P.O.O. COM ÊNFASE EM PADRÕES DE SOFTWARE

MODELAGEM OO

Prof. Marcius Brandão

Mestre em Ciência da Computação

Engenharia de Software

Atividade 1 (Resposta)

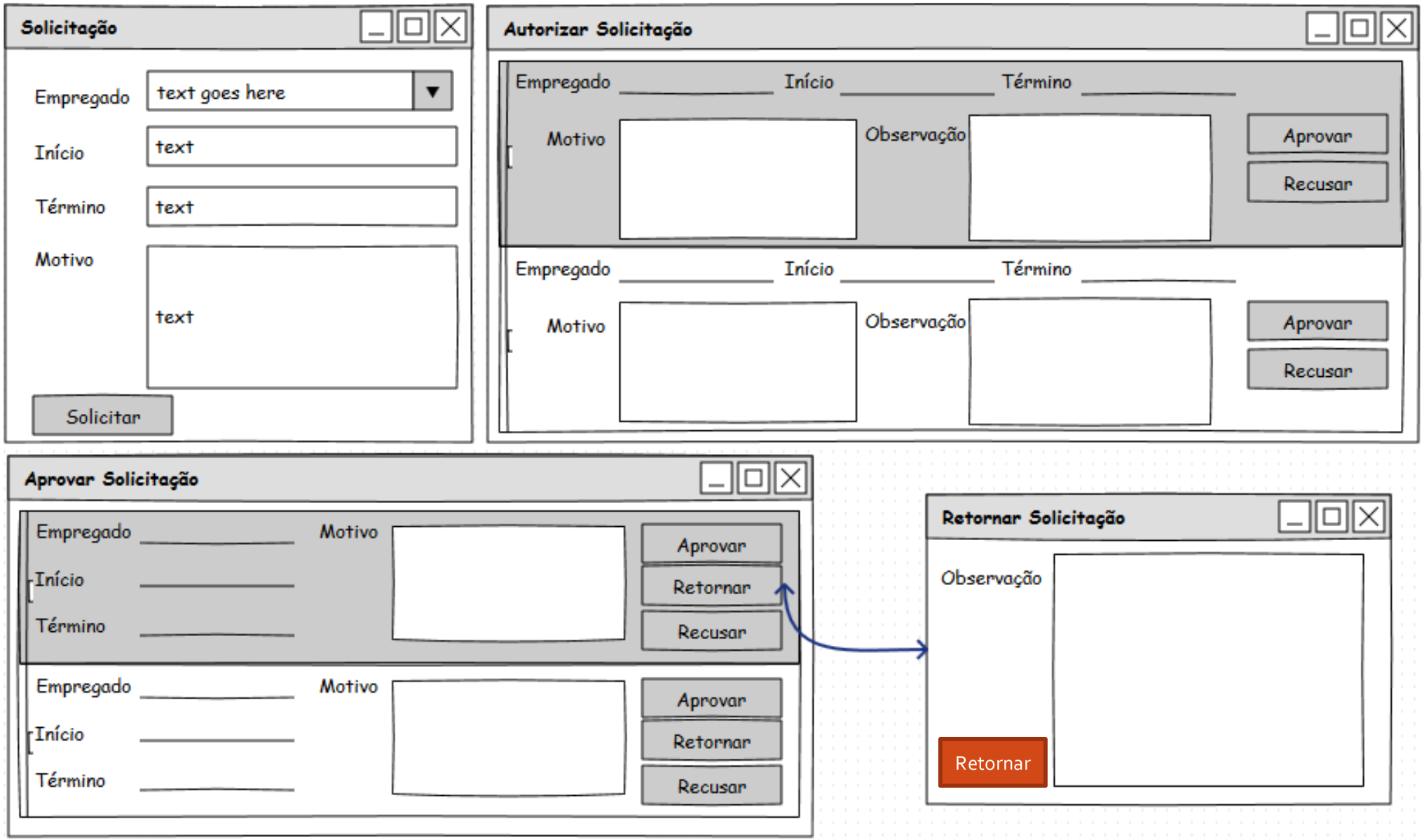
- O que é **Anemic Domain Model?**
  - Frequentemente, **entidades sem lógica de negócio com comportamentos codificados isoladamente** nos denominados business objects caracterizam fortemente o **modelo de domínio anêmico**.
  - É muito fácil acabar jogando a lógica de negócio que poderia estar de certa forma em nossas entidades diretamente em Actions do Struts, ActionListeners do Swing e Managed Beans do JSF.
  - Este modelo acaba ficando com um forte apelo procedural, e vai diretamente na contra mão de boas práticas de orientação a objetos e do Domain-Driven Design.
    - Arquitetura e Design de Software, Caelum, 2009

ESTUDO DE CASO - PARTE I
REQUISITOS

Requisitos

- Solicitação de abono de faltas
  - Uma empresa pretende automatizar seu fluxo atual de solicitação de abono de faltas
  - Cada empregado deverá solicitar o abono de faltas informando o período e o motivo da falta.
  - A chefia imediata deverá analisar as solicitações e aprovar ou não. Ao aprovar, a solicitação será encaminhada para análise do RH.
  - O RH deverá analisar as solicitações aprovadas pela chefia imediata e emitir o aval final (aprovar ou reprovar). O RH poderá, por algum motivo, retornar a solicitação para a chefia imediata. Neste caso, o RH deverá informar o motivo do retorno.

Protótipo


Solicitação
Empregado
text goes here
Início
text
Término
text
Motivo
text
Solicitar
Autorizar Solicitação
Empregado
Início
Término
Motivo
Observação
Aprovar
Recusar
Empregado
Início
Término
Motivo
Observação
Aprovar
Recusar
Aprovar Solicitação
Empregado
Início
Término
Motivo
Aprovar
Retornar
Recusar
Empregado
Início
Término
Motivo
Aprovar
Retornar
Recusar
Retornar Solicitação
Observação
Retornar

ESTUDO DE CASO - PARTE II
ANÁLISE OO

Atividade 2 – Análise OO com UML

- A partir dos requisitos, analise o problema utilizando a UML 2.0 e apresente os seguintes artefatos elaborados:
  - Diagrama de Atividades
  - Casos de Uso (Diagrama e Narrativa)
  - Máquina de Estado
  - Diagrama de Classes

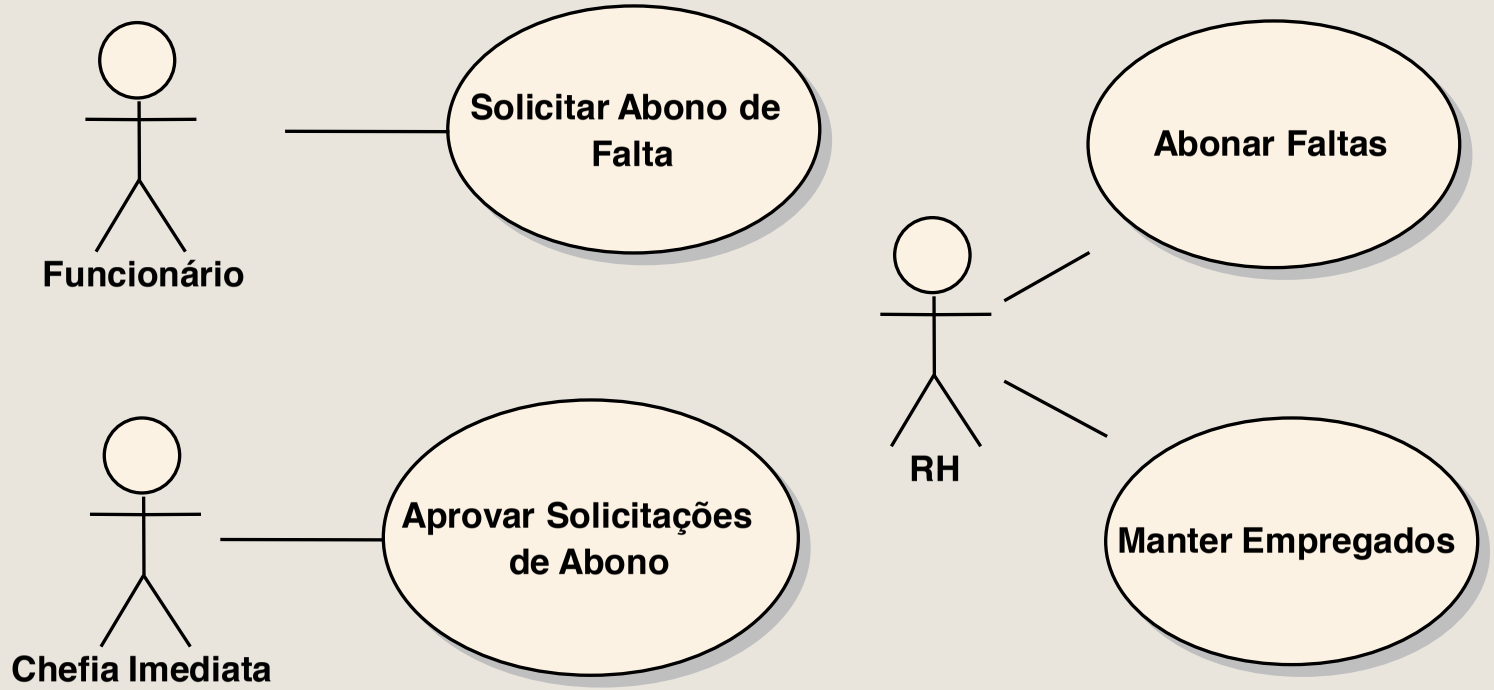
MODELE O QUE PRECISA SER MODELADO!

Criando um Domain Model

- O *Domain Model* representa a principal ferramenta de comunicação entre os desenvolvedores e os especialistas do domínio, e quanto melhor for esta comunicação, melhor será o software desenvolvido a curto e em longo prazo (Nilsson, 2006).
- Existem várias formas de se criar e representar um *Domain Model*. Uma boa opção para construir e visualizar o modelo é utilizando a UML, mas também é possível a modelagem diretamente via código ou outros artefatos
- Uma boa forma de iniciar a construção do Domain Model é identificando os *Domain Patterns*, seus atributos, comportamentos e relacionamentos.

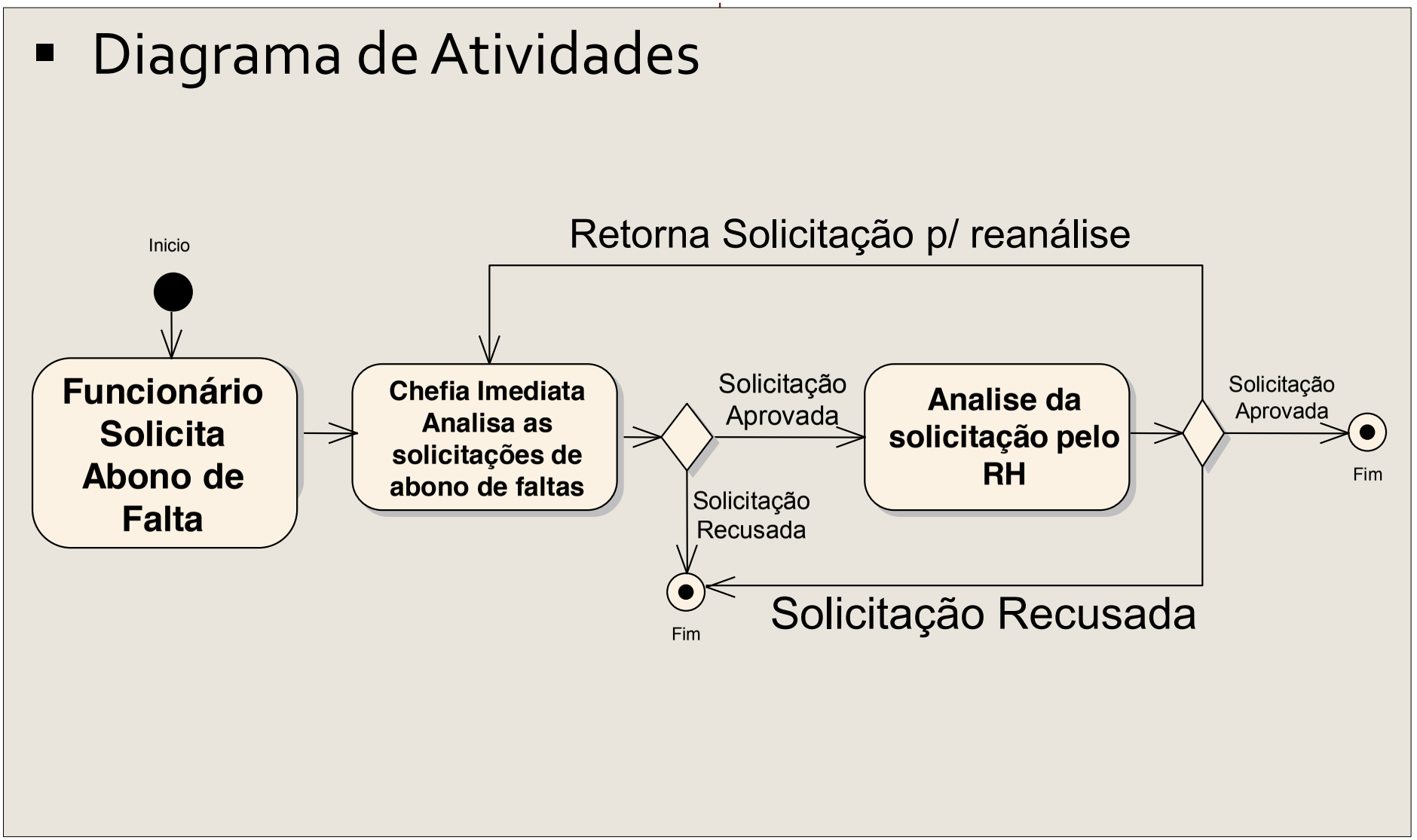
Análise UML

- Diagrama de Casos de Uso



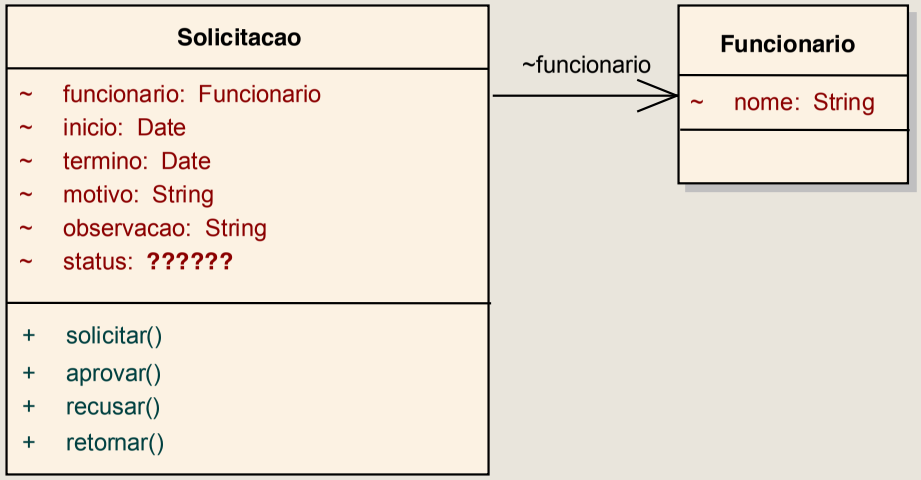
Análise UML

- Diagrama de Atividades



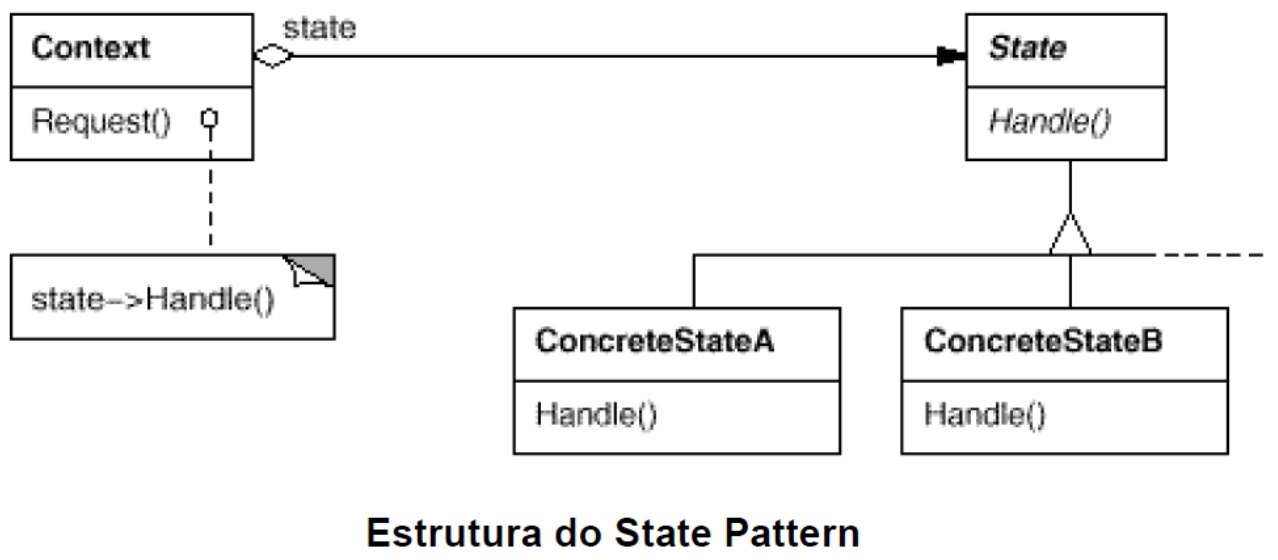
Análise UML

- Diagrama de Classes



Análise UML

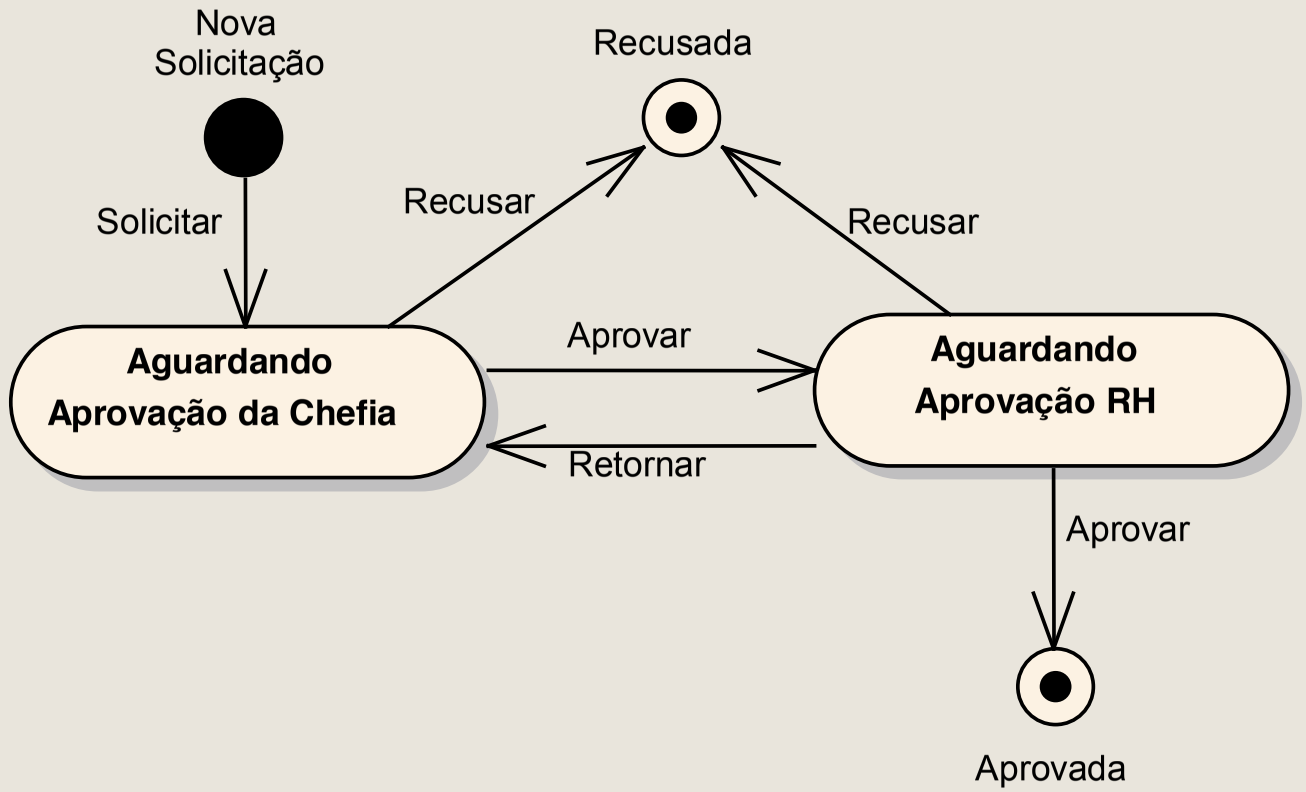
- A Máquina de Estados
  - Um *Objeto pode ter diferentes estados e pode haver regras para a transição de um estado para outro, bem como regras de comportamento do Objeto de acordo com o seu estado.* Para resolver este problema é necessário utilizar uma máquina de estados.



Estrutura do State Pattern

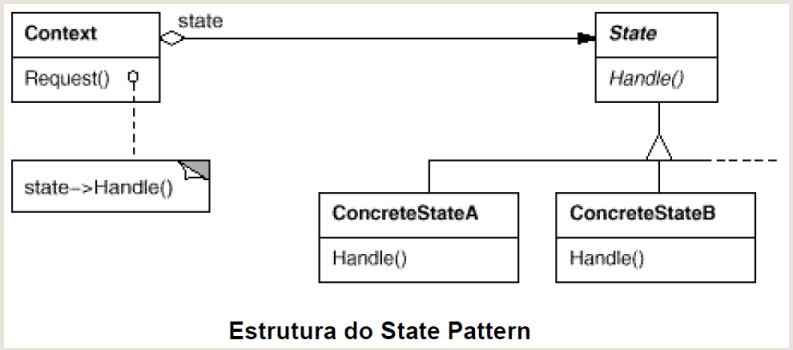
Análise UML

- Máquina de Estados da Solicitação

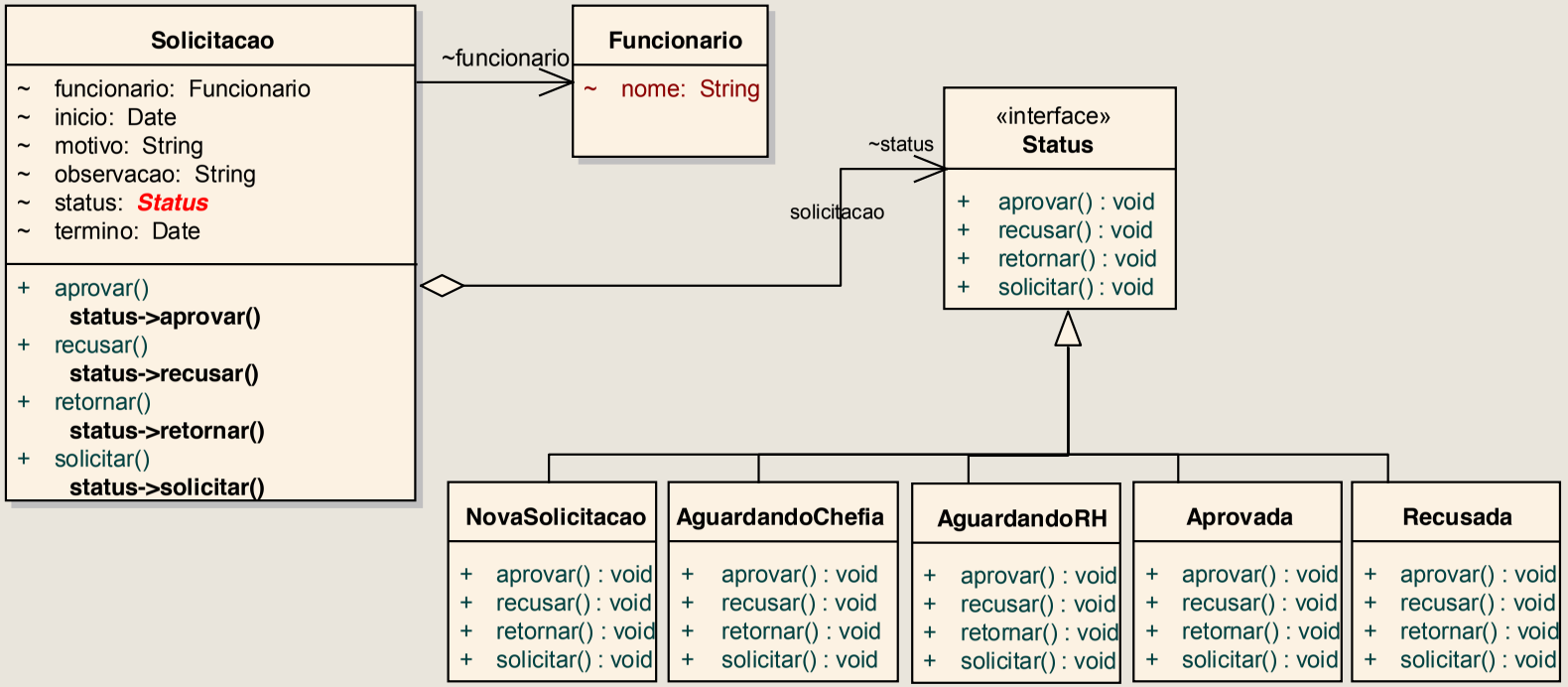


Análise UML

- Diagrama de Classes

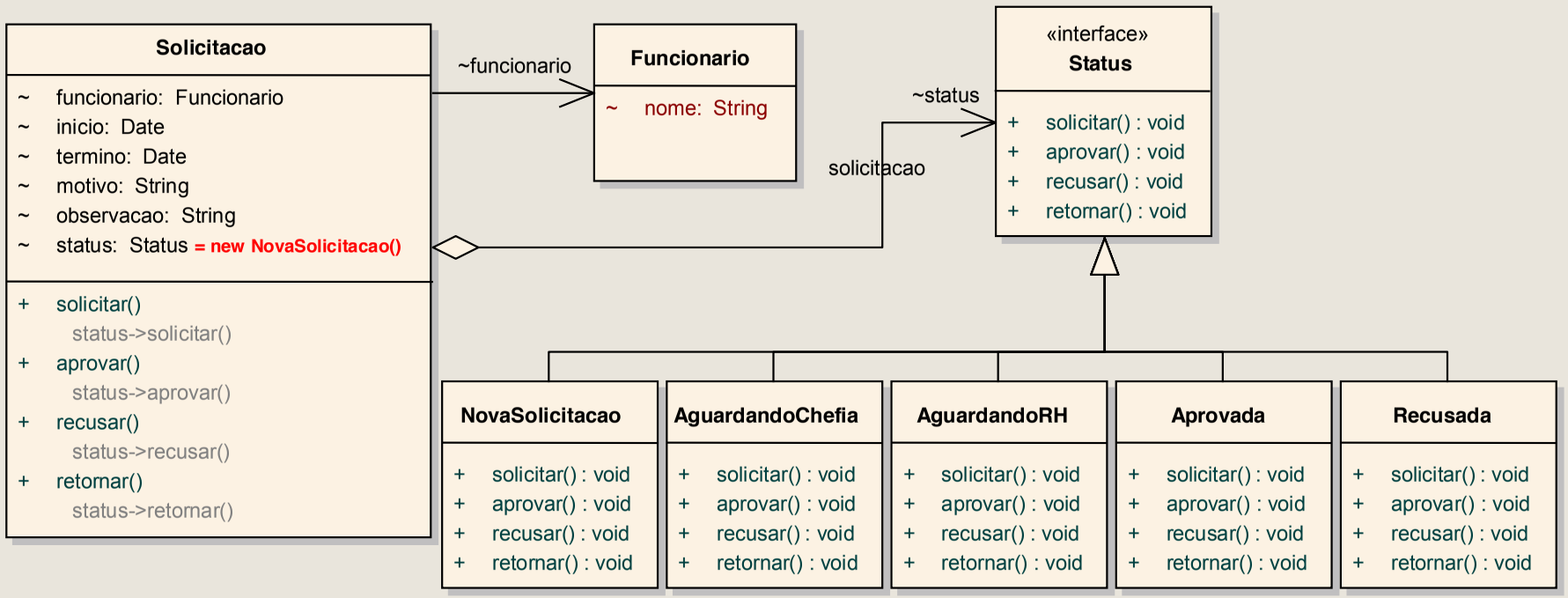


Estrutura do State Pattern



Análise UML

- Diagrama de Classes



Análise UML

- Caso de Uso : Cadastrar Empregado
  - Ator : RH
  - Este caso de uso é responsável por incluir e alterar um empregado.
  - Segue o padrão CRUD

Análise UML

- Caso de Uso : Solicitar Abono de Falta
  - Ator : Empregado
  - O Ator inicia o caso de uso selecionando "Solicitar Abono"
  - O sistema exibe a interface para a solicitação
  - O Ator informa o período e o motivo da falta
  - O Ator confirma os dados da solicitação[A1].
  - O sistema informa que a solicitação foi realizada c/ sucesso
  - Fluxo Alternativo A1 – Solicitação com datas futura
    - O sistema informa que uma solicitação não pode ser para datas futura.
    - Volta ao passo anterior

Análise UML

- Caso de Uso : Aprovar Solicitação de Abono de Falta
  - Ator : Chefia Imediata

1. O Ator inicia o CDU selecionando "Analisar solicitações"
2. O sistema exibe as solicitações em "Aguardando Aprovação"
3. O Ator confirma a solicitação selecionando "Aprovar"[A1]
4. O sistema encaminha a solicitação para aprovação do RH e informa que a solicitação foi aprovada c/ sucesso

  - Fluxo Alternativo A1 – Solicitação recusada
    - O Ator não aprova a solicitação selecionando "Recusar"
    - O sistema informa que a solicitação foi recusada e volta ao passo 2

Análise UML

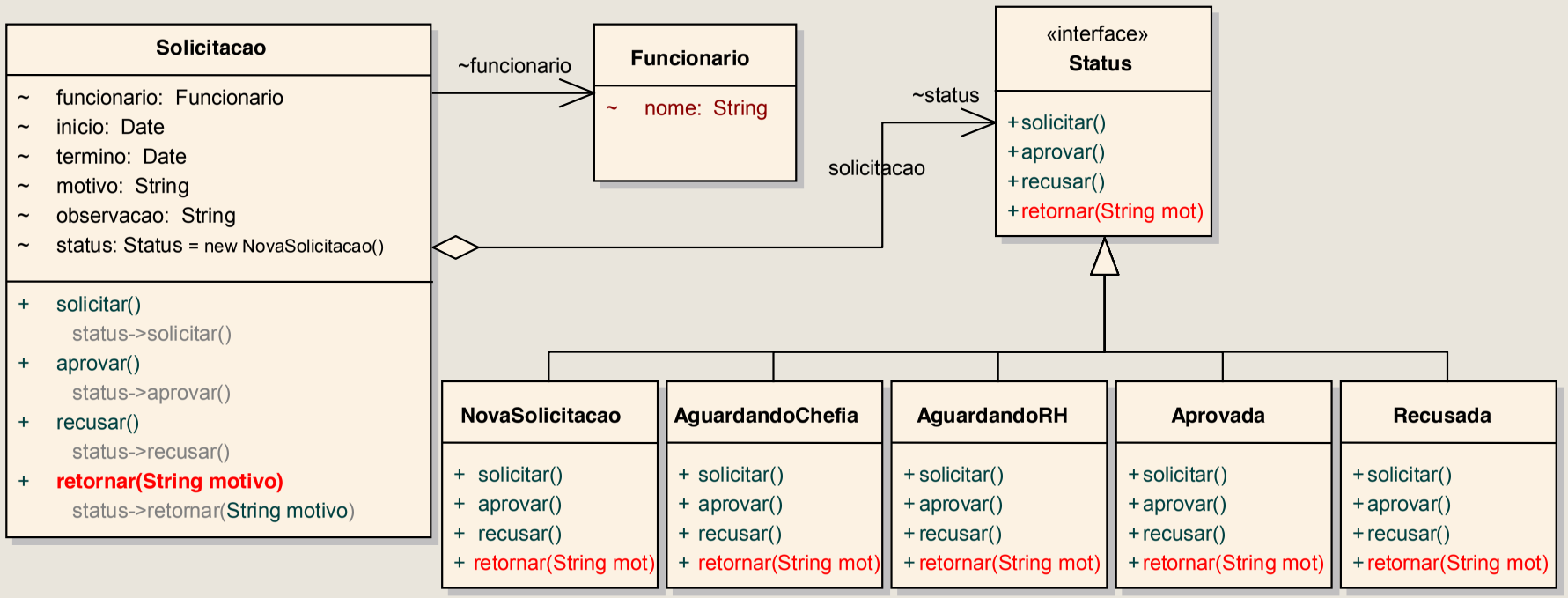
- Caso de Uso : Abonar Faltas
  - Ator : RH

1. O Ator inicia o CDU selecionando "Abonar Faltas"
2. O sistema exibe as solicitações "Aguardando Aprovação RH"
3. O Ator confirma a solicitação selecionando "Aprovar"[A1,A2]
4. O sistema informa que a solicitação foi aprovada c/ sucesso

  - Fluxo Alternativo A1 – Solicitação recusada
    - O Ator não aprova a solicitação selecionando "Recusar"
    - O sistema informa que a solicitação foi recusada e volta ao passo 2
  - Fluxo Alternativo A2 – Retornar solicitação
    - ***O ator retorna a solicitação para a chefia informando o motivo***
    - O sistema informa que a solicitação foi retornada e volta ao passo 2

Análise UML

- Diagrama de Classes



Atividade

- Implemente em JAVA o Diagrama de Classes do Estudo de Caso:
  - Mais ou menos umas 150 linhas de código
  - Para os atributos do tipo data/hora utilize a classe **java.util.Date**.
  - Se em algum método você não souber o que implementar adicione apenas o seguinte código ao método:
    - **throw new UnsupportedOperationException("Not supported yet.");**