DIAGRAMA DE SEQUÊNCIA

Análise Orientada a Objetos

Prof. Rafael Tápia

O QUE É UM DIAGRAMA DE SEQUENCIA

O diagrama de sequência é um tipo de diagrama usado para **mostrar a troca de mensagens entre objetos** ao longo do tempo. Ele é parte da UML e ajuda a representar o **comportamento dinâmico** de um sistema.

Por que usar?

- Para visualizar a **ordem** em que as coisas acontecem.
- Ele ajuda a ver o "**vai e vem**" da comunicação entre partes do sistema.

Imagine um chat online: quando você manda uma mensagem, ela vai do seu app para o servidor, do servidor para o destinatário, e você espera a resposta. O diagrama de sequência pode mostrar exatamente esse "**bate-volta**".

COMPONENTES BÁSICOS

- **Objetos/Atores:** Representam as **entidades** envolvidas na interação (como um usuário, um servidor, ou até mesmo outro sistema).
  - Eles são desenhados como **caixas** no topo do diagrama.
- **Lifeline (Linha de Vida):** É uma linha vertical que parte de cada objeto e mostra por quanto tempo ele está "vivo" no processo.
  - Imagine uma linha que mostra o tempo de vida de cada "personagem" dessa história.
- **Mensagens:** Representam as **comunicações** entre os objetos, como chamadas de métodos ou dados sendo passados.
  - São desenhadas como **setas** de um objeto para outro, mostrando quem fala com quem.
- **Ativação:** É a "barrinha" vertical sobre a linha de vida de um objeto, que indica quando aquele objeto está **executando alguma ação**.

Exemplo Rápido: Imagine que você está comprando algo em uma loja virtual. Os atores seriam: o cliente, o sistema de pagamentos e o banco. O cliente manda a solicitação de pagamento (mensagem), o sistema de pagamentos processa (ativação), e o banco aprova ou não (resposta).

COMPONENTES BÁSICOS

Objetos/Atores:

Representam as **entidades** envolvidas na interação (como um usuário, um servidor, ou até mesmo outro sistema).

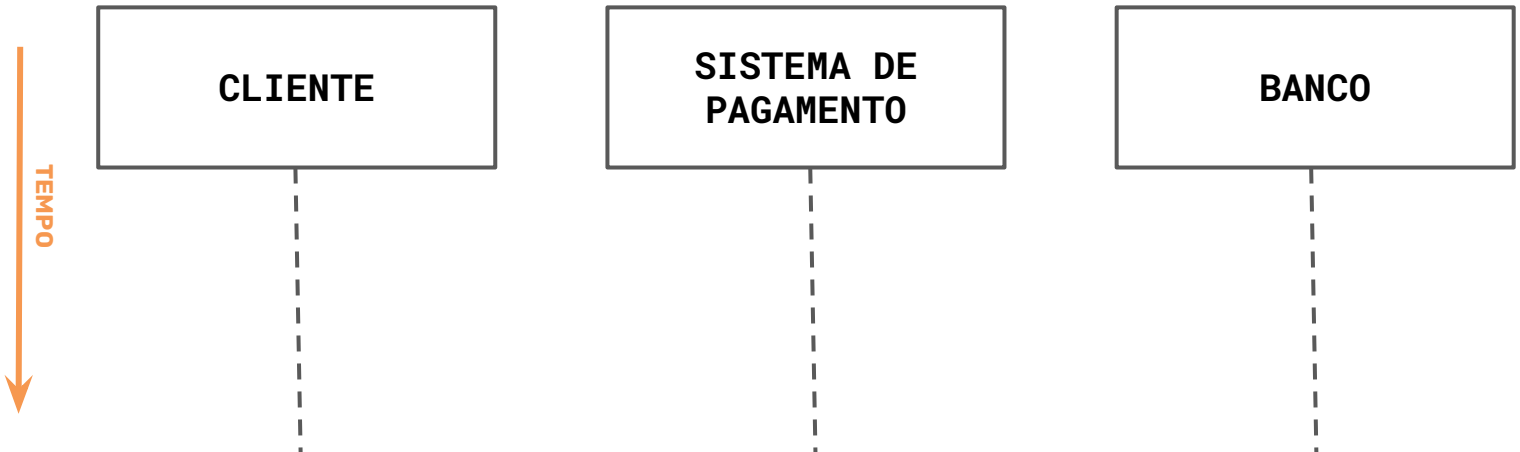
Eles são desenhados como **caixas** no topo do diagrama.

COMPONENTES BÁSICOS

Lifeline (Linha de Vida):

É uma linha vertical que parte de cada objeto e mostra por quanto tempo ele está "vivo" no processo.

Imagine uma linha que mostra o tempo de vida de cada "**personagem**" dessa história.

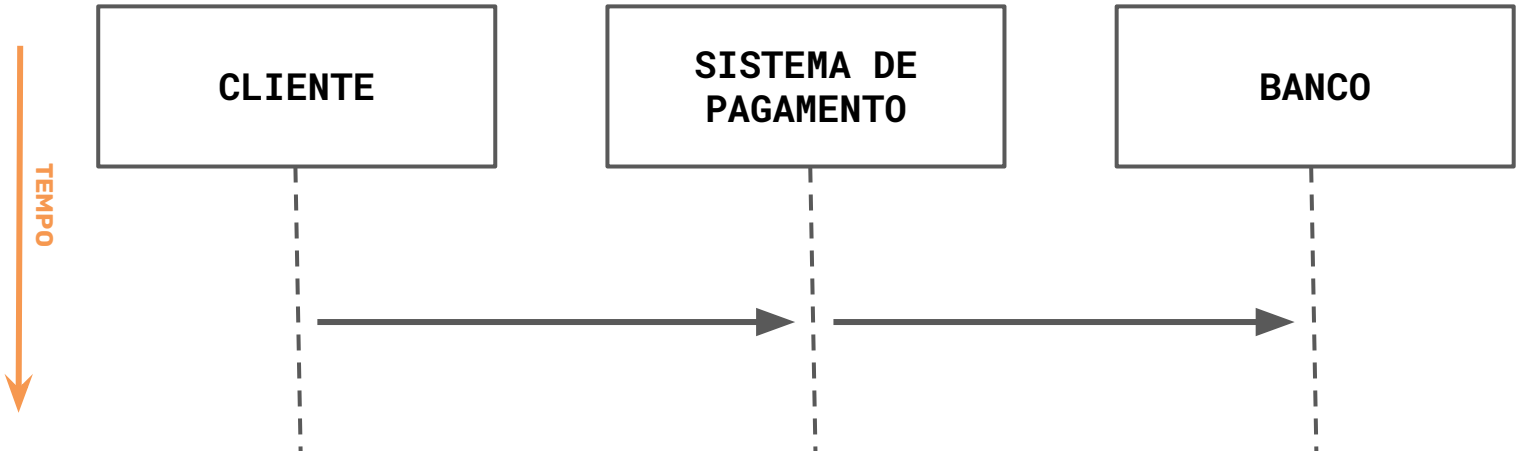


COMPONENTES BÁSICOS

Mensagens:

Representam as comunicações entre os objetos, como chamadas de métodos ou dados sendo passados.

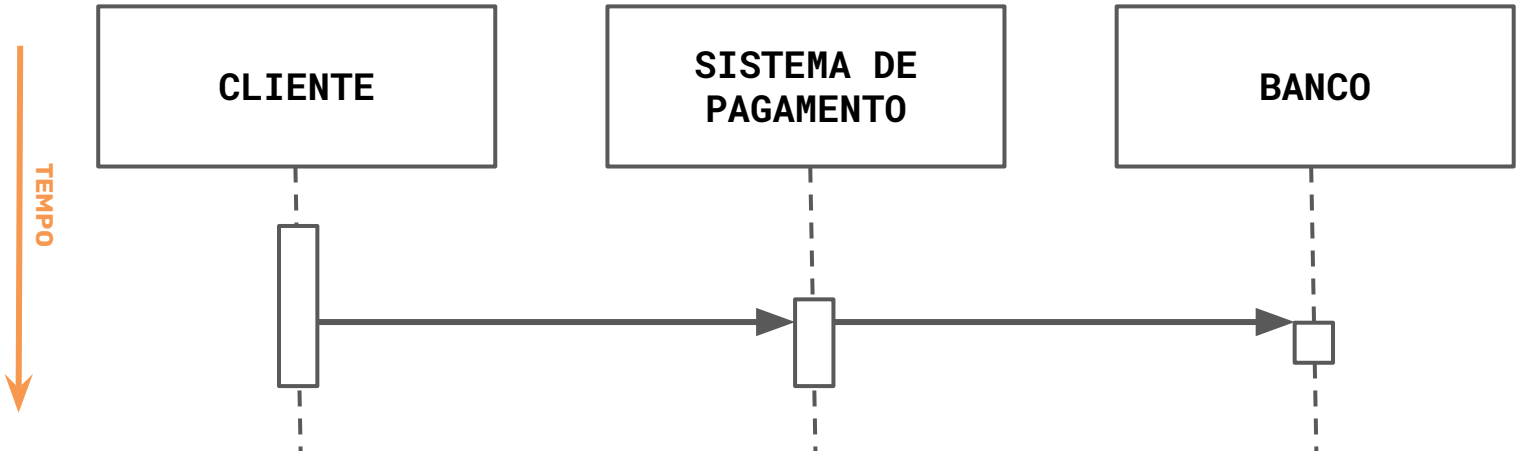
São desenhadas como setas de um objeto para outro, mostrando quem fala com quem.



COMPONENTES BÁSICOS

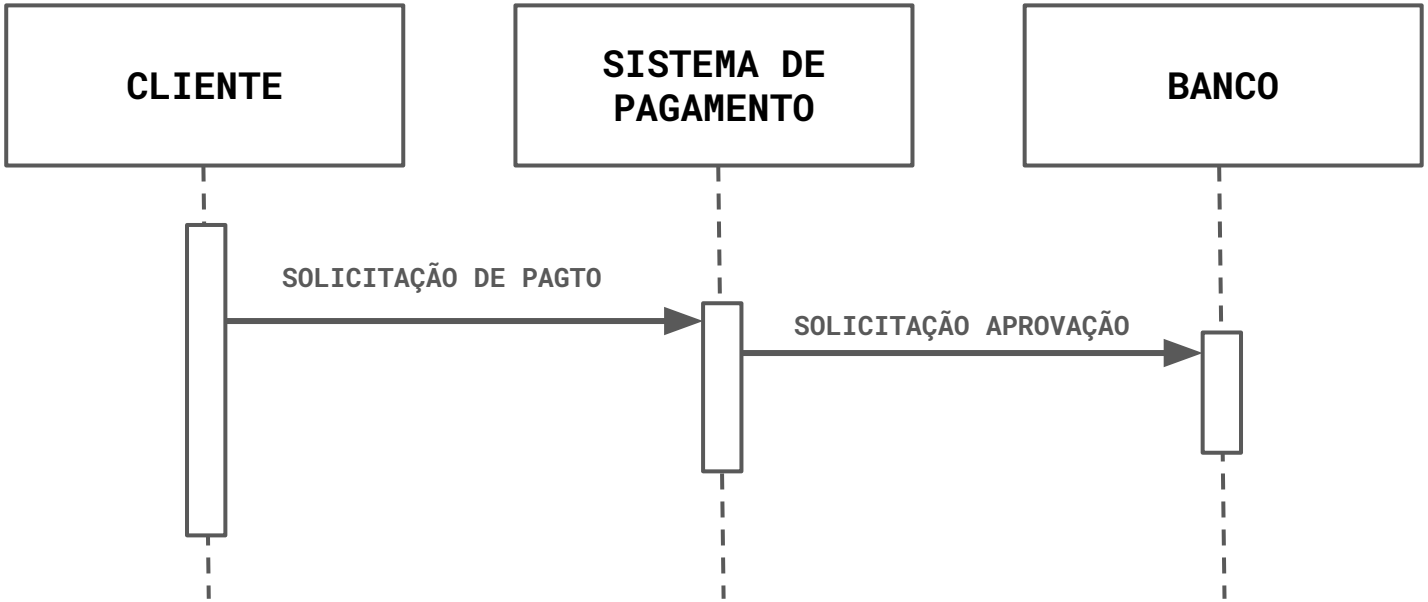
Ativação:

É a "**barrinha**" vertical sobre a linha de vida de um objeto, que indica quando aquele objeto está executando alguma ação.



EXEMPLO

Imagine que você está comprando algo em uma loja virtual. Os atores seriam: o cliente, o sistema de pagamentos e o banco. O cliente manda a solicitação de pagamento (mensagem), o sistema de pagamentos processa (ativação), e o banco aprova ou não (resposta).



MONTANDO UM DIAGRAMA

Vamos praticar com um modelo simples, um **cenário de login** em um site:

- **Atores:** Usuário e Sistema.
- **Passos:**
    1. O usuário insere o login e senha.
    2. O sistema verifica as credenciais no banco de dados.
    3. Se estiver tudo certo, o sistema retorna a confirmação de login.
    4. O usuário recebe a mensagem de que está logado.

Passo a passo:

1. Desenhe dois objetos no topo: "Usuário" e "Sistema".
2. Crie as **lifelines** (linhas de vida) descendo de cada um.
3. Desenhe uma **mensagem** (seta) do Usuário para o Sistema: "Inserir login e senha".
4. Adicione a **ativação** no Sistema (a barra vertical sobre a linha de vida).
5. Desenhe uma **mensagem** do Sistema para ele mesmo (ou para outro objeto, como "Banco de Dados"): "Verificar credenciais".
6. Finalize com a resposta do Sistema para o Usuário: "Login bem-sucedido".

DICAS E BOAS PRÁTICAS

Seja claro:
O diagrama de sequência deve ser fácil de entender. Use nomes simples e representações diretas.

Não exagere no detalhe:
Mostre as mensagens mais importantes para evitar um diagrama muito complexo.

Representação temporal:
O tempo vai de cima para baixo no diagrama, então tudo o que acontece depois deve estar mais abaixo.

EXERCÍCIOS

Exercício 1:
Processo de Compra Online

Cenário: Um cliente está comprando um produto em uma loja virtual. O processo consiste em:

1. O cliente seleciona um produto e adiciona ao carrinho.
2. O cliente realiza o checkout.
3. O sistema da loja verifica o estoque.
4. O sistema de pagamento processa o pagamento.
5. O sistema confirma a compra e envia uma notificação para o cliente.

Instruções:

- Desenhe os atores: **Cliente**, **Sistema da Loja** e **Sistema de Pagamento**.
- Crie as mensagens e interações entre os atores, mostrando cada passo do processo.

Exercício 2:
Solicitação de Amizade em Rede Social

Cenário: Um usuário envia uma solicitação de amizade a outro usuário em uma rede social. O fluxo é o seguinte:

1. O usuário A envia uma solicitação de amizade.
2. O sistema verifica se o usuário B existe.
3. O sistema envia a solicitação ao usuário B.
4. O usuário B aceita ou rejeita a solicitação.
5. O sistema atualiza o status de amizade.

Instruções:

- Desenhe os atores: **Usuário A**, **Sistema**, **Usuário B**.
- Crie as interações mostrando a troca de mensagens e a resposta do Usuário B.

Exercício 3:
Acesso a Caixa de E-mail

Cenário: Um usuário está acessando sua caixa de e-mail. O fluxo consiste em:

1. O usuário entra no sistema de e-mail com login e senha.
2. O sistema de e-mail valida as credenciais com o banco de dados.
3. Se válido, o sistema exibe os e-mails na caixa de entrada.

Instruções:

- Desenhe os atores: **Usuário**, **Sistema de E-mail**, **Banco de Dados**.
- Mostre as interações, desde o login até a exibição dos e-mails.

Exercício 4:
Retirada de Dinheiro no Caixa Eletrônico

Cenário: Um cliente vai a um caixa eletrônico e realiza um saque. O processo é o seguinte:

1. O cliente insere o cartão e digita a senha.
2. O sistema do caixa verifica a senha e consulta o saldo.
3. O sistema verifica se há saldo suficiente.
4. Se o saldo for suficiente, o sistema realiza o saque e dispensa o dinheiro.

Instruções:

- Desenhe os atores: **Cliente**, **Caixa Eletrônico**, **Sistema Bancário**.
- Modele as mensagens que envolvem a verificação do saldo e a execução do saque.

Exercício 5:
Envio de SMS

Cenário: Um usuário envia um SMS para outro usuário. O processo é:

1. O usuário A cria e envia a mensagem SMS.
2. O sistema de mensagens recebe o SMS e envia para o número do destinatário.
3. O usuário B recebe o SMS.

Instruções:

- Desenhe os atores: **Usuário A**, **Sistema de Mensagens**, **Usuário B**.
- Modele as mensagens entre A, o sistema e B.

AVANÇANDO...

MENSAGENS

A seta de mensagem vem com uma **descrição**, que é conhecida como uma assinatura de mensagem, sobre ela. O formato para esta assinatura de mensagem está abaixo. Todas as partes exceto o **nome_da_mensagem** são opcionais.

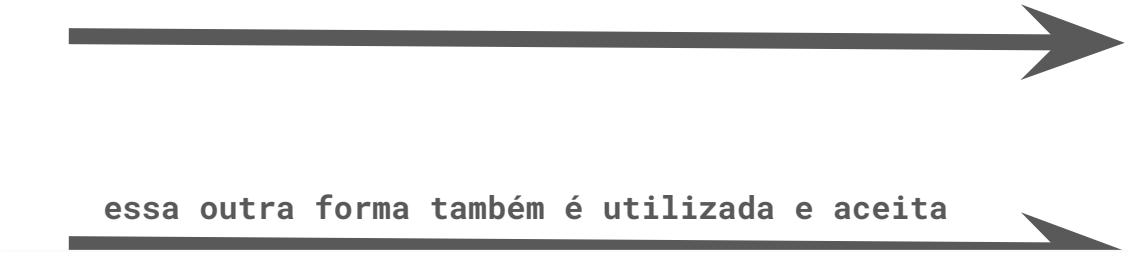
*atributo = **nome_da_mensagem** (argumentos): tipo_de_retorno*

MENSAGENS SÍNCRONAS

Uma mensagem síncrona é usada quando o remetente espera que o receptor processe a mensagem e retorne antes de continuar com outra mensagem. A ponta de seta usada para indicar este tipo de mensagem é sólida, como a que está abaixo.

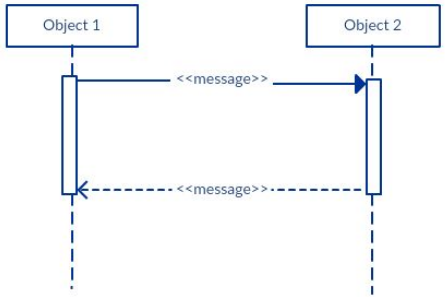
MENSAGENS ASSÍNCRONAS

Uma mensagem assíncrona é usada quando o chamador da mensagem não espera que o receptor processe a mensagem e volte antes de enviar outras mensagens para outros objetos dentro do sistema. A ponta de seta usada para mostrar este tipo de mensagem é uma seta de linha como mostrado no exemplo abaixo



MENSAGENS DE RETORNO

Uma mensagem de retorno é usada para indicar que o receptor da mensagem terminou o processamento da mensagem e está devolvendo o controle para o autor da chamada da mensagem. **As mensagens de retorno são peças opcionais de notação**, para uma barra de ativação que é acionada por uma mensagem **síncrona** implica **sempre uma mensagem de retorno**.

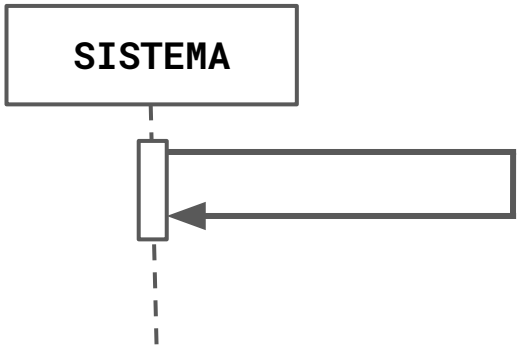


MENSAGEM REFLEXIVA

Quando um objeto envia uma mensagem para si mesmo, ele é chamado de mensagem reflexiva. É indicado com uma seta de mensagem que começa e termina na mesma linha de vida, como mostrado no exemplo abaixo.

Quando usar:

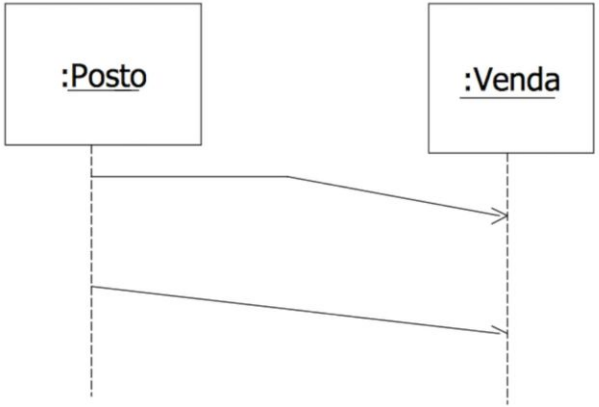
- Quando um objeto executa uma ação que depende de si mesmo (ex: atualizações de estado).
- Para representar métodos recursivos, onde a função chama a si própria.
- Em sistemas que utilizam loops ou verificações internas.



MENSAGEM CONSUMINDO TEMPO EM ATRASO

Refere-se a uma mensagem que inclui um atraso no seu envio ou processamento. Isso indica que, após a mensagem ser enviada, há um tempo de espera antes que a ação seja concluída ou que a resposta seja recebida.

Uso comum: Em sistemas onde há tempos de resposta lentos (como redes, dispositivos externos, ou operações assíncronas).

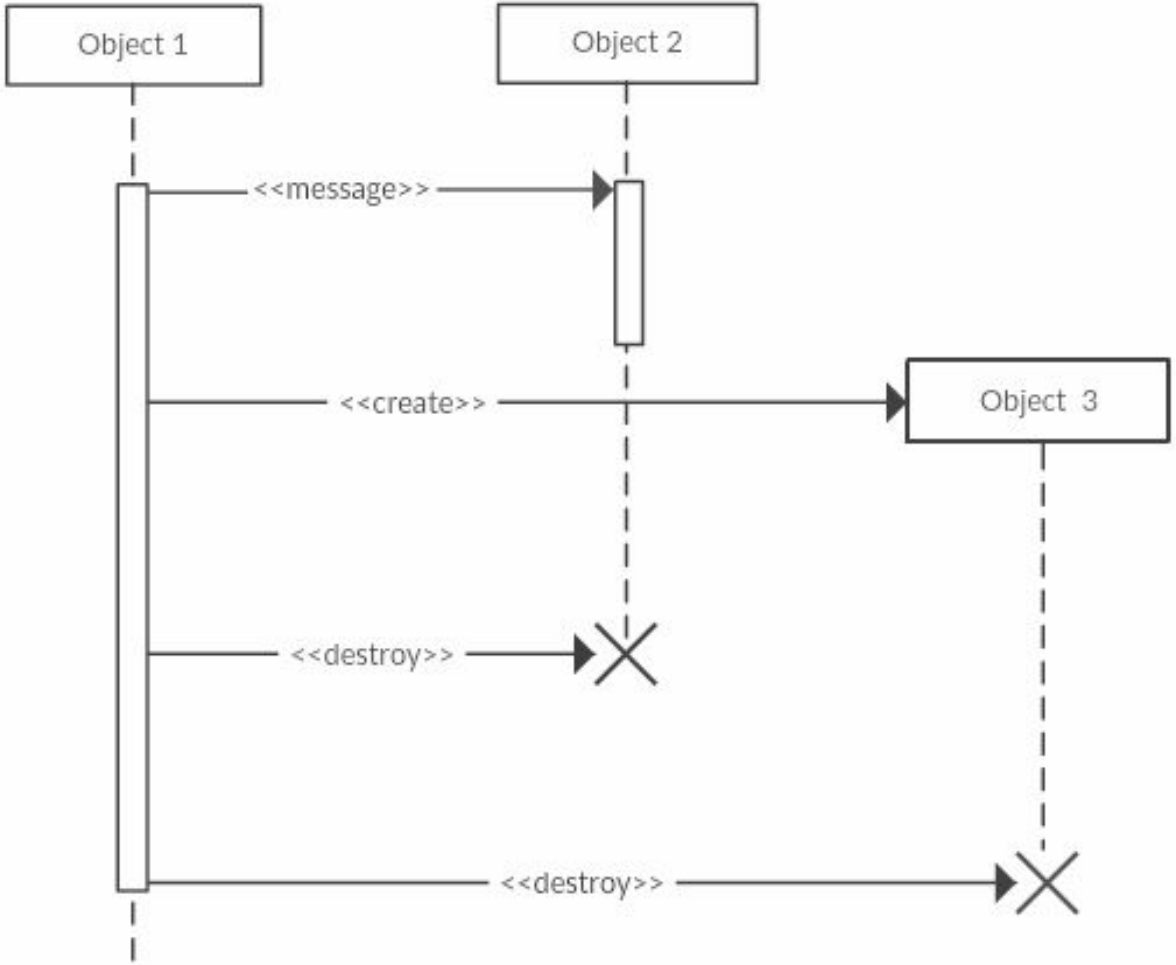


MENSAGENS DE CRIAÇÃO E DESTRUIÇÃO DE PARTICIPANTE

Os objectos não vivem necessariamente durante toda a duração da sequência de eventos. Objetos ou participantes podem ser criados de acordo com a mensagem que está sendo enviada.

A notação pode ser usada quando você precisa mostrar que o participante em particular não existia até que a chamada de criação seja enviada. Se o participante criado faz algo imediatamente após criação, você deve adicionar uma caixa de ativação logo abaixo da caixa do participante.

Da mesma forma, os participantes quando não são mais necessários também podem ser excluídos de um diagrama de seqüência. Isto é feito adicionando um ‘X’ no final da linha de vida do referido participante.


Object 1
Object 2
<<message>>
<<create>>
Object 3
<<destroy>>
<<destroy>>

INTERAÇÕES COMPLEXAS

OPÇÕES

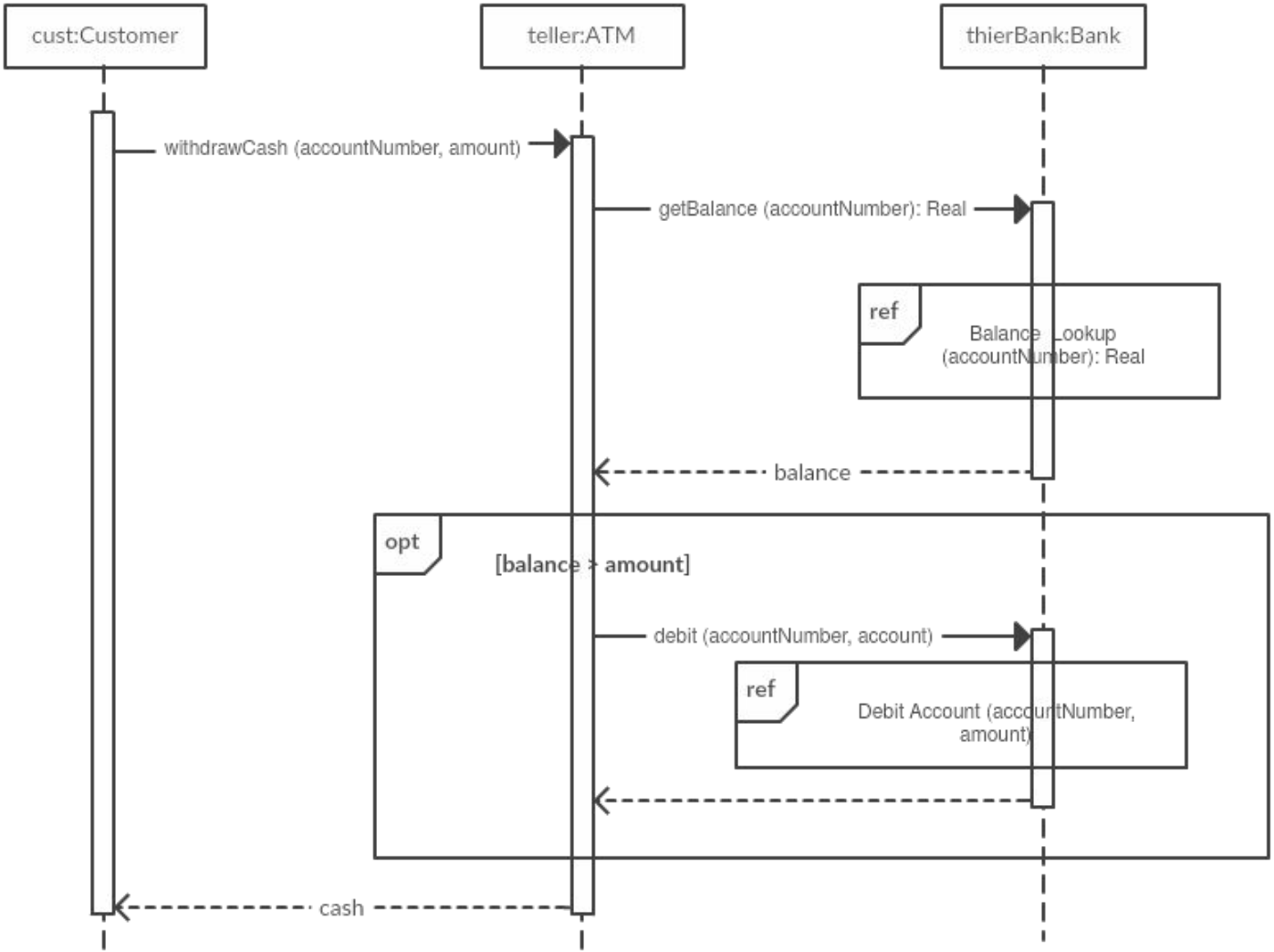
O fragmento de combinação de opções é usado para indicar uma seqüência que só ocorrerá sob uma determinada condição, caso contrário, a seqüência não ocorrerá. Ele modela a afirmação “**if-then**”.

O fragmento de opção é representado com uma moldura retangular onde **‘opt’** é colocado dentro da caixa de nome.

REFERÊNCIA

Você pode usar o fragmento de **referência** para gerenciar o tamanho de grandes diagramas de seqüência. Ele permite **reutilizar parte de um diagrama de seqüência em outro**, ou em outras palavras, você pode **referenciar parte de um diagrama** em outro diagrama usando o fragmento de ref.

Para especificar o fragmento de referência, você tem que mencionar **‘ref’** na caixa de nome do quadro e o nome do diagrama de seqüência que está sendo referido dentro do quadro.

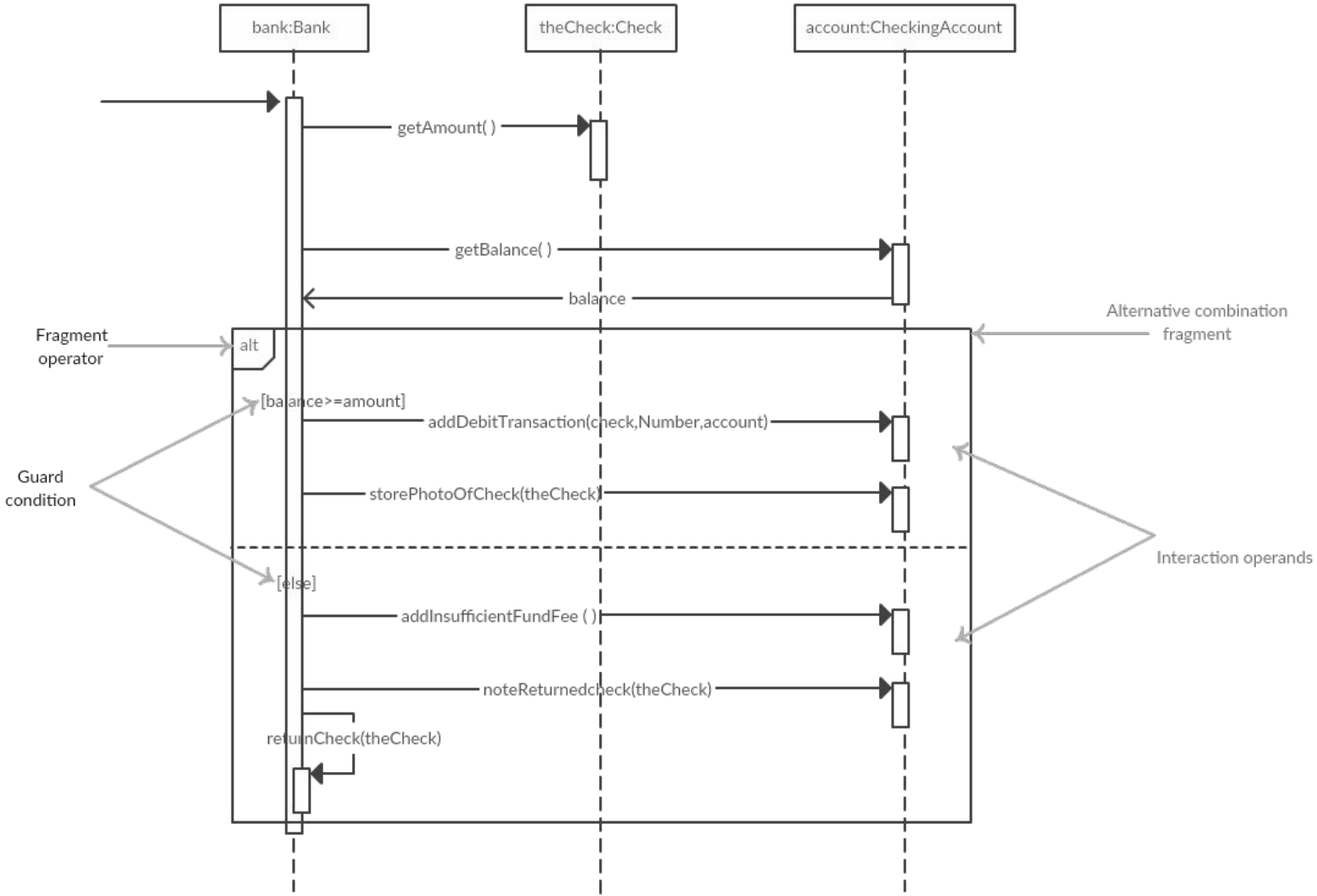

cust:Customer
teller:ATM
thierBank:Bank
withdrawCash (accountNumber, amount)
getBalance (accountNumber): Real
ref
Balance Lookup
(accountNumber): Real
balance
opt
[balance > amount]
debit (accountNumber, account)
ref
Debit Account (accountNumber,
amount)
cash

ALTERNATIVAS

O fragmento de combinação **alternativa** é usado quando é necessário fazer uma escolha entre duas ou mais sequências de mensagens. **Ele modela a lógica do “if-then-else”.**

O fragmento alternativo é representado por um **grande retângulo ou um frame**; ele é especificado mencionando **‘alt’** dentro da caixa de nome do frame.

Para mostrar **duas ou mais alternativas**, o retângulo maior é então dividido no que é chamado de **operandos de interação usando uma linha tracejada**.


bank:Bank
theCheck:Check
account:CheckingAccount
getAmount( )
getBalance( )
balance
Fragment operator
alt
Alternative combination fragment
[balance>=amount]
addDebitTransaction(check,Number,account)
Guard condition
storePhotoOfCheck(theCheck)
Interaction operands
[else]
addInsufficientFundFee ( )
noteReturnedcheck(theCheck)
returnCheck(theCheck)

LOOPS

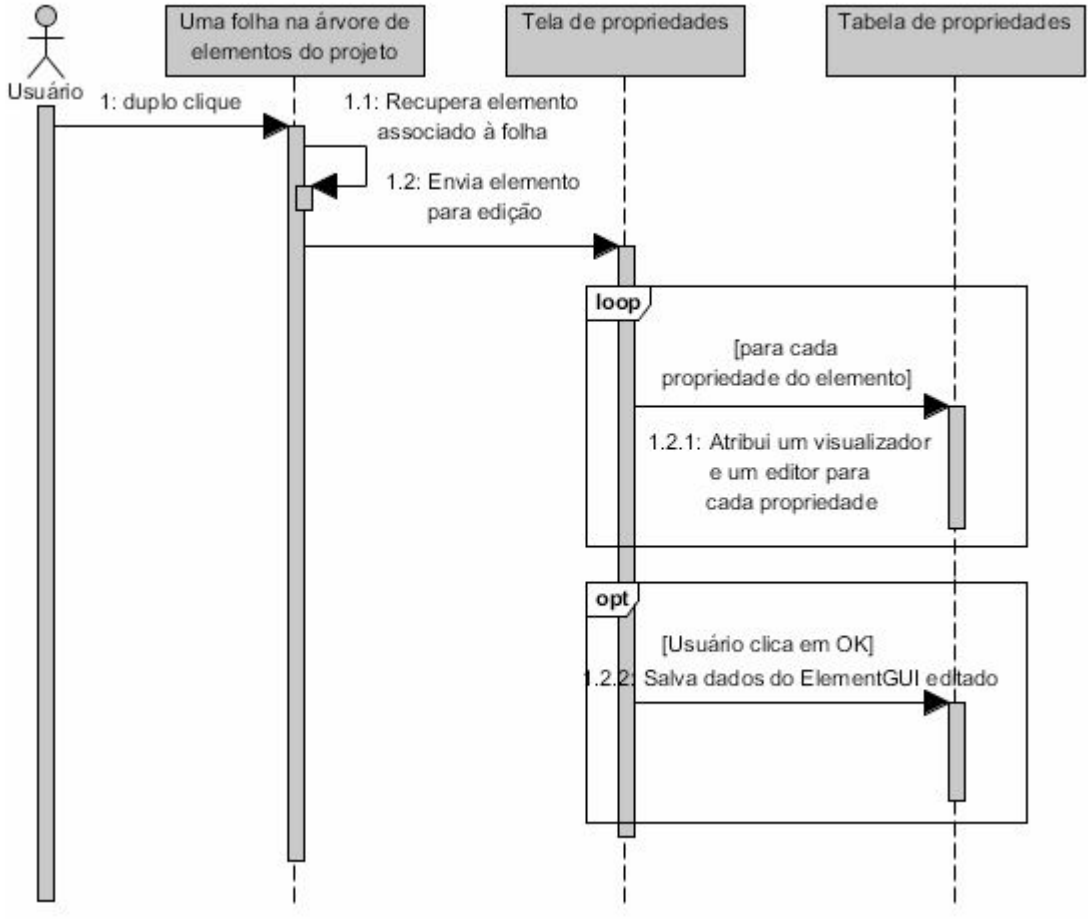
O fragmento de loop é usado para representar uma sequência repetitiva. Coloque as palavras **‘loop’** na caixa do nome e a condição de guarda perto do canto superior esquerdo do quadro.

Além do teste booleano, o guarda em um fragmento de laço pode ter duas outras condições especiais testadas contra.

iterações mínimas (escritas como **min int = [o número]**)

iterações máximas (escritas como **max int = [o número]**)

Se for uma proteção de iterações mínimas, o loop deve executar não menos que o número mencionado, e se for uma proteção de iterações máximas, o loop não deve executar mais do que o número indicado.


Usuário
Uma folha na árvore de elementos do projeto
Tela de propriedades
Tabela de propriedades
1: duplo clique
1.1: Recupera elemento associado à folha
1.2: Envia elemento para edição
loop
[para cada propriedade do elemento]
1.2.1: Atribui um visualizador e um editor para cada propriedade
opt
[Usuário clica em OK]
1.2.2: Salva dados do ElementGUI editado

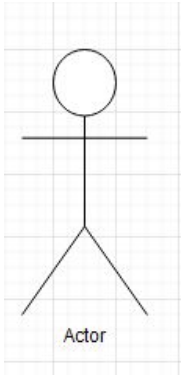
OUTROS ELEMENTOS

ATORES

O que são: Representam usuários ou sistemas externos que interagem com o sistema.

Exemplo: Um cliente usando um aplicativo ou um sistema de pagamento externo.

Função no diagrama: Iniciam ou recebem mensagens de objetos do sistema, mas não fazem parte dele.

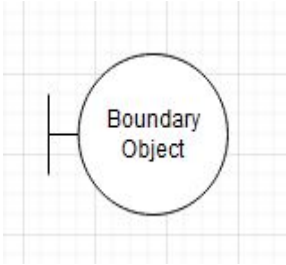


Classe Fronteira (Boundary)

O que é: Representa a interface entre o sistema e o ator, ou seja, como o ator interage com o sistema.

Exemplo: Telas, formulários ou APIs.

Função no diagrama: Recebe as solicitações dos atores e repassa essas solicitações para o sistema.

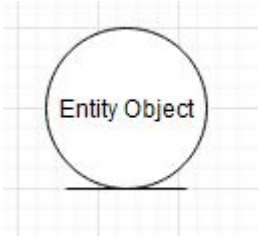


Classe de Entidade (Entity)

O que é: Representa objetos que armazenam informações ou dados persistentes, como registros em um banco de dados.

Exemplo: Um objeto "Cliente" que armazena informações de um cliente.

Função no diagrama: Manipula e armazena dados, geralmente em resposta a solicitações de outras classes.

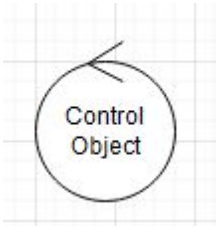


Classe de Controle (Control)

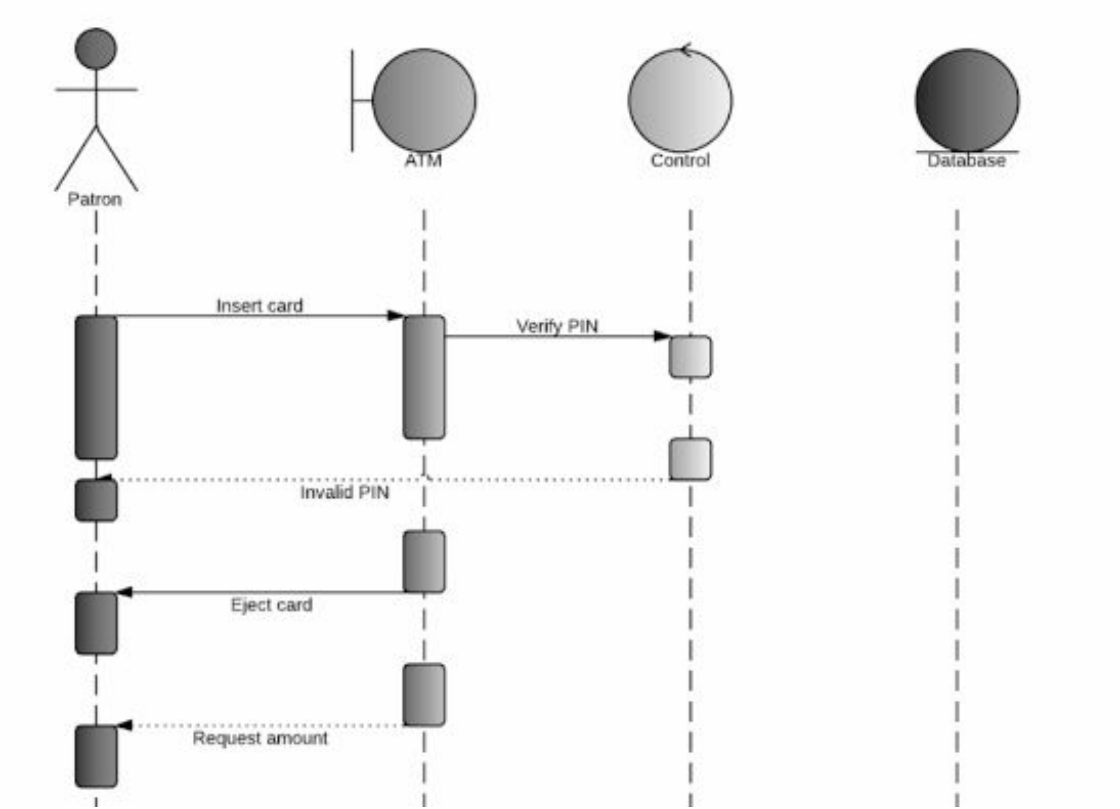
O que é: Coordena o fluxo entre as classes fronteira e de entidade, controlando as regras de negócio e o comportamento do sistema.

Exemplo: Um objeto "Controlador de Pedido" que gerencia o processo de compra.

Função no diagrama: Recebe as mensagens da classe de fronteira, realiza as regras de negócio e interage com as classes de entidade.



EXEMPLOS


Patron
ATM
Control
Database
Insert card
Verify PIN
Invalid PIN
Eject card
Request amount