iscte
INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Módulo 3: Princípios de Desenvolvimento de Software

Aula 5
Diagramas de Sequência

UPskill

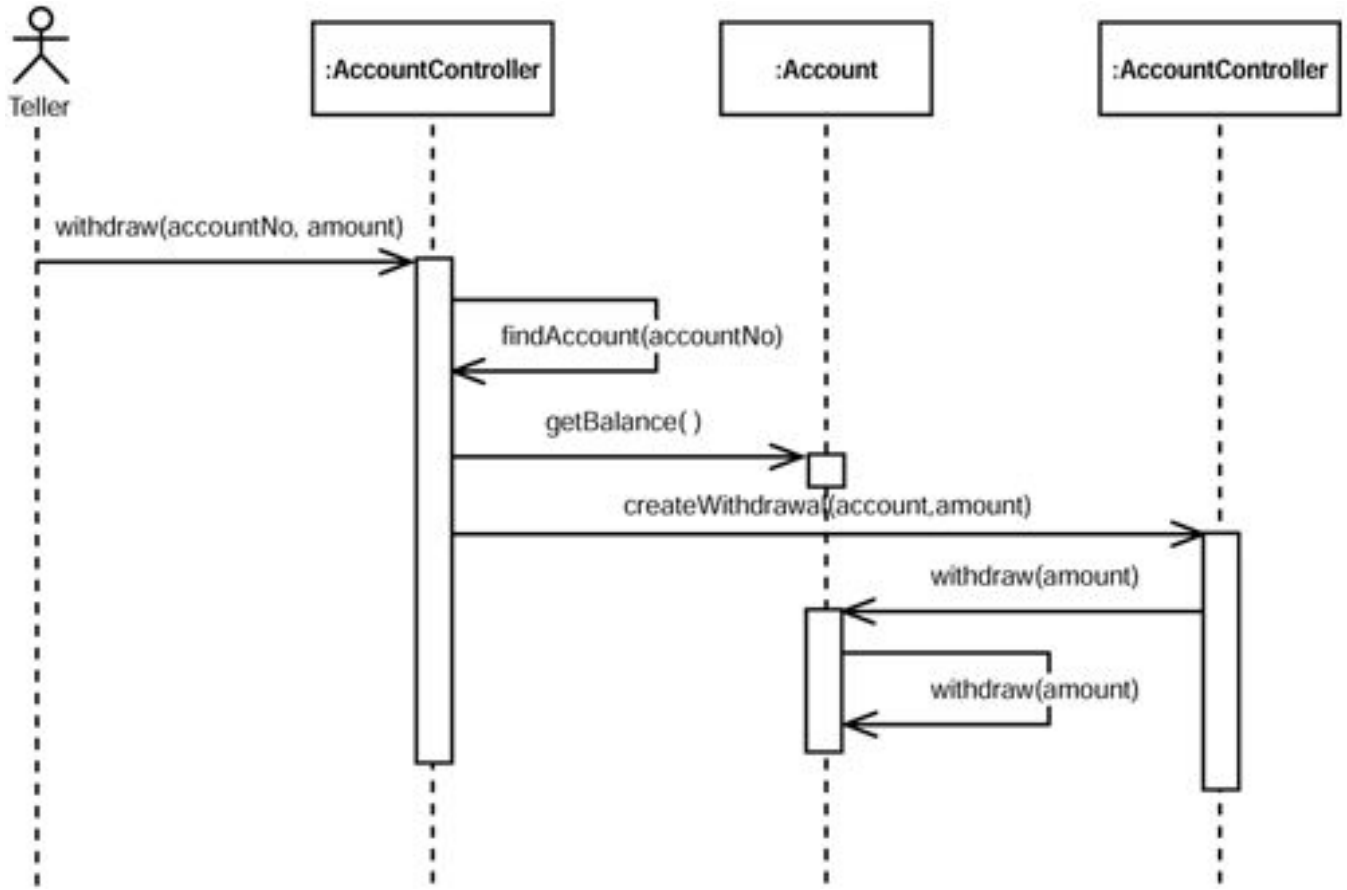
Recordando...

- Os **diagramas de actividades** constituem um elemento de modelação simples, mas eficaz para descrever fluxos de trabalho numa organização ou para detalhar operações de uma classe, incluindo comportamentos que possuam processamento paralelo.
- Os **diagramas de actividades** modelam uma actividade como uma sequência de passos (acções), pontos de decisão e ramos (cenários alternativos).

Diagramas de Sequência

- Os diagramas de *Sequência* são utilizados para representar a **sequência de processos** num programa de computador.
- Representa interações entre objetos, realizadas através de operações ou métodos.
- Este diagrama é construído a partir do Diagrama de Use Case. Primeiro, define-se qual o papel do sistema (Use Cases), depois, é definido como o software realizará seu papel (Sequência de operações).

Os Diagramas de Sequência representam a troca de mensagens entre objetos.

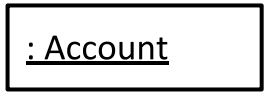

Teller
:AccountController
:Account
:AccountController
withdraw(accountNo, amount)
findAccount(accountNo)
getBalance( )
createWithdrawal(account,amount)
withdraw(amount)
withdraw(amount)

Diagramas de Sequência

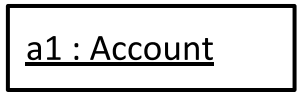
- Nos diagramas de *Sequência* são representados de forma cronológica:
  - **Objectos** (instâncias das classes do Diagrama de Classes);
  - **Trocas de mensagens** (essencialmente chamadas a métodos de objectos);
  - **Estruturas simples de controlo** (*if* e *repeat)*
- Pressupõe-se a prévia definição de um diagrama de classes com a indicação dos métodos associados a cada classe.

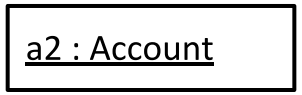
Objectos

- Um objecto representa uma instância de uma classe, como tal na sua representação deverá constar a indicação da classe respectiva.
- Um objecto da classe Account será representado por:



- Podemos no entanto criar instâncias das classes:





O que são mensagens?

- As mensagens trocadas entre objetos representam a invocação de um serviço (operação) disponibilizado por um objecto, com o objectivo de despoletar uma acção ou actividade.
- Uma definição mais formal descreve uma mensagem como a especificação da comunicação entre objectos.
- **Quando uma mensagem é enviada uma acção ocorre!**

Tipos de mensagens

- Existem diferentes tipos de mensagens:
  - Mensagens **síncronas** - O objecto emissor fica suspenso à espera de uma resposta. Utiliza-se esta mensagem quando o objecto emissor necessita de dados provenientes do objeto receptor, para continuar o ser processamento

Tipos de mensagens

- Existem diferentes tipos de mensagens:
  - Mensagens **assíncronas** - Permite que o objecto emissor continue o processamento antes de receber a resposta do receptor. Este tipo de mensagens é utilizado nos processos concorrentes.

Tipos de Mensagens

- Existem diferentes tipos de mensagens:
  - Mensagens **simples** - São utilizadas quando ainda não está definido o tipo de mensagem ou então quando este tipo não é relevante.

  - Mensagens **retorno** - Ilustra o retorna a uma mensagem enviada que poderá ser um valor ou um sinal.

Mensagens Específicas

- Existem mensagens específicas que desencadeiam certas acções:
  - **Call** - invoca sincronamente (fica bloqueado) um método de um objecto. Este tipo de mensagens pode ser enviada ao próprio objecto. [mensagem síncrona];
  - **Return** - Retorna uma mensagem para o objecto que invocou um Call. [mensagem de retorno];

Mensagens Específicas

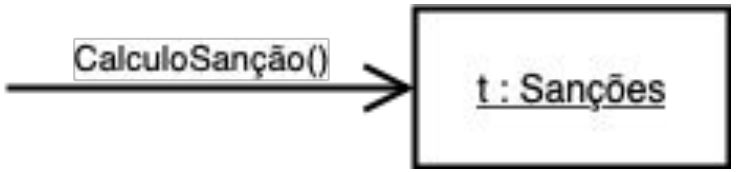
- Existem mensagens específicas que desencadeiam certas acções:
  - **Send** - Envia um sinal a um objecto. [mensagem assíncrona];
  - **Destroy** - Destrói um objecto (um objecto pode destruir-se a ele próprio).
  - **Create** - Cria um objecto;

Mensagens - V

- **Call**

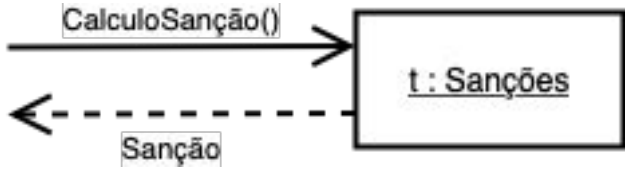
  Podem ser passados parâmetros:

  CalculoSanção (num dias atraso, tipo cliente).

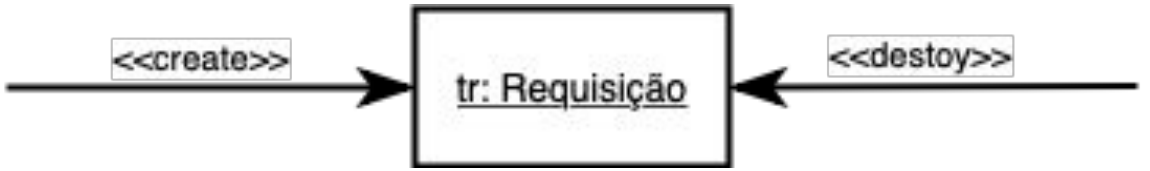


- **Return**

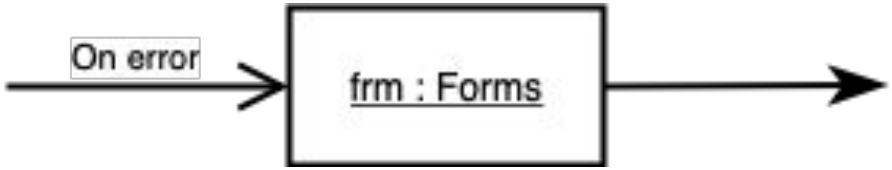
  Representa o retorno de um call.



- **Create** / **Destroy**



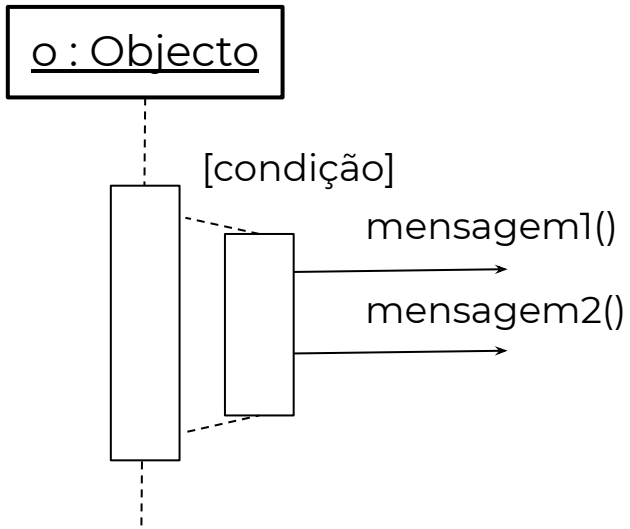
- **Send**



Processos Paralelos

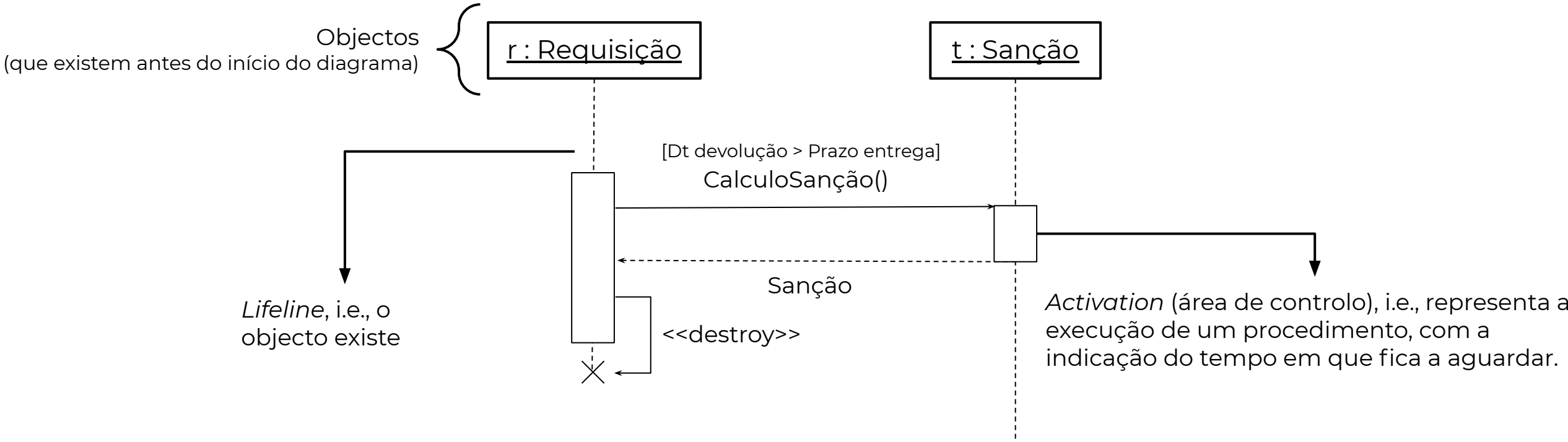
- A utilização de mensagens assíncronas permite efectuar diagramas de sequência onde são enviadas mensagens para objectos distintos que estarão a correr em paralelo.

Representação Gráfica dos Diagramas

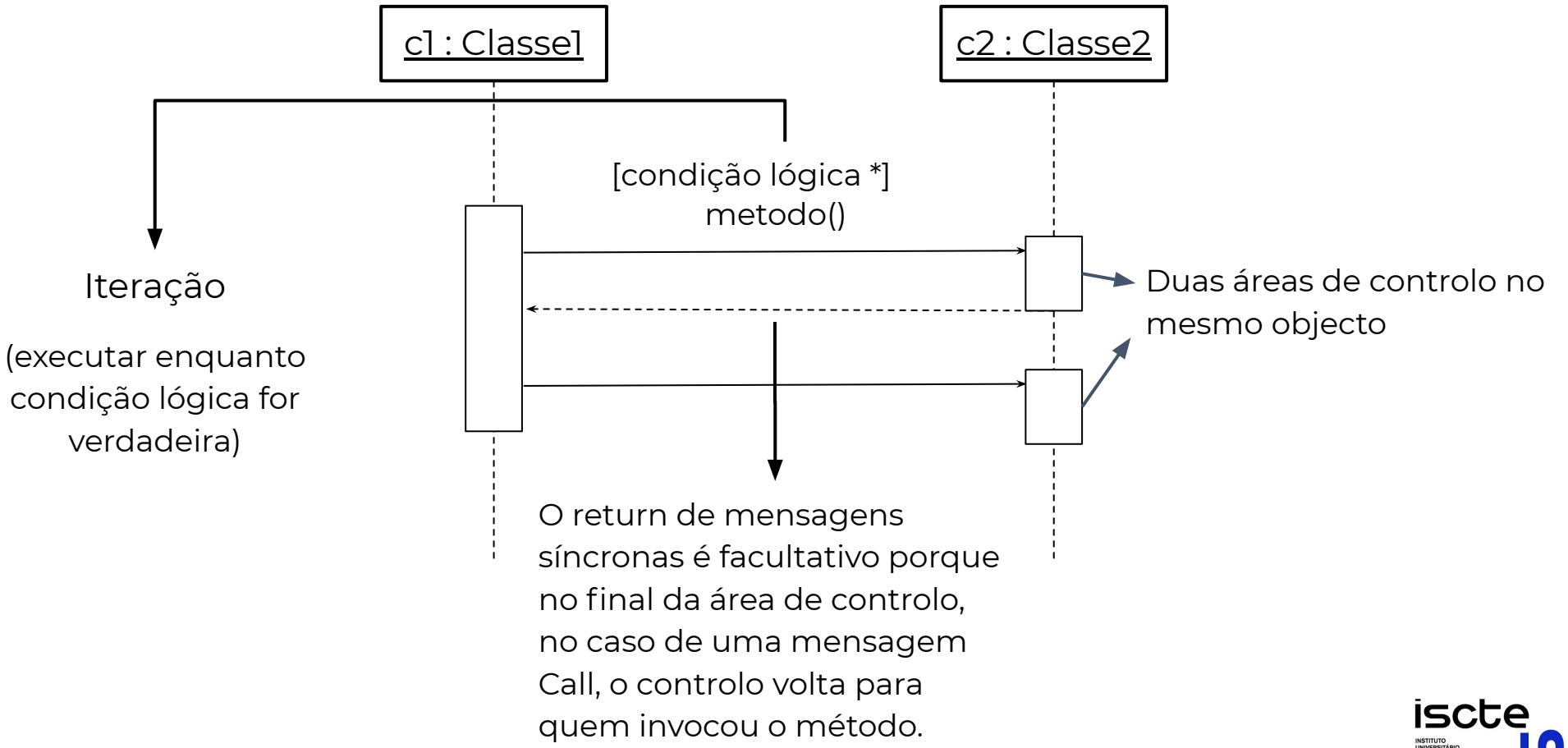


De forma a demonstrar que um conjunto de mensagens são enviadas de acordo com uma condição, pode-se utilizar uma separação na linha de controlo.

Representação Gráfica dos Diagramas

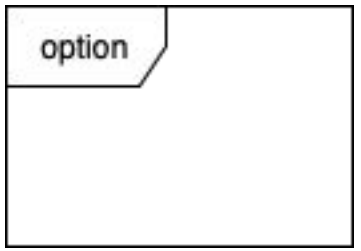


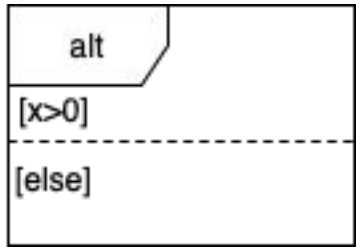
Representação Gráfica dos Diagramas - II

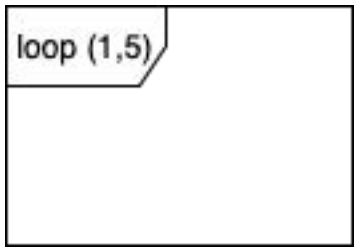


Estruturas de Controlo

- Pode-se representar com maior detalhe algumas zonas.
  - **Option** - Caso uma condição lógica seja verdadeira é invocado o método no interior desta região.
  - **Alt** - A utilização desta região explicita os caminhos alternativos considerados no diagrama. As alternativas são separadas por uma linha a tracejado.
  - **Loop** - Uma região Loop permite realçar uma região que é executada várias vezes.

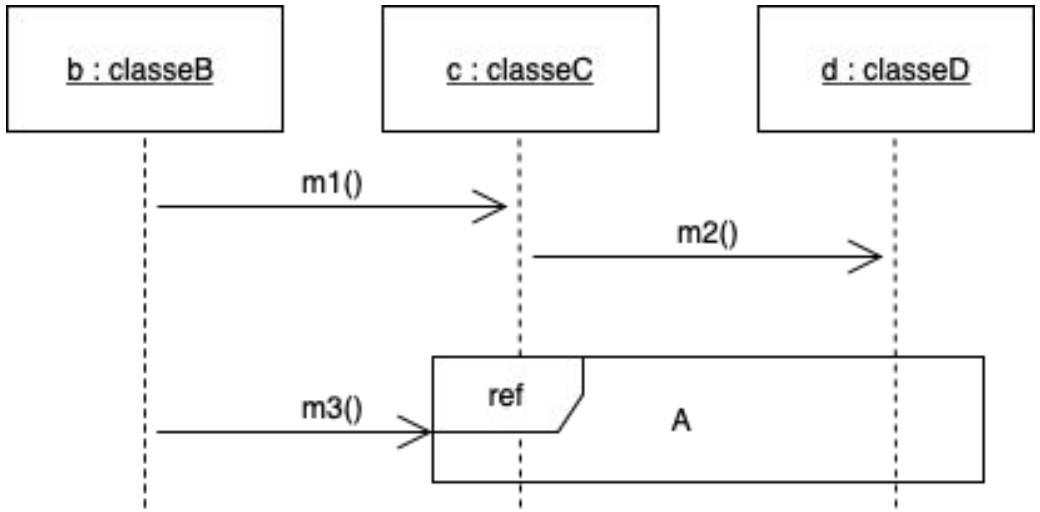


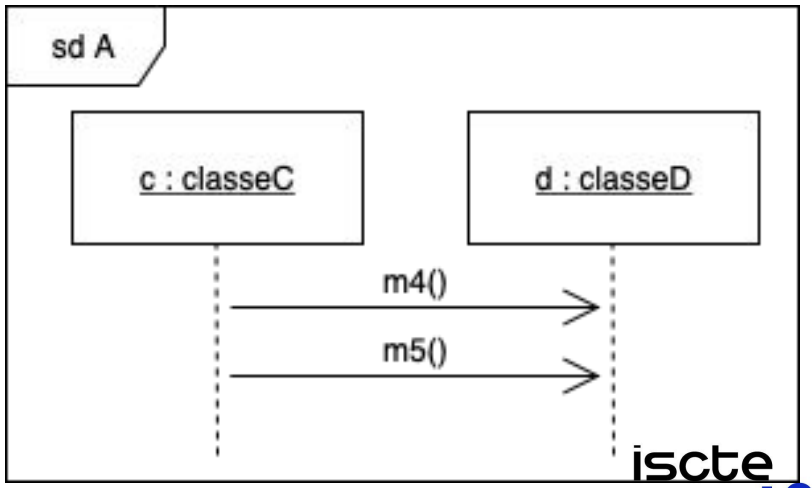




Sub Diagramas

- É possível criar subdiagramas dentro de um diagrama. Esta funcionalidade é útil quando, por exemplo, uma sequência de troca de mensagens ocorre mais do que uma vez dentro de um diagrama.





Mas o Diagrama de Sequência já corresponde ao nível de programação?

Vamos considerar este exemplo:

Vamos calcular o teorema de Pitágoras: $h^2=a^2+b^2$

E vamos considerar o seguinte diagrama de classes:

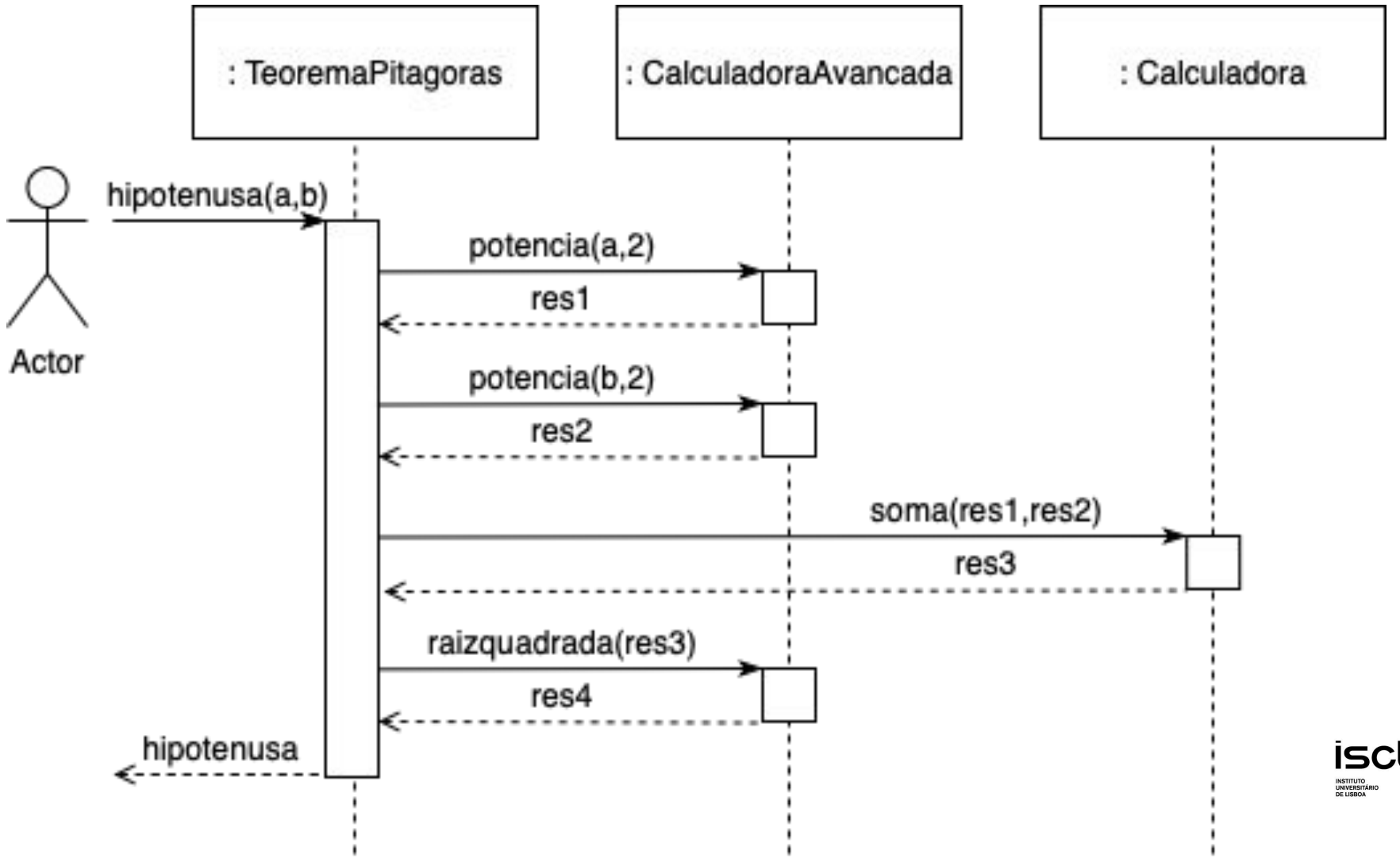
TeoremaPitagoras
hipotenusa(a,b)

CalculadoraAvançada
potencia(x,e) raizquadrada(x)

Calculadora
soma(a,b) subtracao(a,b)

Como fica o diagrama de sequência?

Diagrama de Sequência



Como seria a codificação em Java?

```
public class TeoremaPitagoras {

    public double hipotenusa(double a, double b){
        double res1 = CalculadoraAvancada.potencia(a, 2);
        double res2 = CalculadoraAvancada.potencia(b, 2);
        double res3 = Calculadora.soma(res1, res2);
        double res4 CalculadoraAvancada.raizquadrada(res3);
        return res4;
    }

}
```

Vamos praticar?

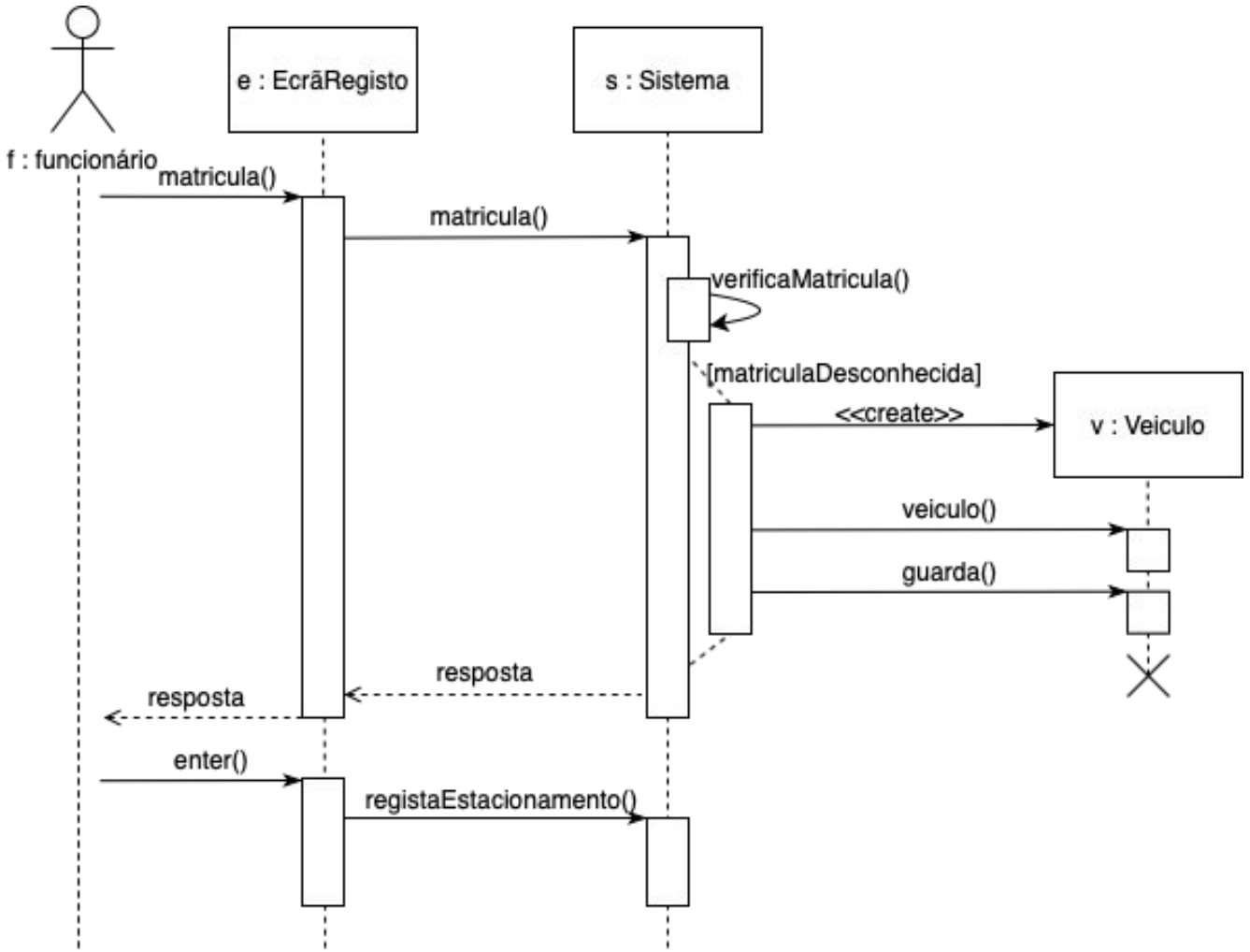
Exercício 1

Considere o seguinte sistema informático para a gestão de um parque de estacionamento. O controlo é efectuado com base na matrícula do veículo. Na entrada do parque existirá um funcionário que introduz as matrículas no sistema.

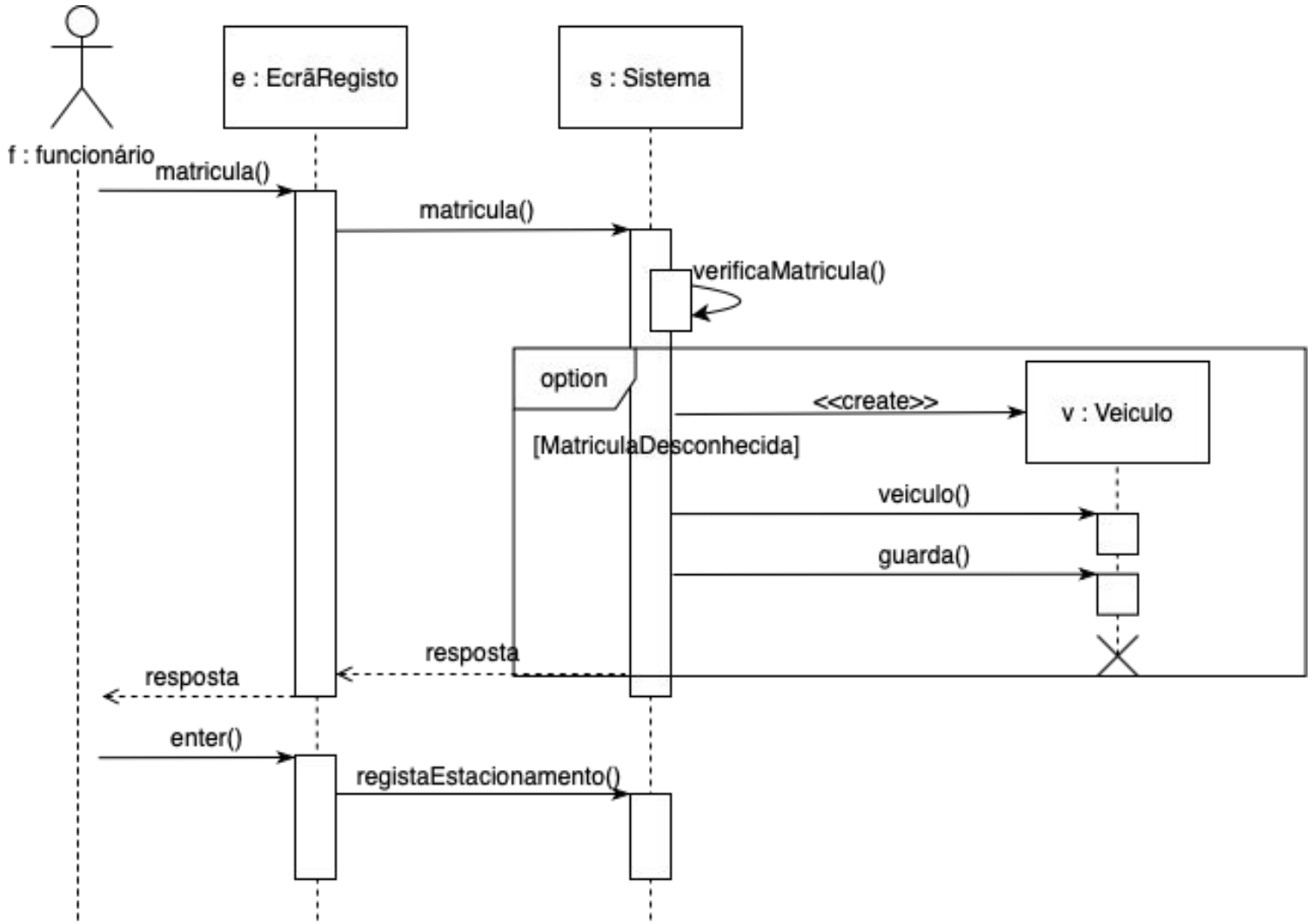
O sistema tem que verificar se a matrícula existe.
O use case começa quando o funcionário introduz uma matrícula no seu terminal. Caso a matrícula não seja reconhecida, será efectuado o registo do novo veículo. Após o funcionários receber uma resposta do sistema, confirma o estacionamento pressionando a tecla Enter. O sistema regista a data e a hora do início do estacionamento.

Elabora o diagrama de Sequência do sistema descrito.

Exercício 1 - Resolução



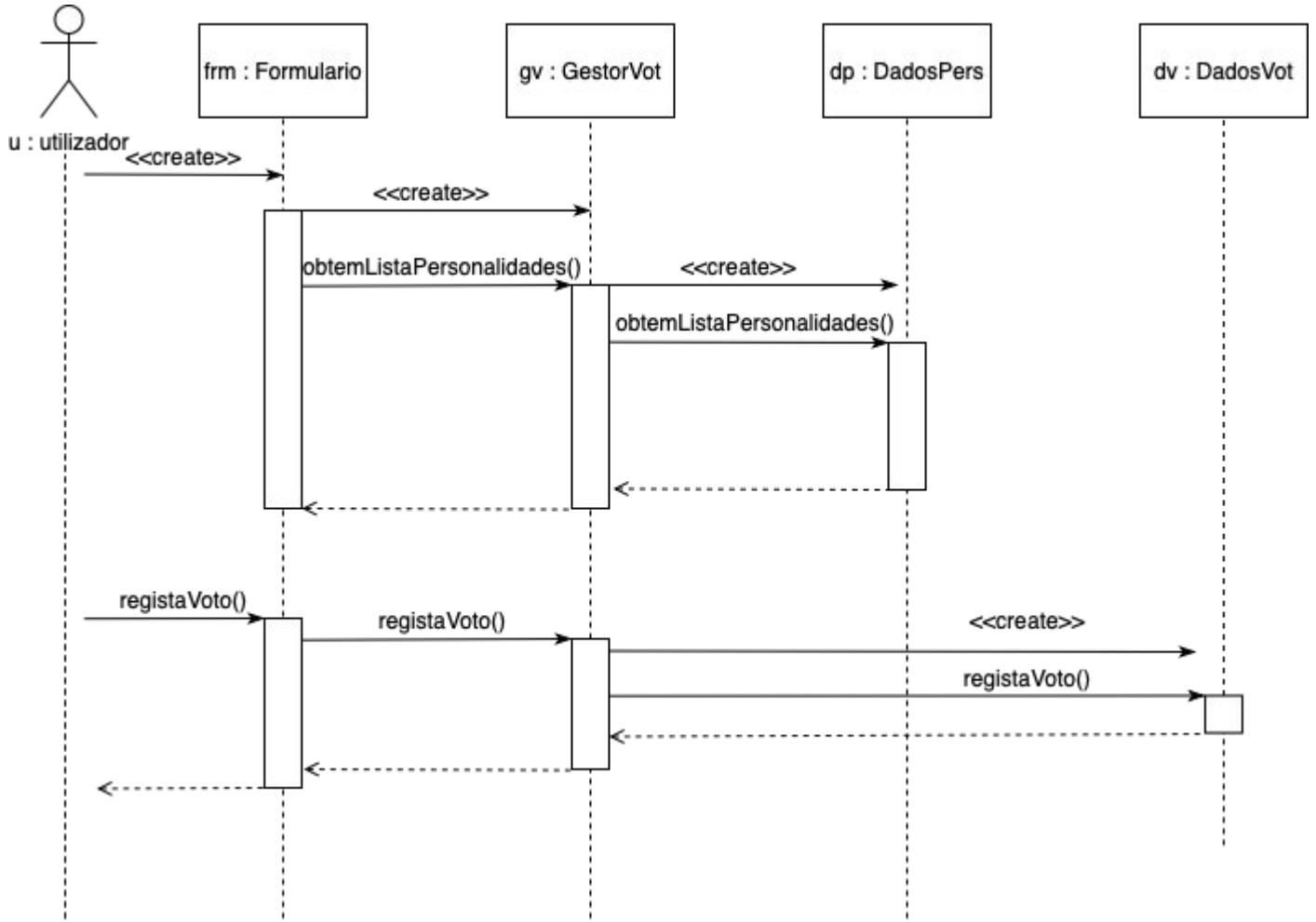
Exercício 1 - Resolução Alternativa



Exercício 2

Pretende-se desenhar um diagrama que especifique as interações que suportam um formulário para um sistema informático de apoio à eleição de personalidades que mais se destacaram no século XX.
Assim que o utilizador abre um formulário, é-lhe mostrada uma lista de personalidades elegíveis que é carregada através da classe “DadosPers”. Para proceder à votação, o utilizador seleciona um nome da lista. O voto fica registado nos “DadosVot”.
O sistema informático deve possuir uma arquitectura estruturada por camadas, nomeadamente evidenciando a distinção entre classes de interface de utilizador (formulário), classes de controlo (ou gestão) e classes de disponibilização de dados.

Exercício 2 - Resolução



Exercício 3

Pretende-se que uma aplicação informática de gestão comercial elabora estatísticas mensais de vendas de produtos, ou seja, para cada mês pretende-se saber o valor das vendas de cada produto. Para além dos valores de vendas, pretende-se também efectuar análises estatísticas (análise sazonal, análise de clusters e análises e correlações) aos valores mensais.

Formulário de Suporte

A aplicação disponibiliza um formulário onde o utilizador apenas tem de indicar o ano sobre o qual vão ser calculadas as estatísticas. Pode-se assumir que o resultado do processamento é exportado para ficheiro.

Exercício 3 - Cont.

Assuma a existência das seguintes classes.

fmrEstatisticas (fmr)
processaEstMensal(int ano) : Object

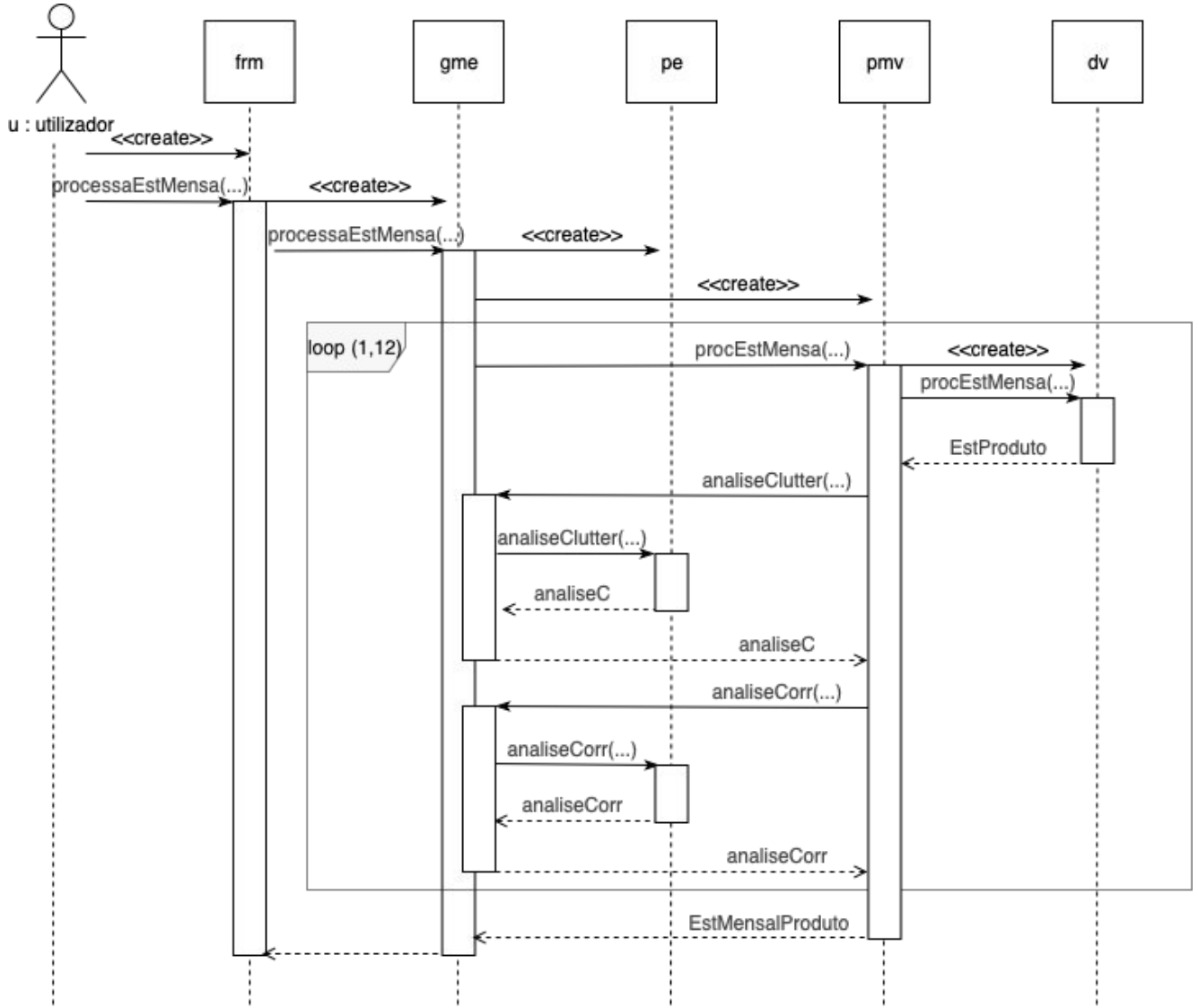
GestorMapasEstatisticas (gme)
analiseSazonal(int idProd) : Object analiseCorr(int idProd) : Object analiseClustter(int idProd) : Object processaEstMensal(int ano) : Object

DadosVendas (dv)
procEstMensal(int m, int a) : Object

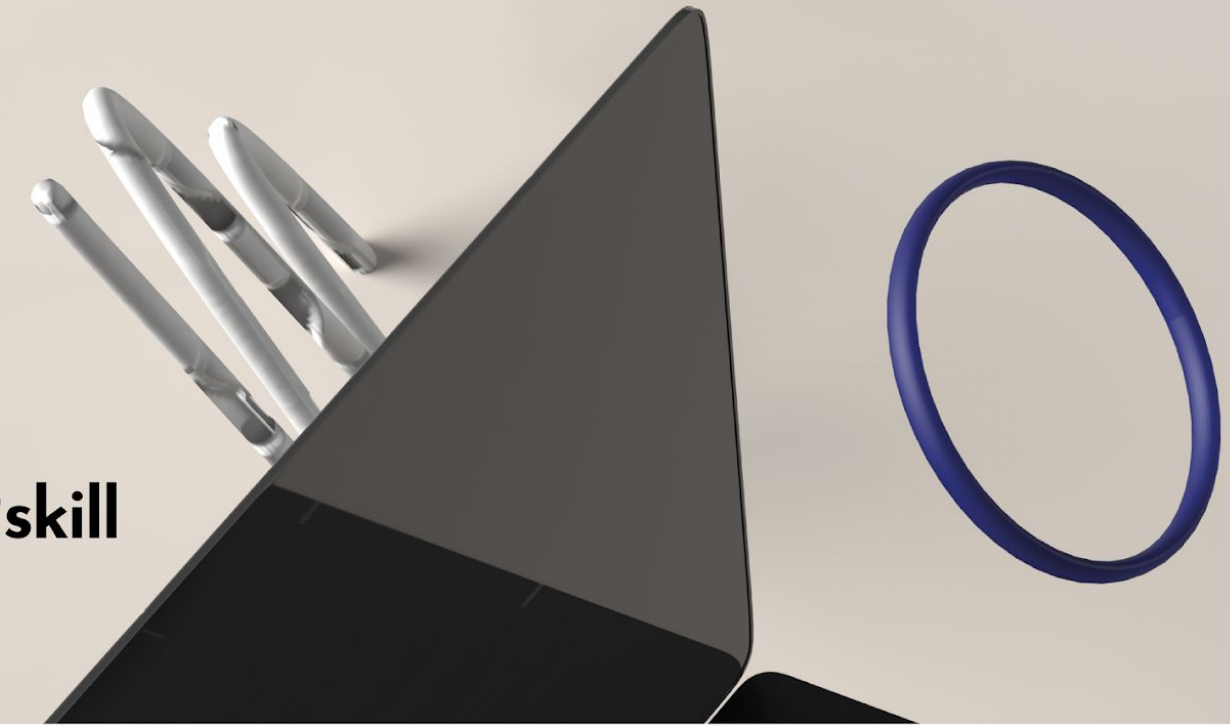
ProcessamentoMensalVendas (pmv)
procEstMensal(int m, int a) : Object

ProcessamentoEstatisticas (pe)
analiseCorr(int idProd) : Object analiseClustter(int idProd) : Object

Exercício 3 - Resolução



O futuro profissional começa aqui



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

UPskill