVEIRA, Paulo et al. **Introdução à arquitetura e design de software:** uma visão sobre a plataforma Java. Rio de Janeiro, RJ: Elsevier, Campus, 2012. xvi, 257 p. ISBN 9788535250299. (Disponível no Acervo).
- VERNON, Vaughn. **Implementando o Domain-Driven Design.** Rio de Janeiro, RJ: Alta Books, 2016. 628 p. ISBN 9788576089520. (Disponível no Acervo).



Projeto de Software

Referências básicas:



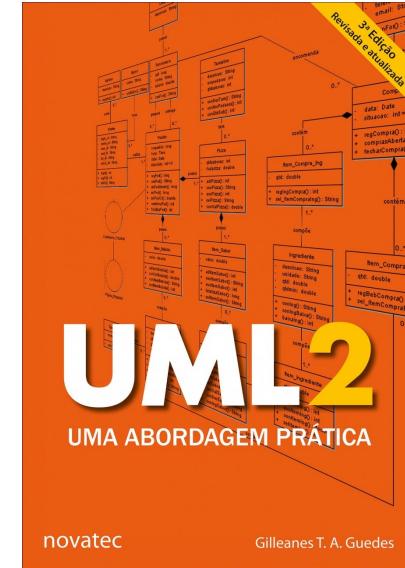
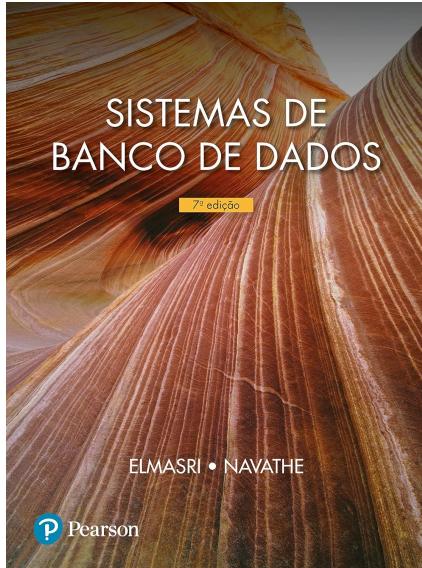
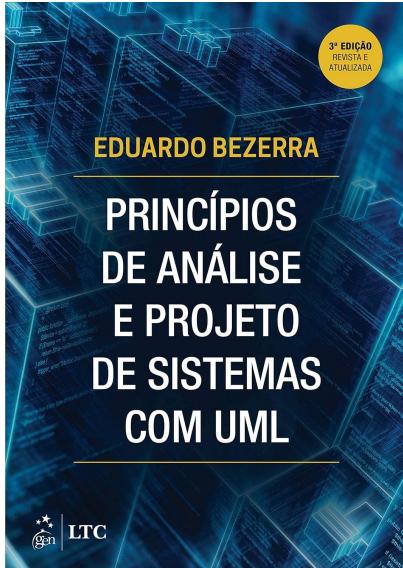
Projeto de Software

Referências complementares:

- BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. 3. ed. rev. e atual. Rio de Janeiro: Elsevier, 2015. xvii, 398 p. ISBN 9788535226263. (Disponível no Acervo).
- ELMASRI, Ramez; Navathe, Shamkant B. **Sistemas de banco de dados**, 7^a ed. Editora Pearson 1152 ISBN 9788543025001. (Livro Eletrônico).
- GUEDES, Gilleanes T. A. **UML 2**: uma abordagem prática. 2. ed. São Paulo: Novatec, c2011. 484 p. ISBN 9788575222812. (Disponível no Acervo).
- **IEEE TRANSACTIONS ON SOFTWARE ENGINEERING**. New York: IEEE Computer Society, 1975-. Mensal,. ISSN 0098-5589. Disponível em: <https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=32>. Acesso em: 19 jul. 2024. (Periódico On-line).
- SOMMERVILLE, Ian. **Engenharia de software**. 10. ed. São Paulo: Pearson Education do Brasil, c2019. xii, 756 p. ISBN 9788543024974. (Disponível no Acervo).
- WAZLAWICK, Raul Sidnei. **Análise e design orientados a objetos para sistemas de informação**: modelagem com UML, OCL e IFML. 3. ed. Rio de Janeiro, RJ: Elsevier, Campus, c2015. 462 p. ISBN 9788535279849. (Disponível no Acervo).

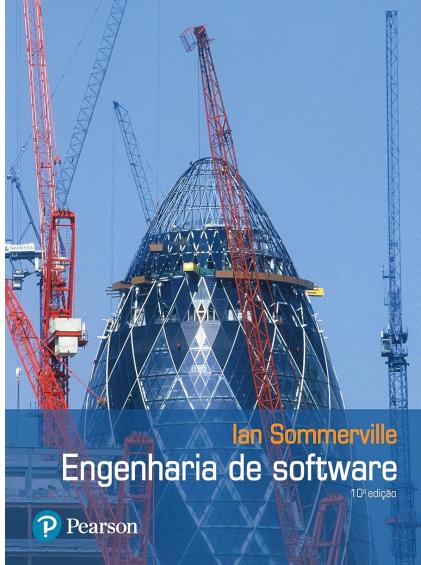
Projeto de Software

Referências complementares:



Projeto de Software

Referências complementares:



Obrigado!

Dúvidas?

joaopauloaramuni@gmail.com



[GitHub](#)



[LinkedIn](#)



[Lattes](#)

...



...