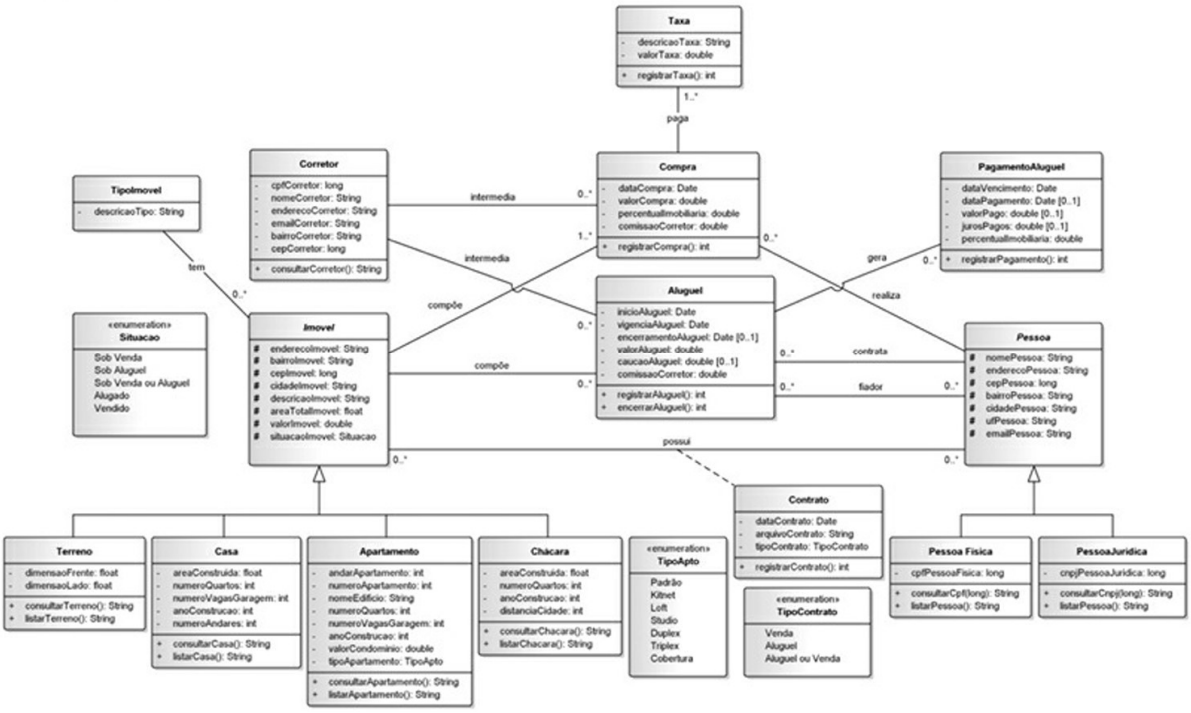


Modelagem de interação.

7-1: Descreva a posição dos diagramas de interação no processo de desenvolvimento incremental e iterativo. Quando eles são utilizados? Para que são utilizados?

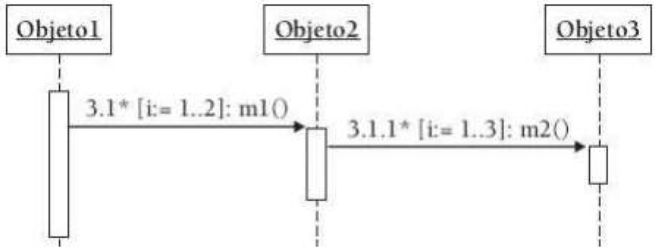
Os diagramas de interação são usados para mostrar a estrutura dinâmica dos objetos, mostrando como os objetos trocam mensagens uns com os outros. Os diagrama de iteração usam o modelo de classe e o modelo de caso de uso para serem construídos, e podem após sua construção revelar a necessidade de refinar os modelos de classe e modelo de caso de uso.

7-2: Ivar Jacobson, um dos proponentes da UML, disse uma vez: "Somente após a construção de diagramas de interação para os cenários de um caso de uso pode-se ter certeza de que todas as responsabilidades que os objetos devem cumprir foram identificadas." Reflita sobre essa afirmativa. A construção dos diagramas de interação é realmente essencial para a definição das responsabilidades dos objetos participantes de um caso de uso? Justifique.

Isso se mostra uma realizada, pois após a modelagem de interação é possível verificar se as classes realmente possuem as operações que precisam. Caso uma operação não seja

usada na modelagem de interação ela deve ser tirada da classe a qual pertence, ou adicionar uma operação ainda não adicionada a determinada classe.

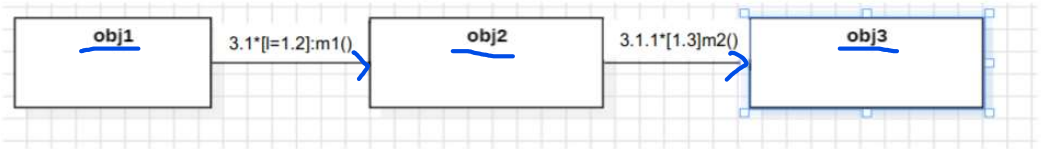
7-3: Considere o fragmento de diagrama de sequência a seguir. Determine a ordem na qual as mensagens m1 e m2 serão passadas.



As mensagens são enviadas na seguinte ordem, de cima para baixo:

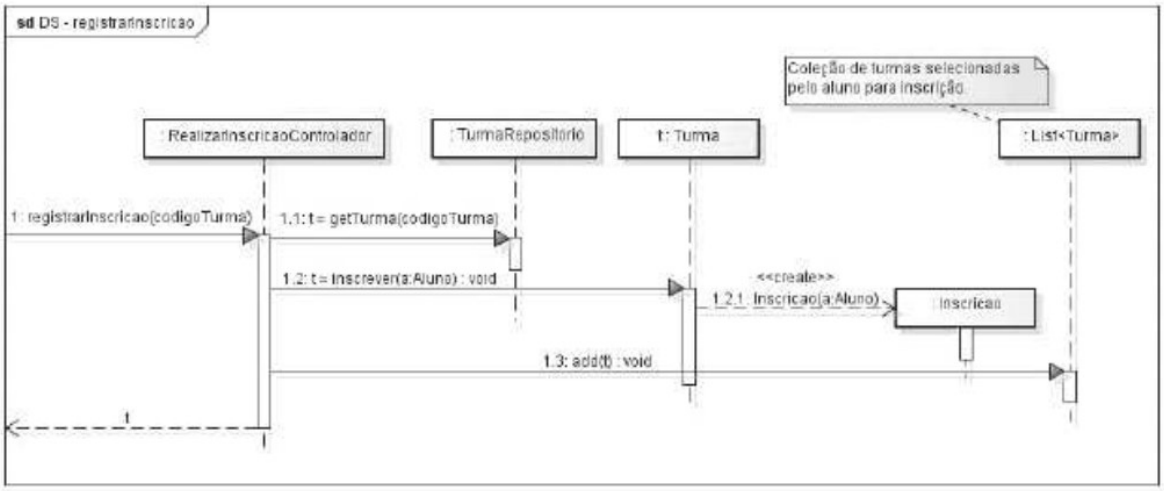
m1()
m2()
m2()
m2()
m1()
m2()
m2()
m2()

7-4: Desenhe um diagrama de comunicação equivalente ao diagrama de sequência do exercício anterior.



7-5: Construa os diagramas de comunicação equivalentes aos diagramas de sequência apresentados na Figura 7-24 e na Figura 7-25.

Figura 7-25: (livro bezerra)


sd DS - registrarInscricao
Coleção de turmas selecionadas pelo aluno para inscrição
: RealizarInscricaoControlador
: TurmaRepositorio
t: Turma
: List<Turma>
1: registrarInscricao(codigoTurma)
1.1: t = getTurma(codigoTurma)
1.2: t = inscrever(a:Aluno) : void
<<create>>
1.2.1: Inscricao(a:Aluno)
: Inscricao
1.3: add(t) : void
t

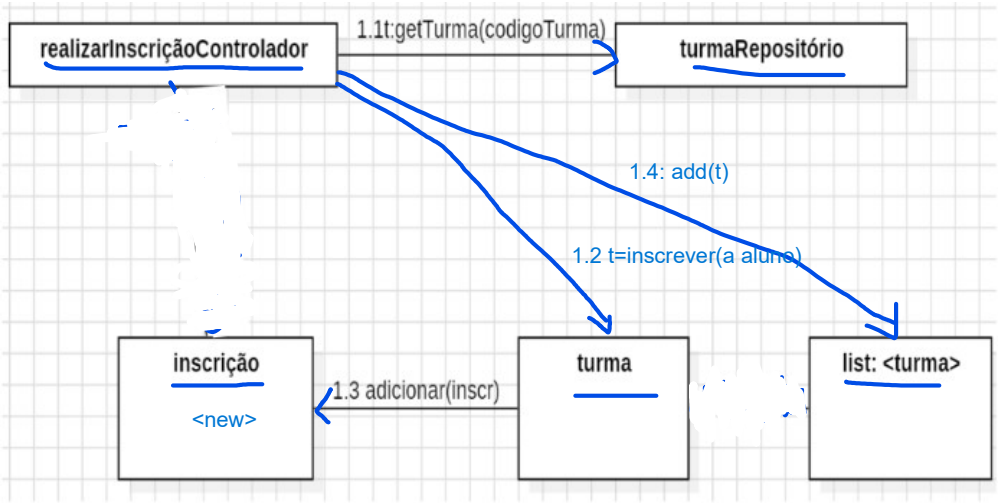
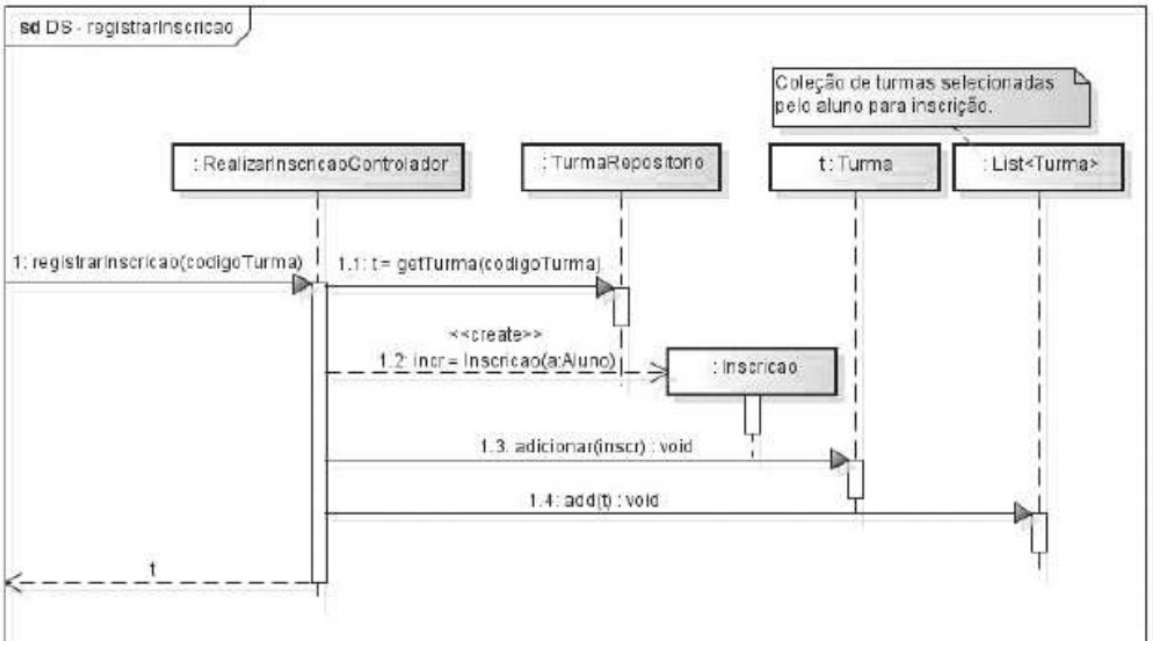
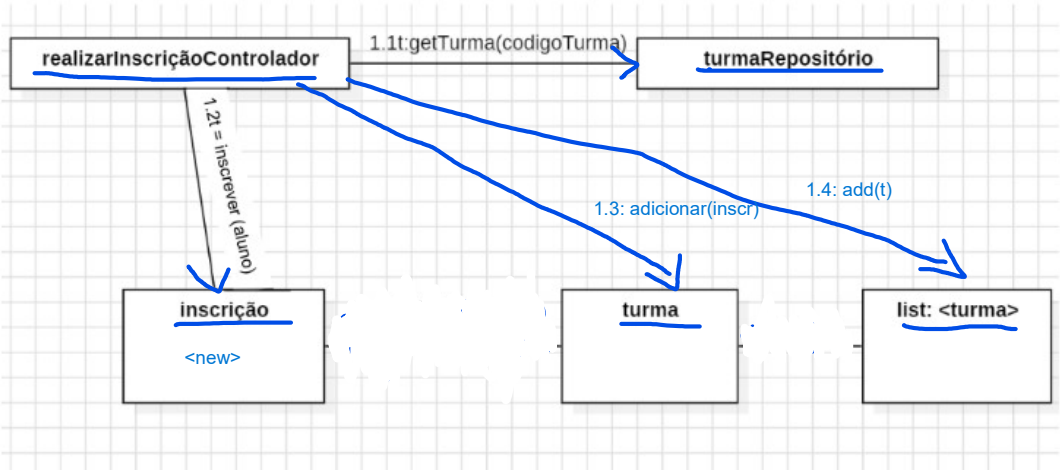
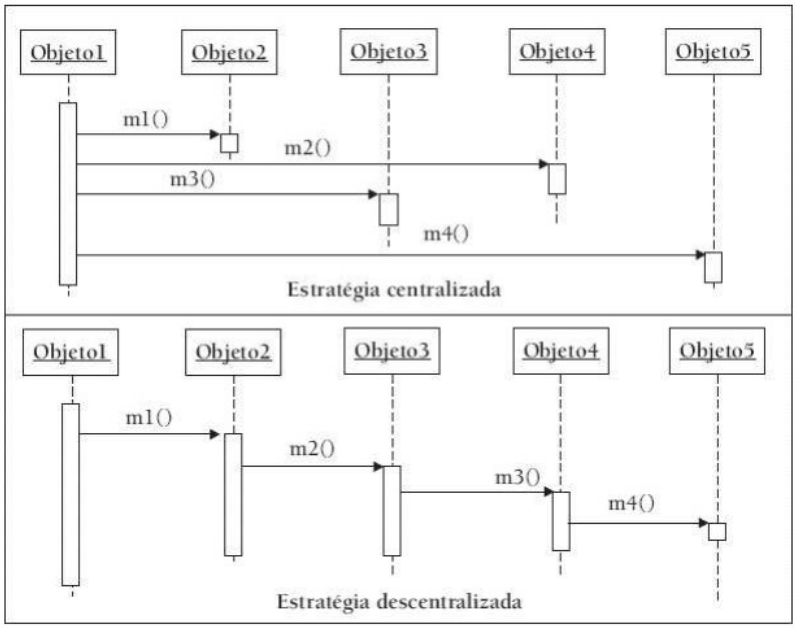

realizarInscriçãoControlador
1.1t:getTurma(codigoTurma)
turmaRepositório
1.4: add(t)
1.2 t=inscrever(a aluno)
inscrição
<new>
1.3 adicionar(inscr)
turma
list: <turma>

Figura 7-24 (livro bezerra)





7-9: De acordo com a divisão de responsabilidades pelos objetos de um sistema, a colaboração entre eles para a realização de um cenário pode ser classificada em um espectro que vai desde a forma *centralizada* até a forma *descentralizada*.[2] Na primeira forma de colaboração (centralizada), a inteligência do sistema está concentrada em um único objeto. Por outro lado, na forma descentralizada, a inteligência do sistema está mais uniformemente espalhada pelas classes. A figura a seguir apresenta de maneira esquemática as duas estratégias de colaboração, utilizando a notação vista para diagramas de sequência. Note que, na estratégia centralizada, há um objeto que controla os demais (Obj1). Já na estratégia descentralizada, há uma cadeia de *delegações* entre os objetos; não há um objeto central que "conhece" todos os demais. Cada objeto realiza uma parte da tarefa. Discuta as vantagens e desvantagens de cada uma dessas estratégias.



Centralizada: Vantagem, centraliza a orquestração e “inteligência” do sistema em classes de controle, deixando as classes de entidade fazerem o que sabem para a sua entidade em si. Isso centraliza as manutenções das regras de negócio na classe de controle. A desvantagem é que isso deixa a classe de controle com uma alta coesão das regras de negocio, sobrecarregando o mesmo. O objeto de controle acaba trocando mensagem com todos os objetos da entidade, deixando o sistema com alto acoplamento.

Descentralizada: Vantagem, é deixar o sistema com uma alta coesão em todos os objetos, sendo distribuída as regras de negócio nos próprios objetos da entidade. Não espera-se ter um objeto de controle central nesta arquitetura. O sistema fica mais desacoplado. Desvantagem: A manutenção de uma arquitetura assim fica comprometida, pois as regras de negócio não estão centralizadas. Adicionar novas camadas nesse tipo de sistema fica mais complexa e difícil.

Opção: ter uma arquitetura com mais de uma classe de controle, deixando as regras de negócio melhor distribuídas e a orquestração não ser centralizada em apenas uma classe de controle.

8-1: No Capítulo 7, descrevemos os conceitos de coesão e acoplamento. Qual é a relação desses conceitos com a qualidade resultante da modelagem de classes de projeto?

8.1: O grau de coesão do modelo mostra o quanto as classes estão focadas em suas responsabilidades, sem realizar as responsabilidade de outros objetos. O grau de acoplamento demonstra o quanto as classes estão conectadas e dependem umas das outras. Um modelo de qualidade resulta em uma modelagem de sistema com classes com alta coesão e baixo acoplamento. Ou seja, as classes focam suas responsabilidades e pedem colaboração de outras apenas quando necessário.

8-2: Este capítulo apresenta o relacionamento de dependência e descreve diversos tipos de dependência que podem existir (por atributo, por variável local etc.). Seja uma classe A que possui pelo menos uma operação cujo tipo de retorno é a classe B. Que tipo(s) de dependência pode haver entre A e B?

8.2:

- **Dependência por variável global - Uma variável externa aos métodos de uma classe é criada, em seguida, recebe um objeto já instanciado, que posteriormente poderá ser utilizado por todos os métodos da classe.**

- **Dependência por variável de método. Uma variável interna de um método de uma classe é criada, em seguida, recebe um objeto já instanciado, que posteriormente poderá ser utilizado apenas por aquele método da classe.**

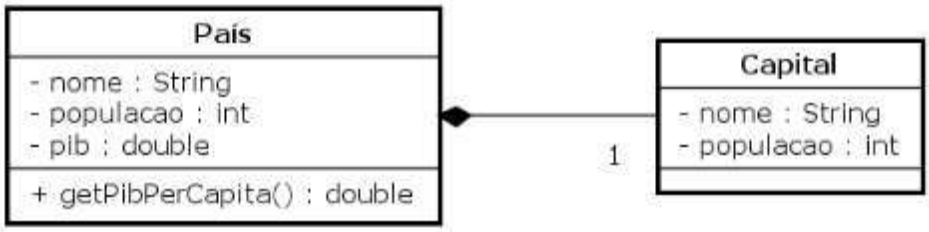
8.3:

8-3: Em cada um dos itens a seguir, discuta qual tipo de relacionamento é mais adequado (associação, agregação, composição). Desenhe o diagrama de classes correspondente, indicando as multiplicidades. Especifique, ainda, atributos e possíveis restrições.

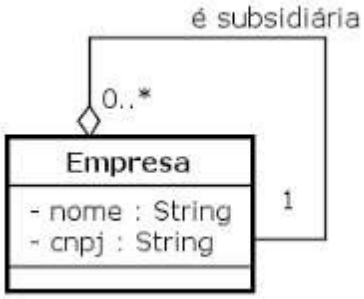
a) Um País possui uma Capital.
b) Uma Empresa é subsidiária de diversas outras Empresas.
c) Uma Pessoa à mesa de jantar está usando uma Faca.
d) Um Polígono é composto por um conjunto ordenado de Segmentos.
e) Um Estudante acompanha uma Disciplina com um Professor.
f) Uma Caixa contém Garrafas.
g) Um Livro contém Capítulos.
h) Um Arquivo contém Registros.

a-)

A)



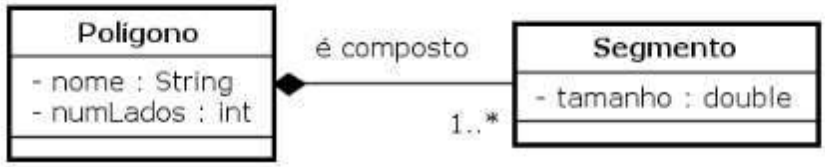
B)



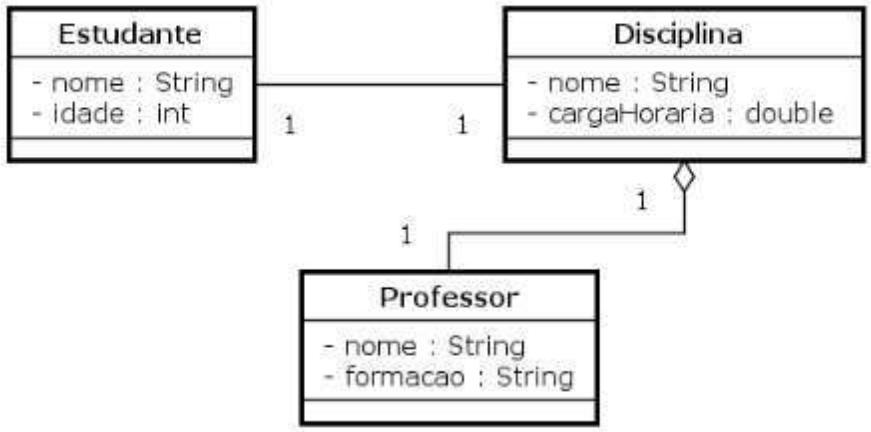
C)


Pessoa
- nome : String
- idade : int
- sexo : char
usa
1
1
Faca
- modelo : String
- marca : String
- cor : String

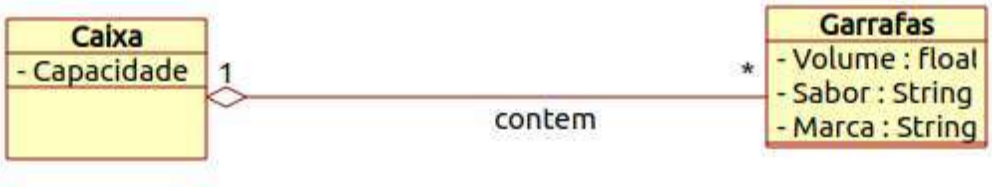
D)


Poligono
- nome : String
- numLados : int
é composto
1..*
Segmento
- tamanho : double

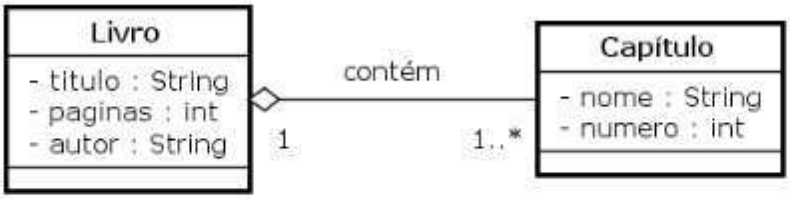
E)


Estudante
- nome : String
- idade : int
1
1
Disciplina
- nome : String
- cargaHoraria : double
1
1
Professor
- nome : String
- formacao : String

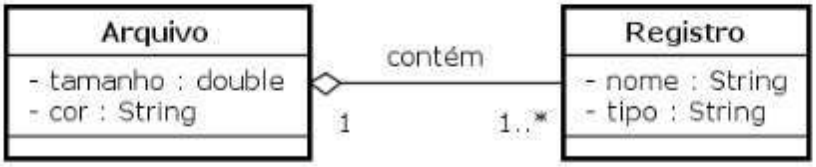
F)



G)

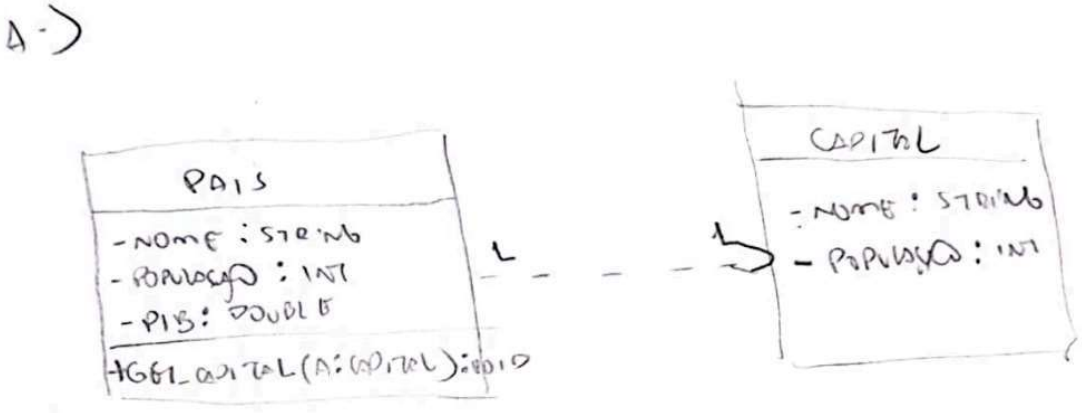


H)



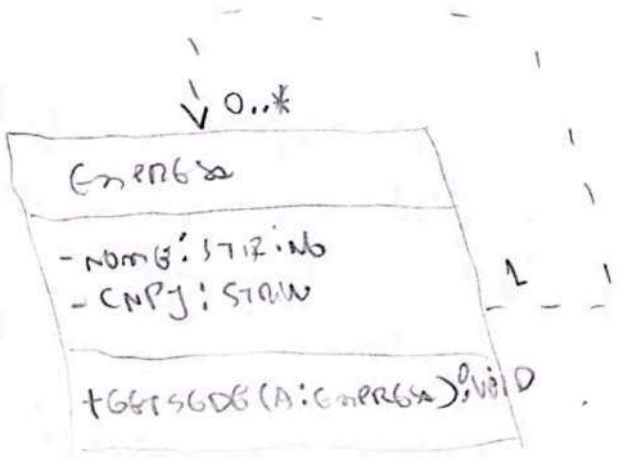
8-4: refinamento da classe analise para classe de projeto.

A)



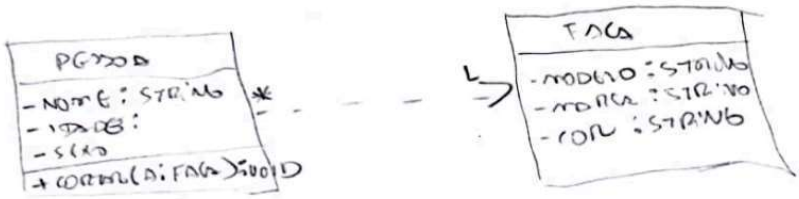
B)

B.)


0..*
- NOME : STRING
- CNPJ : STRING

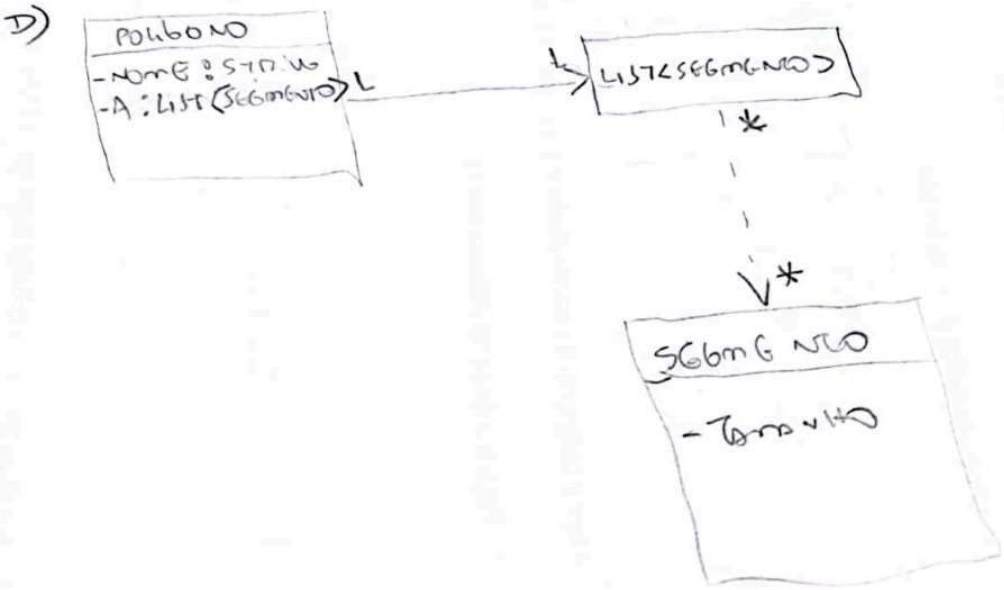
C)

c)


PESSOA
- NOME : STRING
FAIXA
- NOME : STRING

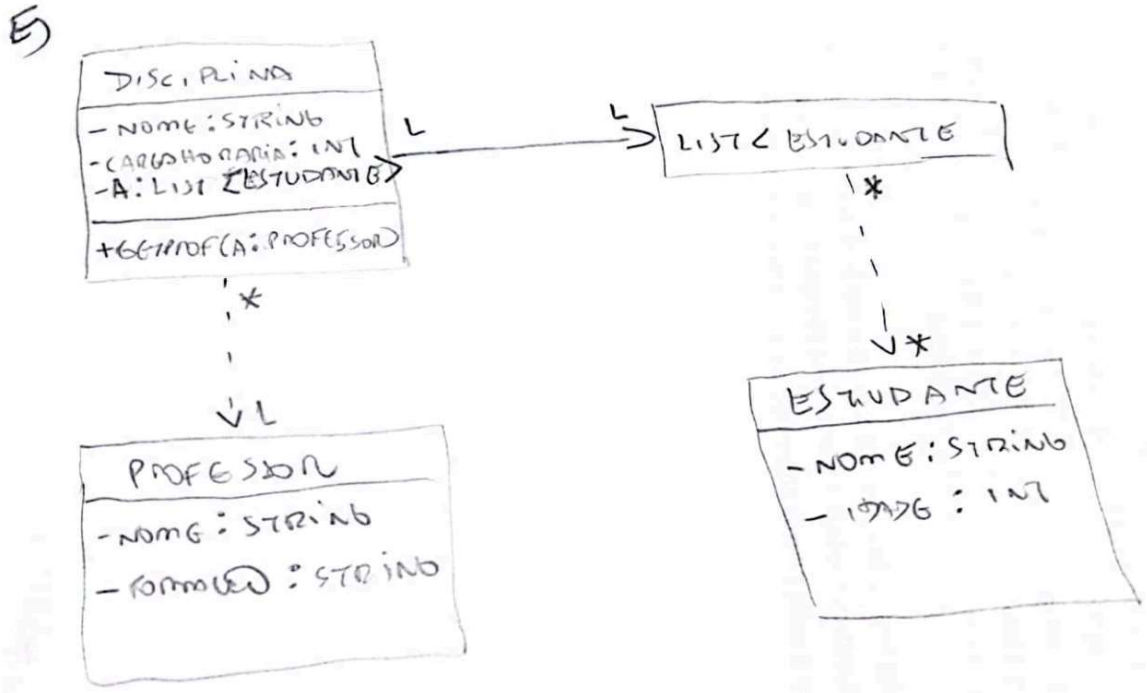
D)

D)


POLIGONO
- NOME : STRING
- A : LIST (SEGMENTO)
LISTA SEGMENTO
SEGMENTO
- TERMINO

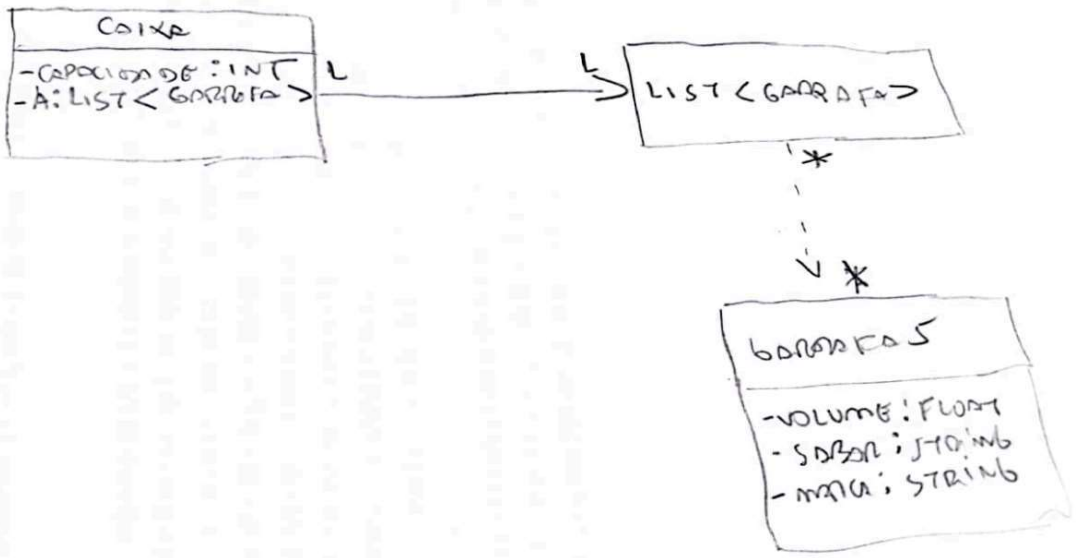
E)

E)

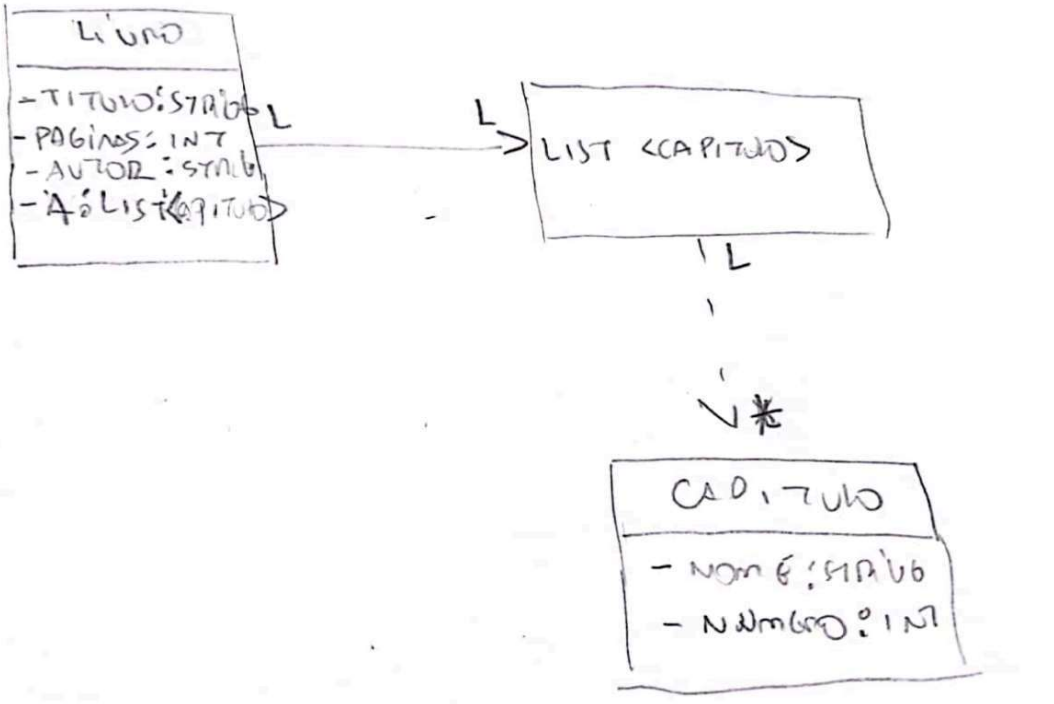

DISCIPLINA
-NOME:STRING
-CARGAHORARIA:INT
-A:LIST<ESTUDANTE>
+GETPROF(A:PROFESSOR)
L
L
LIST<ESTUDANTE>
1*
*
PROFESSOR
-NOME:STRING
-FORMACAO:STRING
L
*
ESTUDANTE
-NOME:STRING
-IDADE:INT

F)

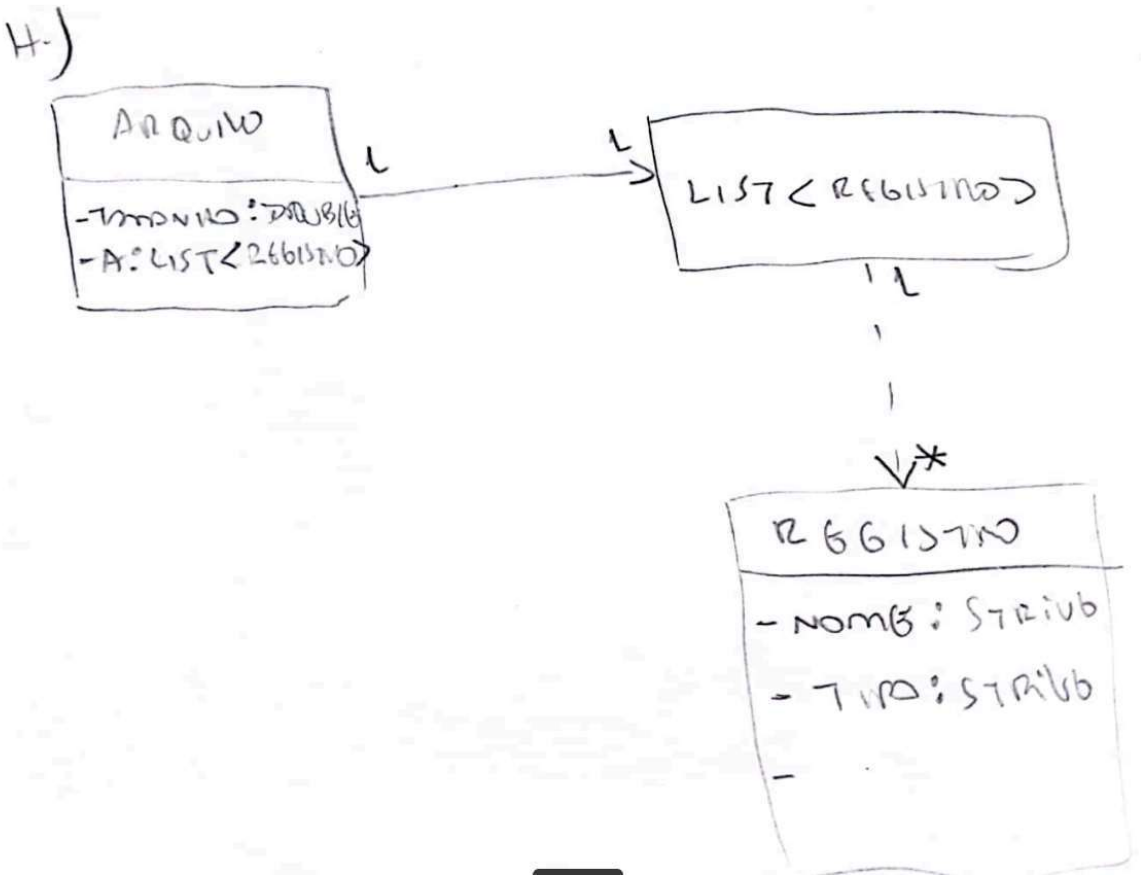
F)


Caixa
-CAPACIDADE : INT
-A: LIST < GARRAFA >
L
L
LIST < GARRAFA >
*
*
GARRAFAS
-VOLUME : FLOAT
- SABOR : STRING
- MARCA : STRING

G)

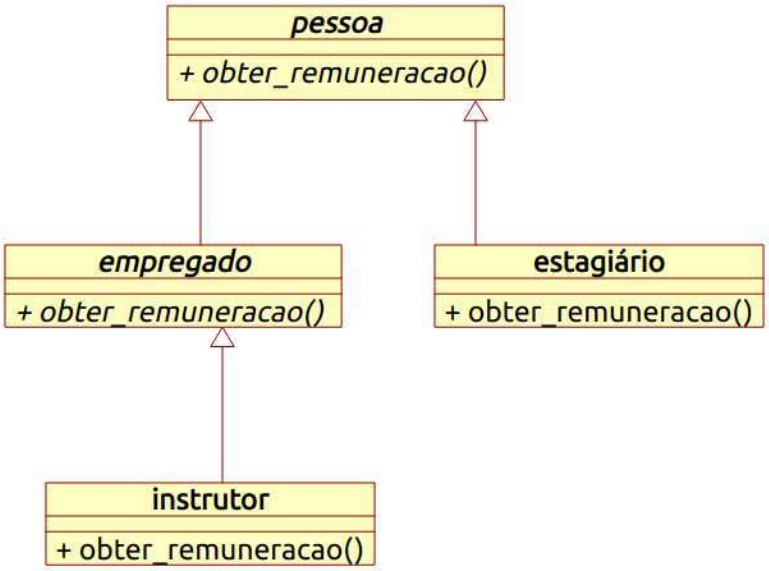

LIVRO
-TITULO: STRING
-PAGINAS: INT
-AUTOR : STRING
-A: LIST<CAPITULO>
L
L
LIST <CAPITULO>
L
*
CAPITULO
- NOME : STRING
- NUMERO : INT

H)



8-5: Forme uma hierarquia de classes a partir dos seguintes conceitos: pessoa, empregado, instrutor e estagiário.

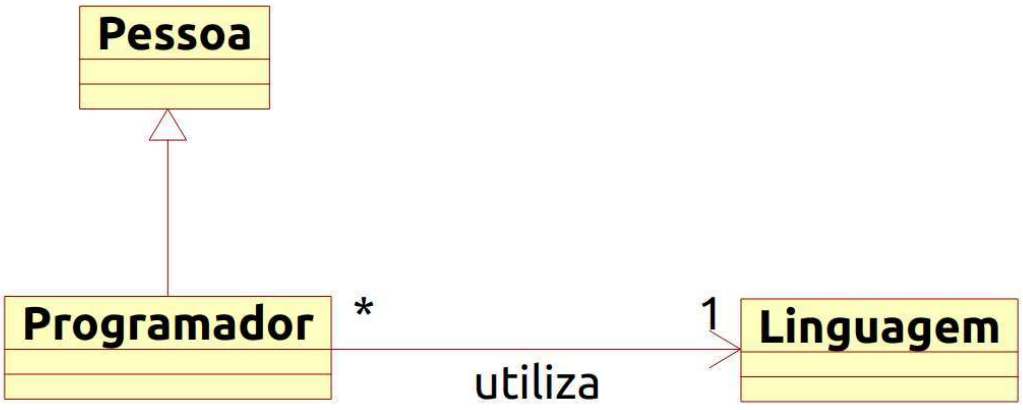
8-5:



8-6:

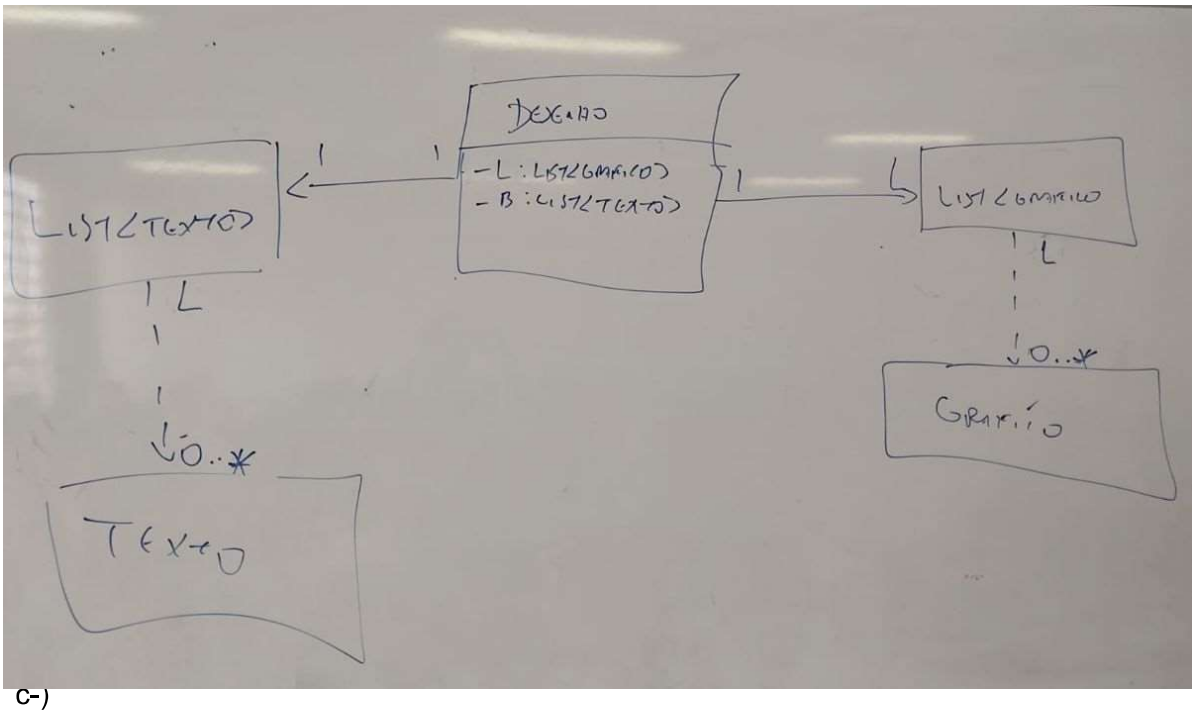
a-)

a) Uma pessoa, como programador, utiliza uma linguagem de programação.



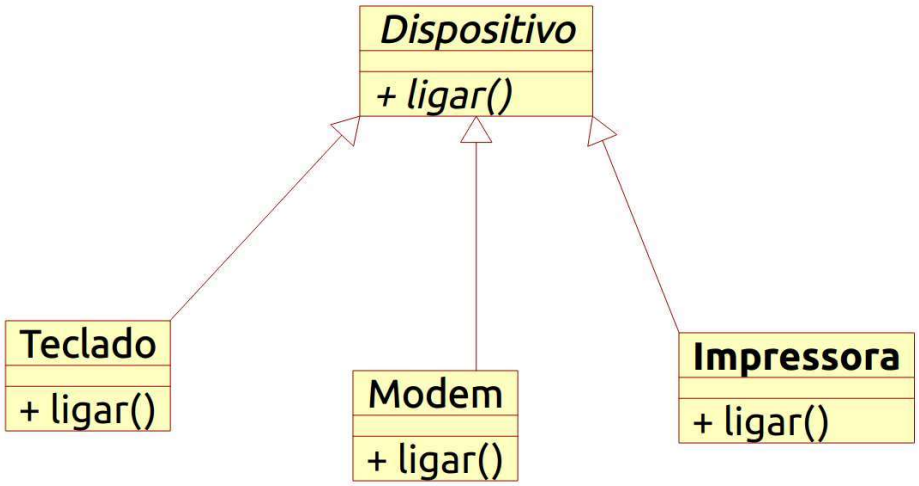
b-)

b) Um objeto de desenho pode ser um texto, um gráfico ou um grupo de objetos.



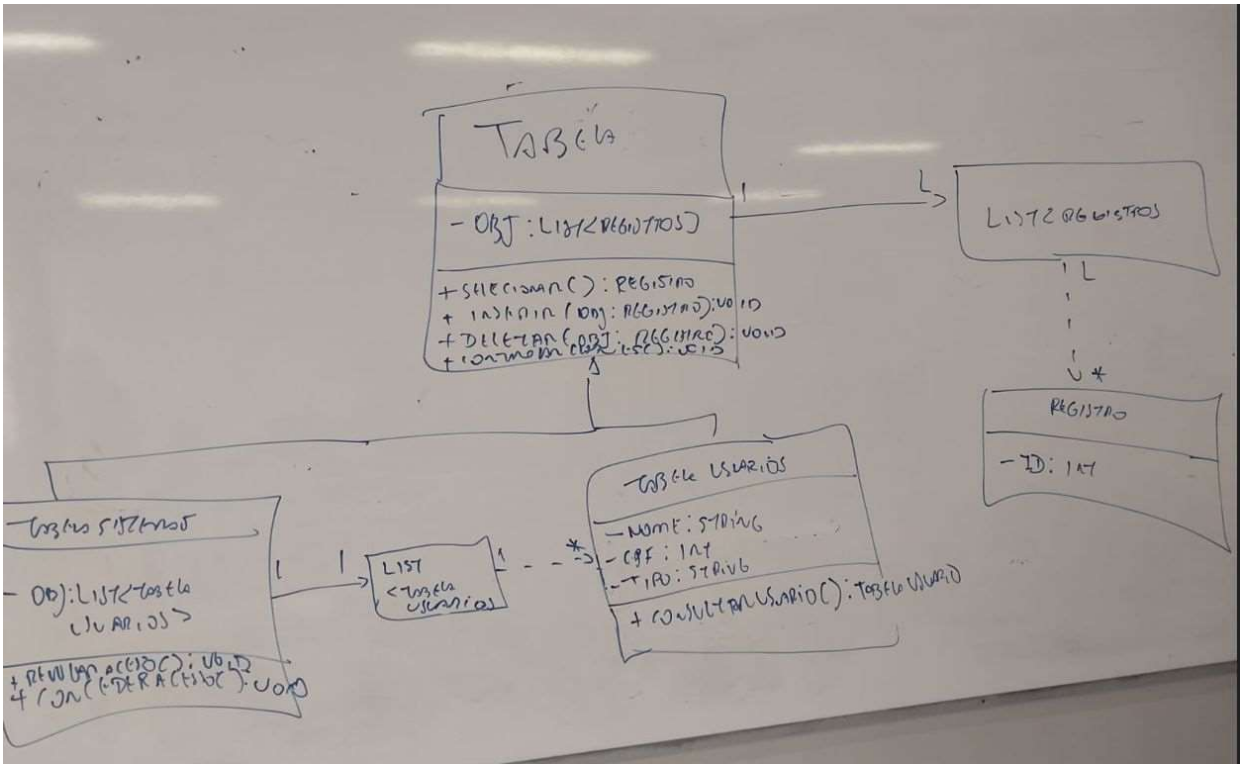
c-)

c) Modem, teclado e impressora são dispositivos de entrada e saída.



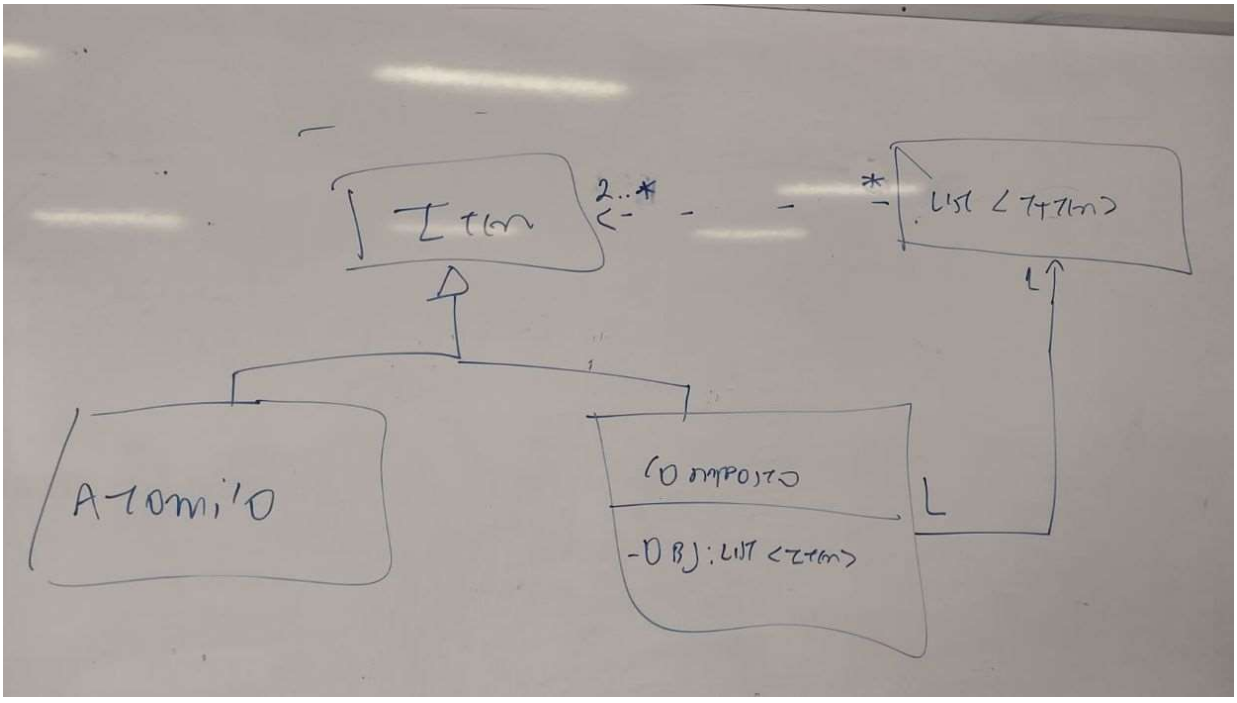
d-)

d) Um banco de dados contém tabelas de sistema e tabelas de usuário. Uma tabela de sistema mantém informações sobre uma ou várias tabelas de usuário. Uma tabela contém registros.



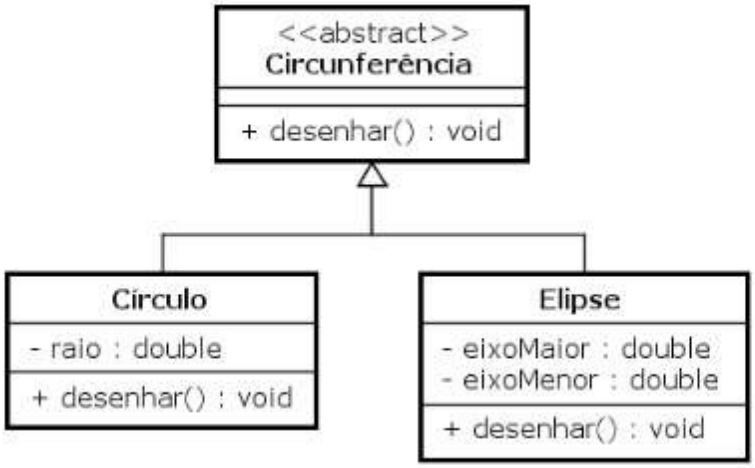
e-)

e) Um Item pode ser atômico ou composto. Um item composto possui dois ou mais Itens.

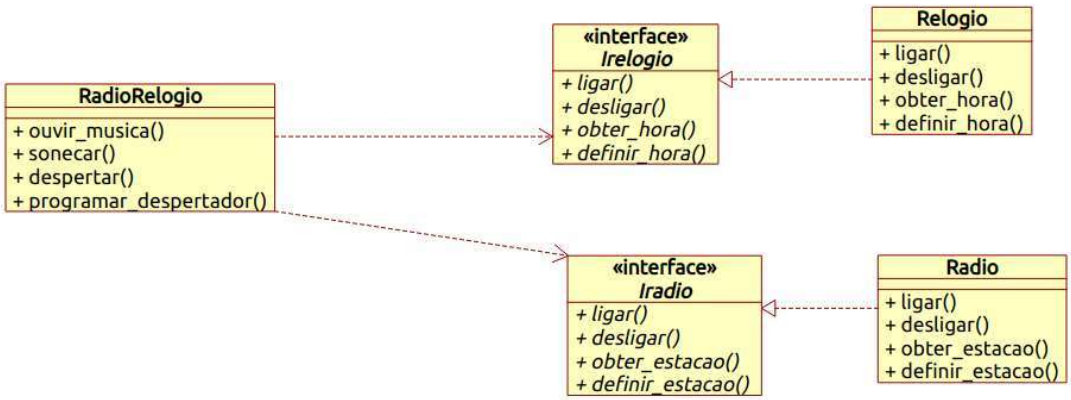


8-8: Crie um diagrama de classes de projeto para representar os conceitos de círculo e de elipse como classes. Defina uma operação desenhar em ambas as classes. Dica: crie uma superclasse abstrata para organizar a hierarquia.

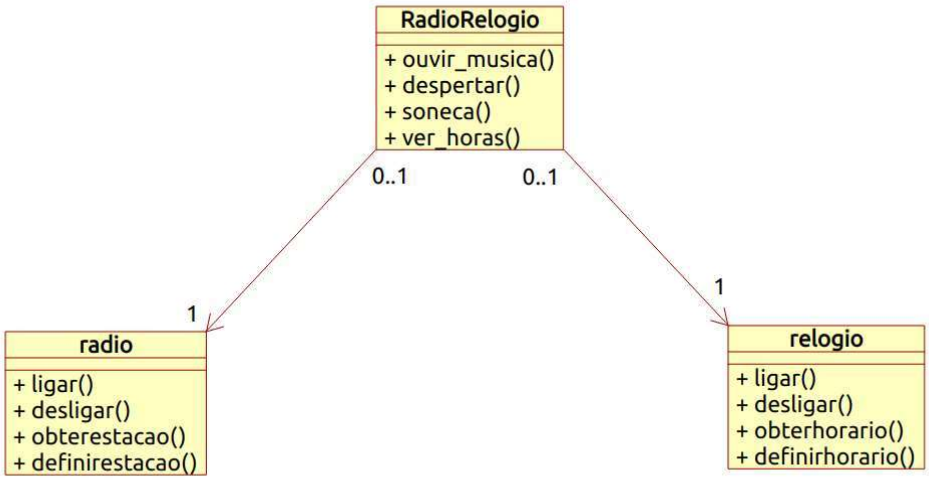
8-8:



8-9: **Por interface:**



Por delegação:



8.10: A situação da esquerda traz uma generalização comum, onde a classe pessoa é superclasse, e as classes cliente e funcionário herdam tudo deste, ou seja atributos, operações e relacionamentos. Porém, podem existir coisas que as filhas não precisem, o que faz ocorrer um “desperdício” na herança.

Neste caso, o fragmento a direita, seria mais interessante pois, as classes filhas passariam a delegar o que não sabem fazer sozinhas para a classe pessoa, deixando de existir uma herança, passando a fazer uma relação de delegação. A delegação, diferente da herança, faz com que as classes tenham uma dependência (acoplamento), o que pode tornar a alternativa uma desvantagem.

- No caso de clientes poderem ser funcionários, é indicado a criação da classe **ClienteFuncionário**, que permita que um mesmo objeto faça as duas atividades, uma metamorfose adaptada..

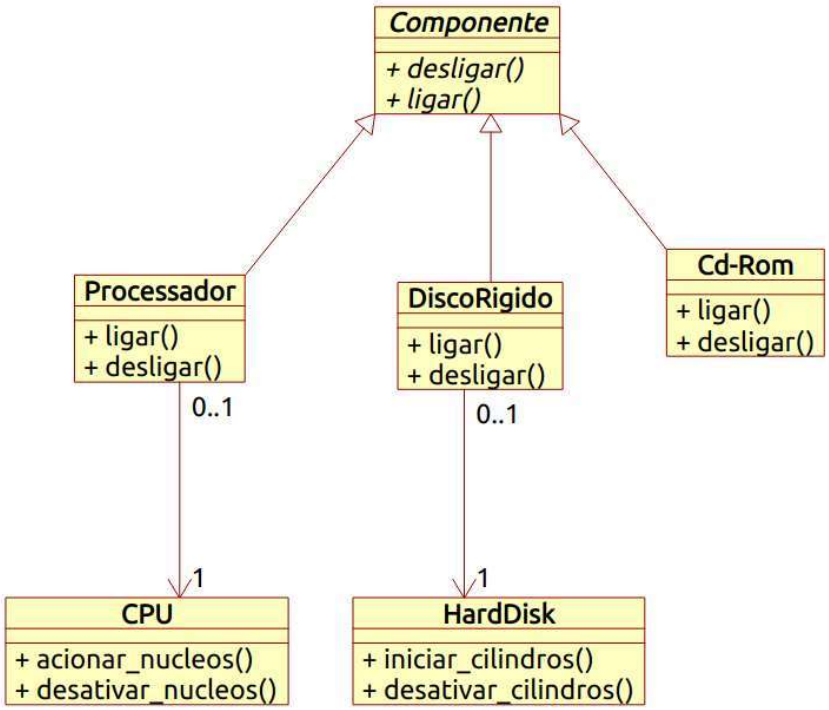
- No caso de uma classe gerente tivesse que ser modelada, poderia ser adicionada uma nova classe Gerente, que delegasse o que não sabe para a classe funcionário, fazendo com que a delegação se propagasse até a classe pessoa. Isso deixaria o modelo muito acoplado, sendo, nesse caso, recomendável o uso do fragmento do lado esquerdo, pois seria melhor representada a hierarquia da herança.

8-12: Sim, o refinamento foi adequado para atender ao princípio do polimorfismo e encapsulamento. Além disso, fez o sistema deixar as classes fornecedoras ficarem desacopladas das suas classes consumidoras. Qualquer classe nova pode assumir as

interfaces comestível ou vendável, pois respeitando os contratos, poderão ser incorporadas dentro do sistema sem a necessidade de alterar as outras classes, pois as interfaces foram mantidas. O fragmento b é uma ótima solução, conforme justificativas anteriores.

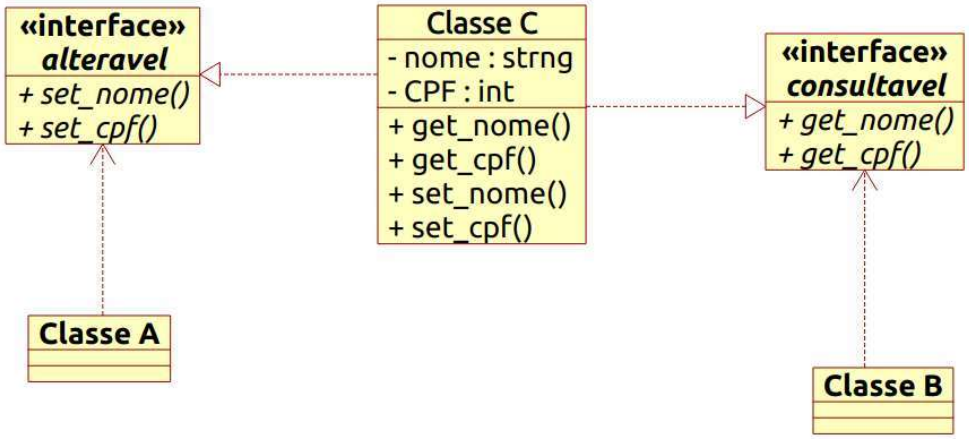
8-13:

8-13: Você está desenvolvendo uma aplicação para uma empresa que vende componentes de computador. Atualmente, você está construindo uma hierarquia de classes para representar os diferentes tipos de componentes (você nomeou essas classes como *Processador, DiscoRígido* e *CD-ROM*). Essa hierarquia contém também uma superclasse abstrata denominada *Componente*, da qual são derivadas as demais classes. Por outro lado, um amigo seu já desenvolveu um sistema semelhante e lhe passou classes para representar discos rígidos e CPUs (as classes dele são denominadas *HardDisk* e *CPU* respectivamente). De que maneira você pode construir sua hierarquia de classes aproveitando (reutilizando) as classes fornecidas pelo seu amigo sem precisar modificá-las? Que padrão de software poderia ser empregado para auxiliar na definição da estrutura de classes a fim de representar a situação discutida anteriormente? Utilizando esse padrão, forneça um fragmento de diagrama de classes que represente essa situação.

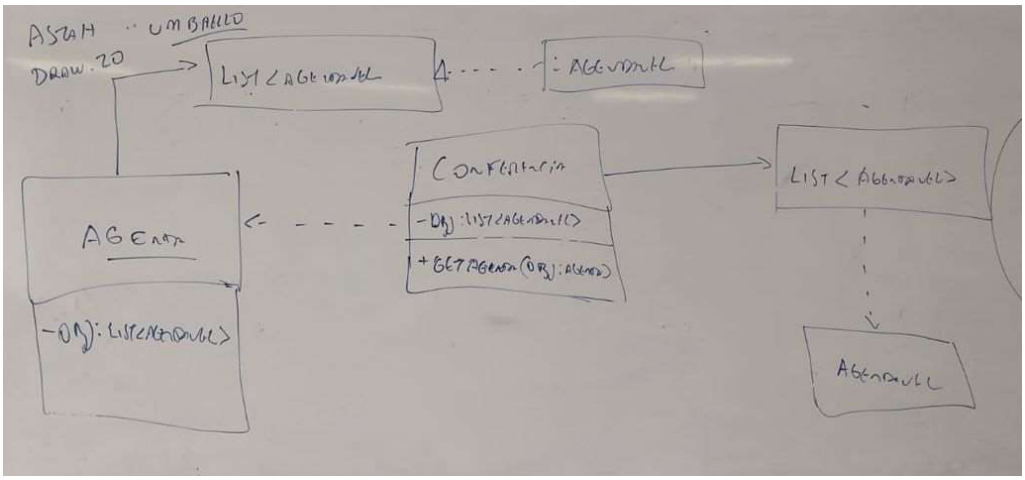


8-15: Suponha que você precise ter uma classe C em sua aplicação para a qual a forma de acesso é diferenciada, no seguinte sentido: há um conjunto de classes que têm permissão para alterar o estado dos objetos da classe C; há também outro conjunto de classes que, embora façam uso dos serviços fornecidos por objetos da classe C, não têm permissão para modificar o estado desses objetos. Defina uma solução de projeto para esse problema. Dica: utilize o conceito de interface.

8-15:



8-16:



8-17: Analise as relações existentes entre o conceito de interface e os princípios de polimorfismo e de encapsulamento da orientação a objetos.

8-17: A interface proporciona um desacoplamento robusto para o modelo de um sistema orientado a objetos, uma vez que uma classe, ao requisitar o serviço de outra classe, não sabe nem de que tipo é a mesma, uma vez que a classe cliente sabe apenas a que tipo de contrato a classe assumiu. Ou seja, ao chamar o método vender, o mesmo não sabe como será a venda, uma vez que a implementação foi feita pelo objeto que está implementando a interface vendável, que pode ser uma bicicleta, um relógio, uma piscina, um caderno, um

software ou qualquer outra coisas que seja vendável. Isso faz o poliformismo acontecer da maneira mais primitiva possível, onde não se sabe nem qual o objeto que está fazendo a ação, mas apenas o que ele sabe fazer, através de uma interface conhecida.

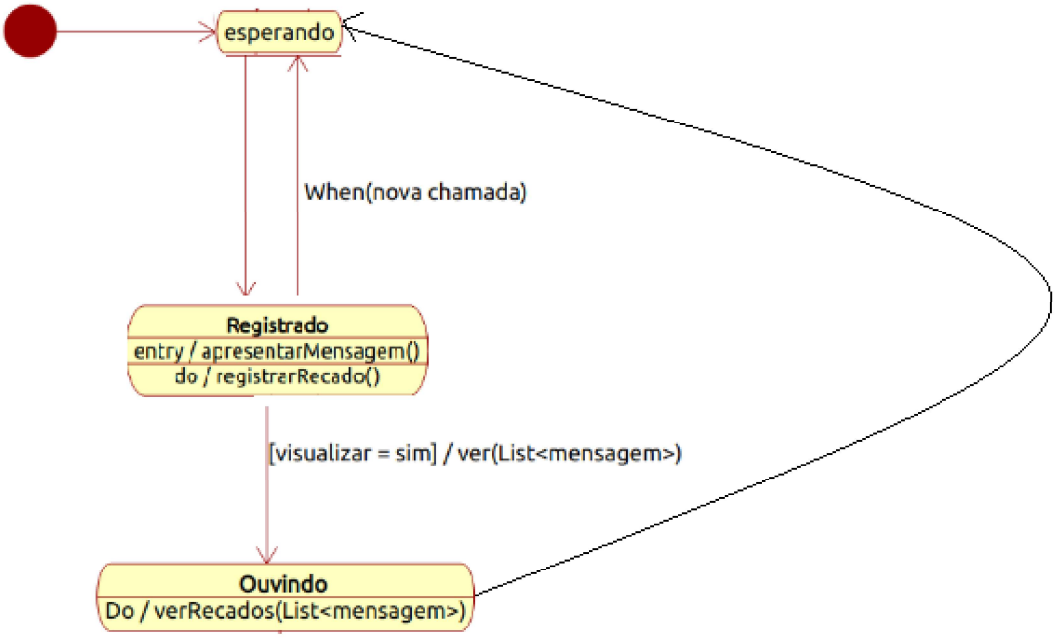
8-18: Defina as diferenças e semelhanças existentes entre uma classe abstrata e uma interface. Qual é a diferença entre uma classe concreta que herda de uma classe abstrata e a realização de uma interface.

8-18:

Classe abstrata e interfaces não podem ser instanciadas e possuem métodos não implementados. Uma classe abstrata existe para organizar uma hierarquia de generalização, enquanto uma interface existe para formalizar contrato entre classes, de forma a oferecer o desacoplamento das classes fornecedoras das classes consumidoras de seus serviços.

Uma classe concreta que herda de uma classe abstrata precisa implementar seus métodos abstratos e herdar seus métodos concretos.

Uma classe que realiza uma interface, precisa implementar todos os seus métodos, pois assume o contrato com a interface.



Modelagem de atividades

10-1: Descreva a posição do diagrama de atividade no processo de desenvolvimento incremental e iterativo. Quando eles são utilizados e para quê?

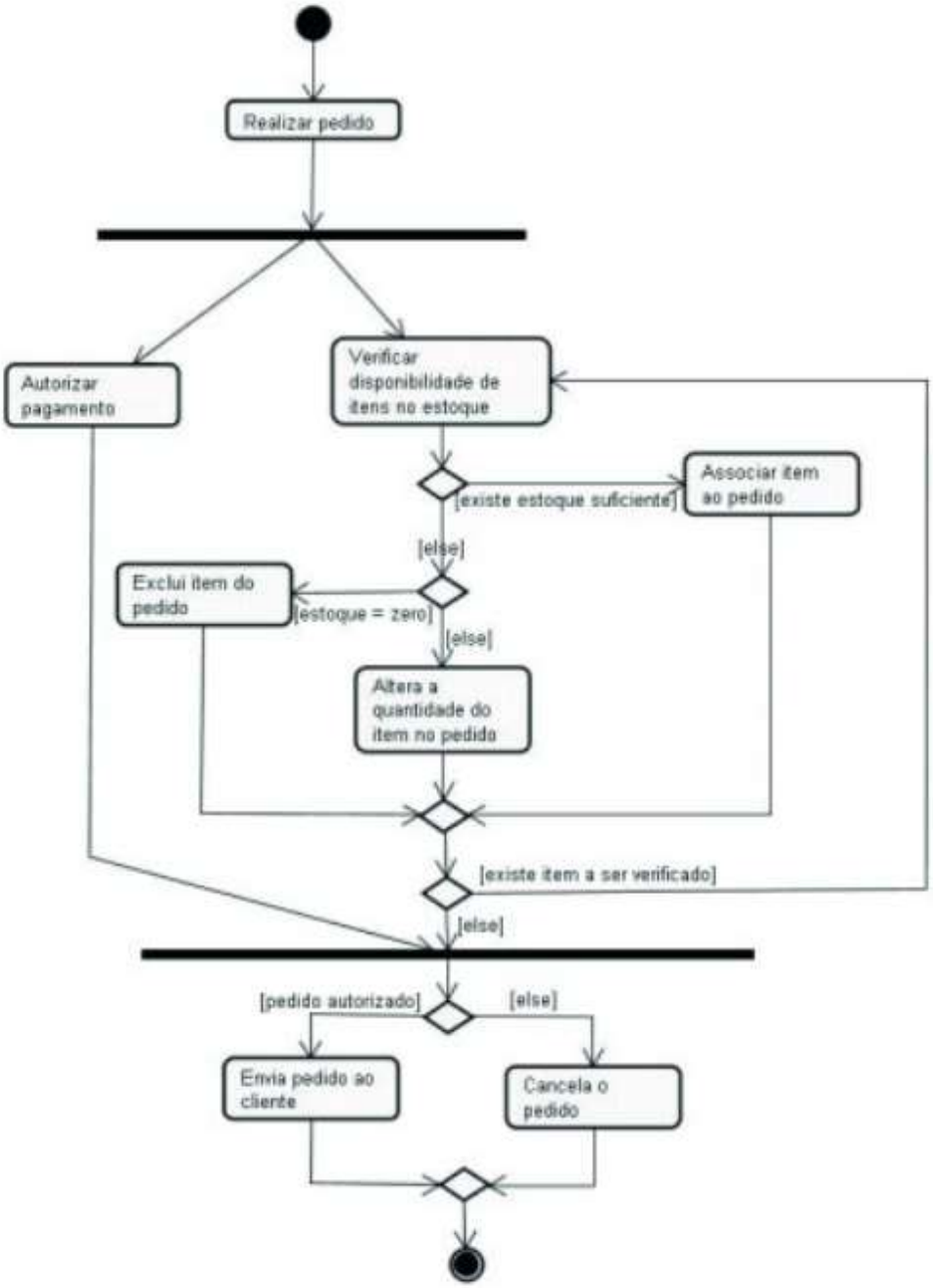
10-2: Construa um diagrama de atividades para o seguinte processo de negócio: a autorização do pagamento tem início após um pedido ter sido feito pelo cliente. Ao mesmo tempo, a disponibilidade para cada um dos itens do pedido é verificada pelo depósito. Se a quantidade requisitada de um determinado item existe em estoque, tal quantidade é associada ao pedido. Caso contrário, somente a quantidade disponível no momento é associada ao pedido. O pedido é enviado pelo depósito ao cliente quando todos os itens estiverem associados e o pagamento estiver autorizado. O pedido será cancelado se a ordem de pagamento não tiver sido autorizada.

Resposta 10-1: A posição do diagrama de atividades neste processo é auxiliar no entendimento de um fluxo para um caso de uso que se mostra complexo. Neste caso, o diagrama de atividades pode ser anexado à descrição do caso de uso, de maneira a elucidar o seu fluxo principal, de exceção, e de controle. Podemos também considerar que um diagrama de atividades pode mostrar um conjunto de casos de uso, como casos de uso

que são estendidos ou incluídos ao principal, mostrando dessa forma um fluxo que envolve mais de um caso de uso, e que se mostra complexo, auxiliando assim, a equipe envolvida na construção do software.

Resposta 10-2:

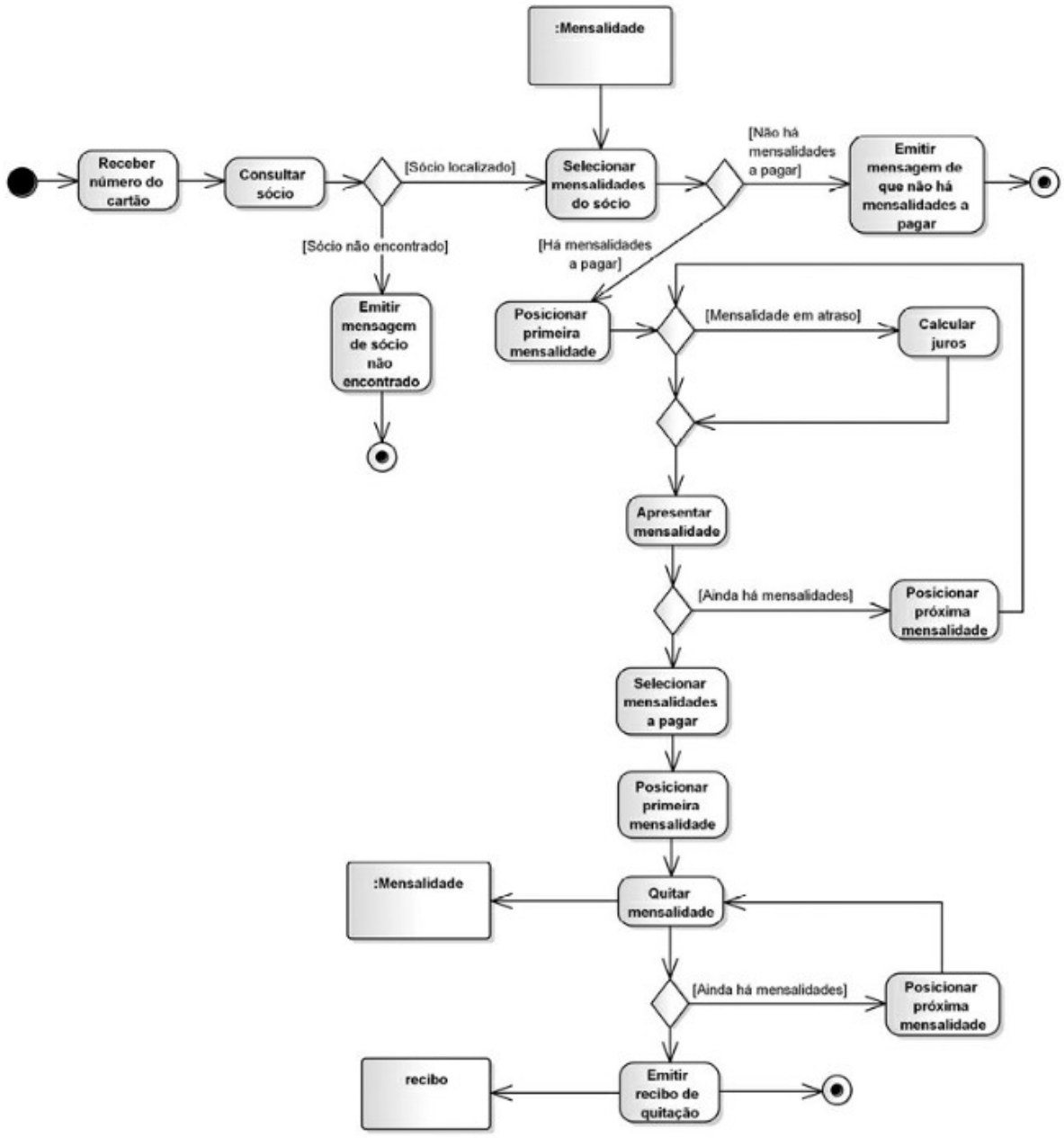
R.:



10.30.2 Sistema de Controle de Clube Social – Processo de Pagamento de Mensalidade

Desenvolva o diagrama de atividade referente ao processo de pagamento de mensalidade para um sistema de clube social, levando em consideração os seguintes fatos:

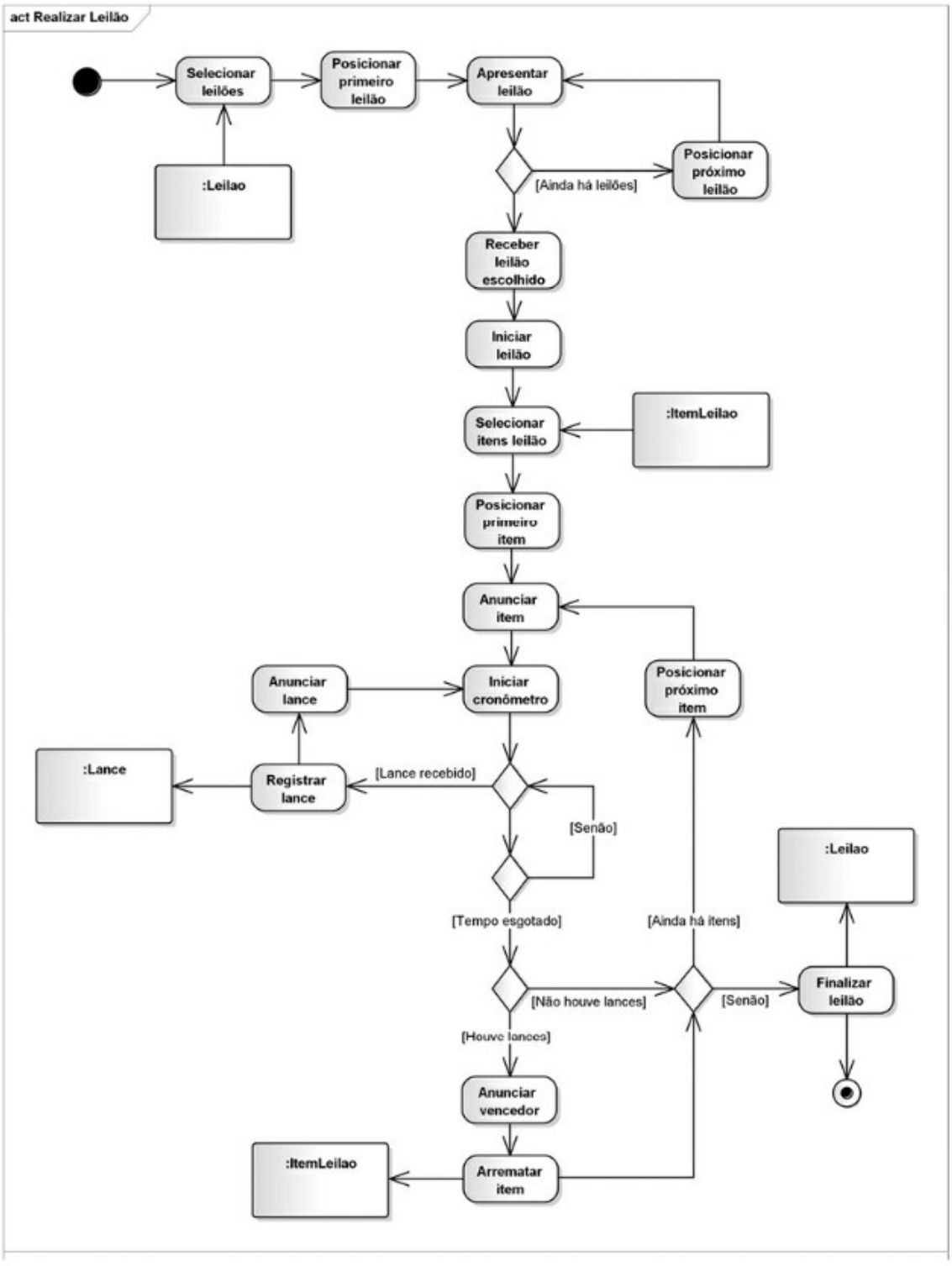
- Primeiramente, deve-se consultar o sócio que deseja pagar mensalidades.
- Após a consulta do sócio, deve-se consultar a(s) mensalidade(s) por ele devida(s).
- Se houver alguma mensalidade em atraso, deve-se calcular os juros referentes ao atraso do pagamento.
- Deve-se, então, apresentar a(s) mensalidade(s) devida(s) e aguardar que o sócio escolha quais deseja pagar.
- Finalmente, deve-se quitar a(s) mensalidade(s) escolhida(s).


:Mensalidade
Receber número do cartão
Consultar sócio
[Sócio localizado]
[Sócio não encontrado]
Emitir mensagem de sócio não encontrado
Selecionar mensalidades do sócio
[Não há mensalidades a pagar]
Emitir mensagem de que não há mensalidades a pagar
[Há mensalidades a pagar]
Posicionar primeira mensalidade
[Mensalidade em atraso]
Calcular juros
Apresentar mensalidade
[Ainda há mensalidades]
Posicionar próxima mensalidade
Selecionar mensalidades a pagar
Posicionar primeira mensalidade
:Mensalidade
Quitar mensalidade
[Ainda há mensalidades]
Posicionar próxima mensalidade
recibo
Emitir recibo de quitação

10.30.4 Sistema para Controle de Leilão Via Internet – Processo de Realizar Leilão

Desenvolva o diagrama de atividade referente ao processo de realizar leilão para um sistema de controle de leilão via internet, de acordo com os seguintes requisitos:

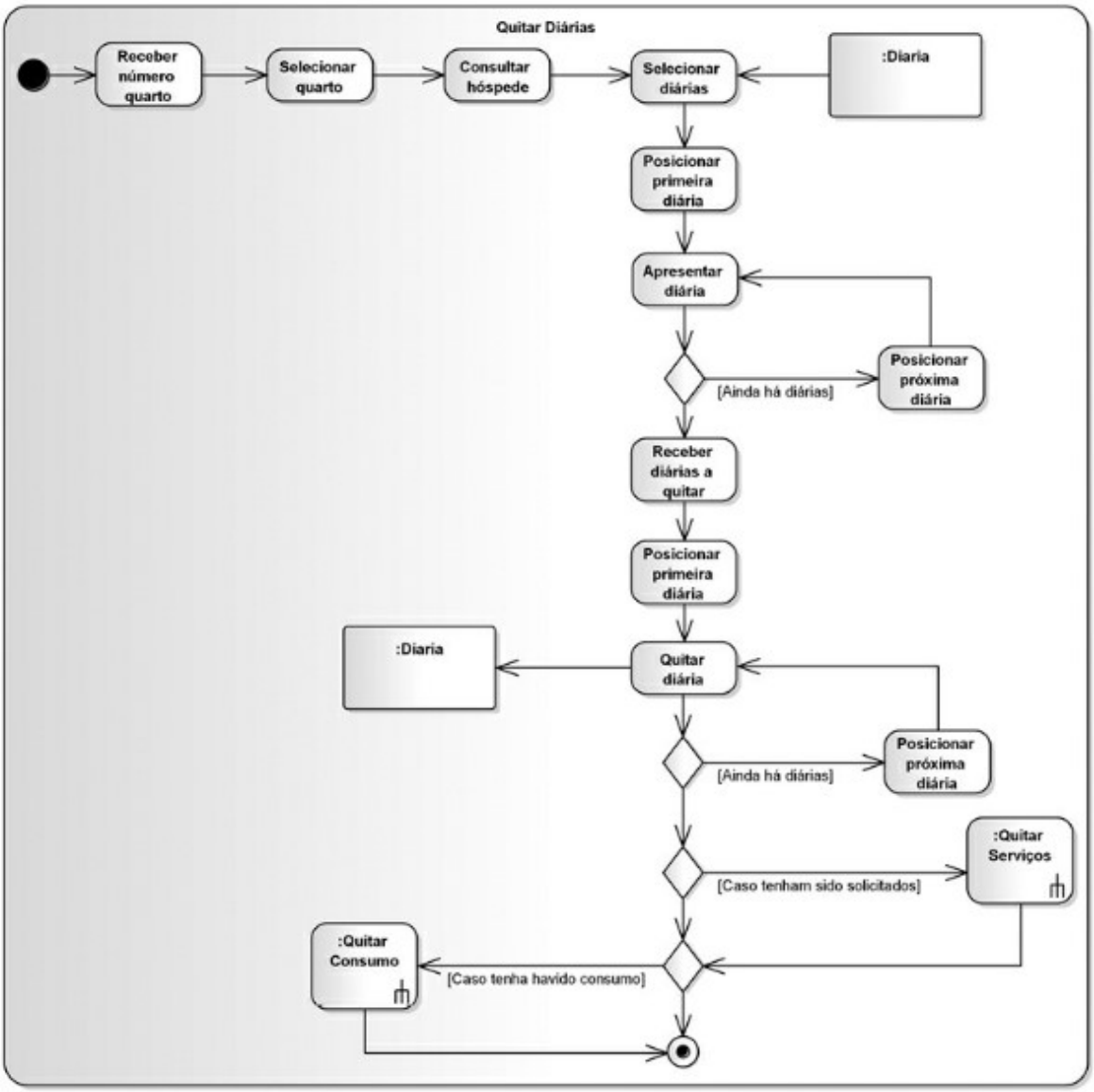
- Ao receber a solicitação do serviço de realizar leilões, o sistema deve carregar todos os leilões ainda não encerrados na interface.
- A partir dessa listagem, o leiloeiro deve selecionar qual leilão deseja iniciar.
- No momento em que um leilão é escolhido para ser iniciado, o sistema precisa carregar todos os itens a serem leiloados nele.
- A partir da listagem dos itens a serem leiloados, o leiloeiro deve escolher um item e anunciá-lo.
- Se houver algum lance para o item anunciado, o sistema deve anunciá-lo e, em seguida, registrá-lo.
- Existe um tempo máximo de espera para que haja lances. Enquanto esse tempo não for atingido, o item permanece sendo anunciado. Sempre que houver um lance, esse cronômetro é reiniciado.
- Quando o tempo máximo de espera por um lance for atingido, o processo deve verificar se houve ofertas para o item, caso em que se deve anunciar o participante que ofereceu o maior lance como vencedor. Caso contrário, deve-se simplesmente encerrar o anúncio do item.
- Depois de ter sido encerrado o anúncio de um item, deve-se verificar se ainda há itens a anunciar, caso em que o processo passa a anunciar o novo item. Caso contrário, o leilão deve ser encerrado.


act Realizar Leilão
Selecionar leilões
Posicionar primeiro leilão
Apresentar leilão
:Leilao
[Ainda há leilões]
Posicionar próximo leilão
Receber leilão escolhido
Iniciar leilão
Selecionar itens leilão
:ItemLeilao
Posicionar primeiro item
Anunciar item
Anunciar lance
Iniciar cronômetro
Posicionar próximo item
:Lance
Registrar lance
[Lance recebido]
[Senão]
:Leilao
[Tempo esgotado]
[Ainda há itens]
[Não houve lances]
[Senão]
Finalizar leilão
[Houve lances]
Anunciar vencedor
:ItemLeilao
Arrematar item

10.30.5 Sistema de Controle de Hotelaria – Processo de Pagamento de Diárias

Desenvolva um diagrama de atividade referente ao processo de pagamento de diárias para um sistema de controle de hotelaria, de acordo com as seguintes definições:

- No momento em que o hóspede informa o número do quarto para quitar as diárias, o sistema deve consultar o hóspede e todas as diárias devidas pelo aluguel do quarto, apresentando-as ao funcionário.
- A partir dessa listagem, deve-se quitar as diárias apresentadas.
- Se tiver ocorrido a solicitação de quaisquer serviços no período em que o quarto estava ocupado, estes deverão ser quitados também;
- Isso também ocorre se houver quaisquer consumos de frigobar, sendo necessário também os quitar.


Quitar Diárias
Receber número quarto
Selecionar quarto
Consultar hóspede
Selecionar diárias
:Diaria
Posicionar primeira diária
Apresentar diária
[Ainda há diárias]
Posicionar próxima diária
Receber diárias a quitar
Posicionar primeira diária
:Diaria
Quitar diária
[Ainda há diárias]
Posicionar próxima diária
:Quitar Serviços
[Caso tenham sido solicitados]
:Quitar Consumo
[Caso tenha havido consumo]