Programa Formação de Talentos

Micro Mídia Informática
Fevereiro/2009

UML

- Introdução
- Fases de Desenvolvimento
- Notação
- Visões
- Análise de Requisitos
- Casos de Uso
- StarUML
- Criando Casos de Uso
- Orientação a Objetos
- Diagrama de Classes

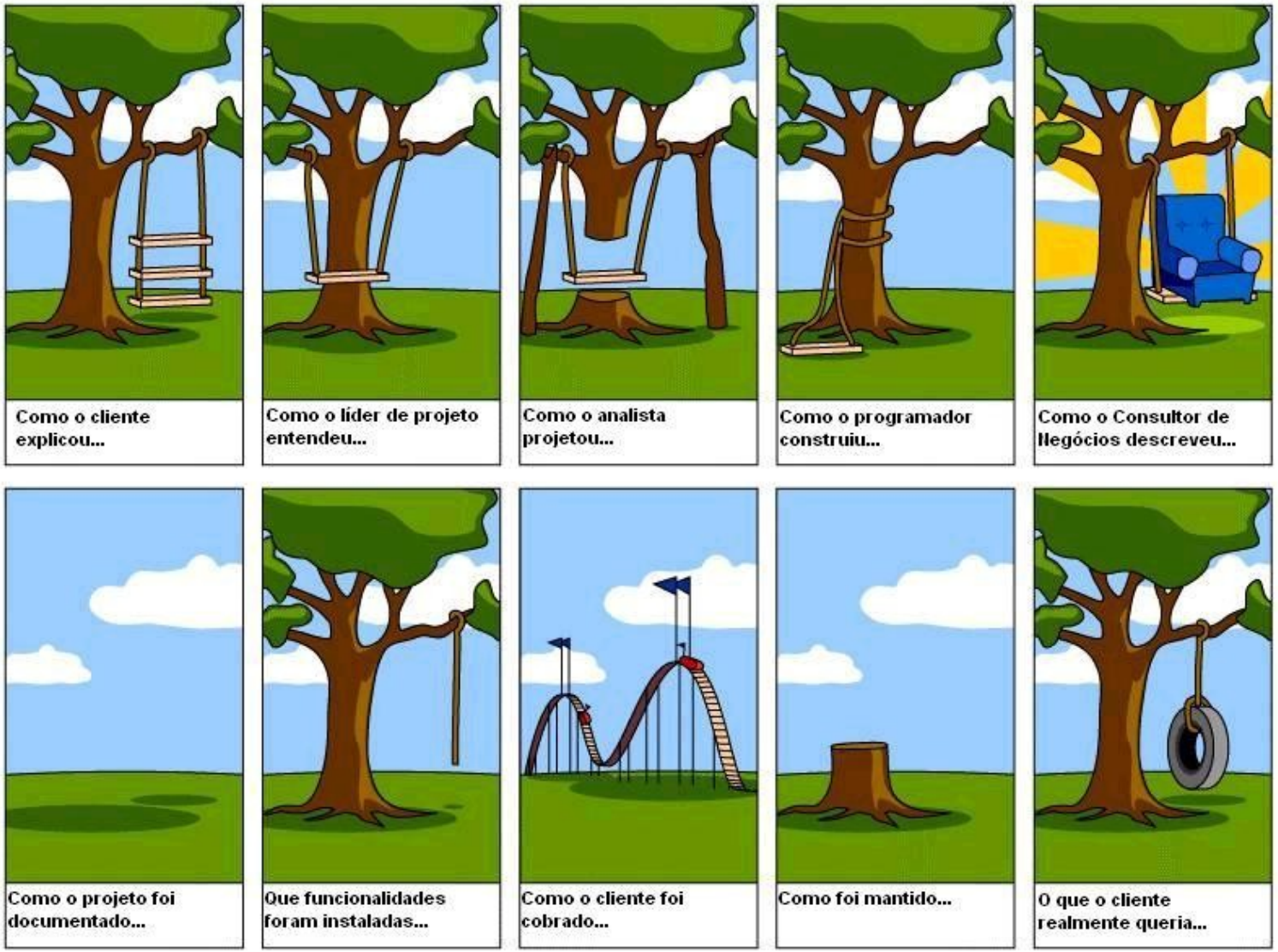
UML

- Classes
- Definição de Perfil
- Geração de Código-fonte
- Associação
- Generalização
- Relacionamentos
- Diagrama de Seqüência
- Diagrama de Estados
- Diagrama de Componentes
- Diagrama de Distribuição

Introdução

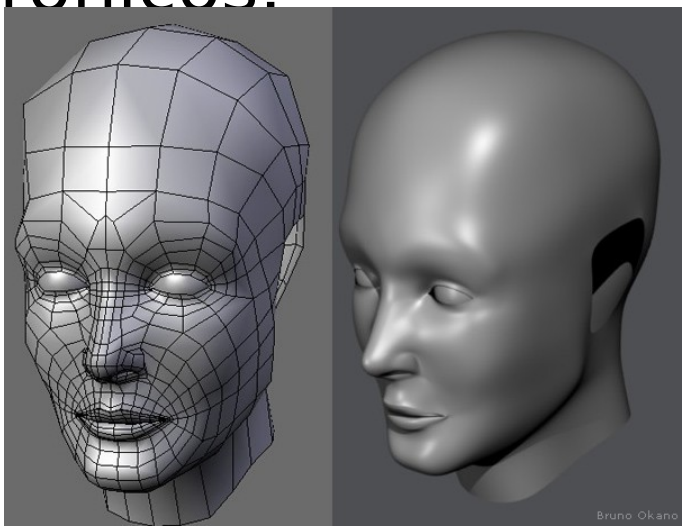
- **Problema:** Muitos desenvolvedores de software começam querendo construir prédios altos, como se estivessem fazendo uma casinha de cachorro (Booch et al).

Introdução


Como o cliente explicou...
Como o líder de projeto entendeu...
Como o analista projetou...
Como o programador construiu...
Como o Consultor de Negócios descreveu...
Como o projeto foi documentado...
Que funcionalidades foram instaladas...
Como o cliente foi cobrado...
Como foi mantido...
O que o cliente realmente queria...

Introdução

- **Modelagem –** largamente empregada nas mais diversas áreas do conhecimento como, por exemplo:
- Indústria de construção civil;
- Construções de aviões;
- Dispositivos elétricos e eletrônicos;
- Sistemas de telefonia;
- Indústria cinematográfica.



Introdução

- **O que é um modelo?**

  Um modelo é uma simplificação da realidade.

- **Por que fazer modelagem?**
  - Ajudar a visualizar o sistema como ele é ou como se deseja que ele seja;
  - Permite especificar a estrutura de um sistema;
  - Proporcionar um guia para a construção do sistema;
  - Documentar as decisões tomadas.

Introdução

- Os conceitos da orientação a objetos são bem difundidos, desde o lançamento da primeira linguagem orientada a objetos, a SIMULA;
- O desenvolvimento de sistemas de software de grande porte são suportados por métodos de análise e projeto que modelam esse sistema de modo a fornecer para toda a equipe envolvida uma compreensão única do projeto.

Introdução

- A UML (Unified Modeling Language) é o sucessor de um conjunto de métodos de análise e projeto orientados a objeto;
- A UML está, atualmente, em processo de padronização pela OMG (Object Management Group);
- UML foi desenvolvida por Grady Booch, James Rumbaugh, e Ivar Jacobson;
- A UML é a junção do que havia de melhor nestas três metodologias adicionado novos conceitos e visões da linguagem.

Fases de Desenvolvimento

- A UML suporta as cinco fases de desenvolvimento de software:
  - Análise de requisitos;
  - Análise;
  - Projeto;
  - Implementação;
  - Testes.
- Sendo que estas fases não necessariamente precisam ser executadas em seqüência.

Fases de Desenvolvimento

- **Análise de Requisitos**: Nesta fase são obtidas as necessidades dos usuários e o comportamento do sistema através de análise de **casos de uso**;
- As entidades externas, denominadas “atores externos”, que interagem com o sistema são modeladas em conjunto com as funções que elas requerem;
- O diagrama de casos de uso é usado para identificar como o sistema se comporta em várias situações que podem ocorrer durante sua operação.

Fases de Desenvolvimento

- **Análise**: Nesta fase são identificadas as classes, objetos e os mecanismos que estarão presentes no domínio do problema;
- As classes são modeladas e interligadas através de relacionamentos utilizando o **diagrama de classes**;
- Na análise só serão modeladas classes que pertençam ao domínio do problema;
- Classes auxiliares que gerenciem banco de dados, comunicação, interface e outros não estarão presentes neste tipo de diagrama.

Fases de Desenvolvimento

- **Projeto**: Os resultado da análise é expandido nesta fase em termos de soluções técnicas;
- Novas classes serão adicionadas para prover uma infra-estrutura técnica: a interface com o usuário e periféricos, gerenciamento de banco de dados e comunicação com outros sistemas entre outros.
- As classes do domínio do problema, modeladas na fase de análise, são mescladas nessa nova infra-estrutura tornando possível alterar tanto o domínio do problema quanto a infra-estrutura.

Fases de Desenvolvimento

- **Implementação**: As classes são convertidas para código real em uma linguagem orientada a objetos como, por exemplo, Java, C++ e C# entre outras;
- Dependendo da capacidade da linguagem usada o grau de dificuldade na conversão poderá ser variado.

Fases de Desenvolvimento

- **Testes**: É idêntico a qualquer outro método de modelagem;
- Podendo ser sub-dividido em testes de unidades, testes de integração, teste de sistema e testes de aceitação.

Notação

- Em cada uma das fases do desenvolvimento podem ser utilizadas cinco tipos de visões, nove tipos de diagramas e vários modelos de elementos;
- Todos em conjunto especificam e exemplificam a definição do sistema, tanto no que diz respeito à funcionalidade estática e dinâmica do desenvolvimento de um sistema;
- Antes de estudar os principais componentes separadamente, serão definidas as partes que compõem a UML: visões, modelos, mecanismos e diagramas.

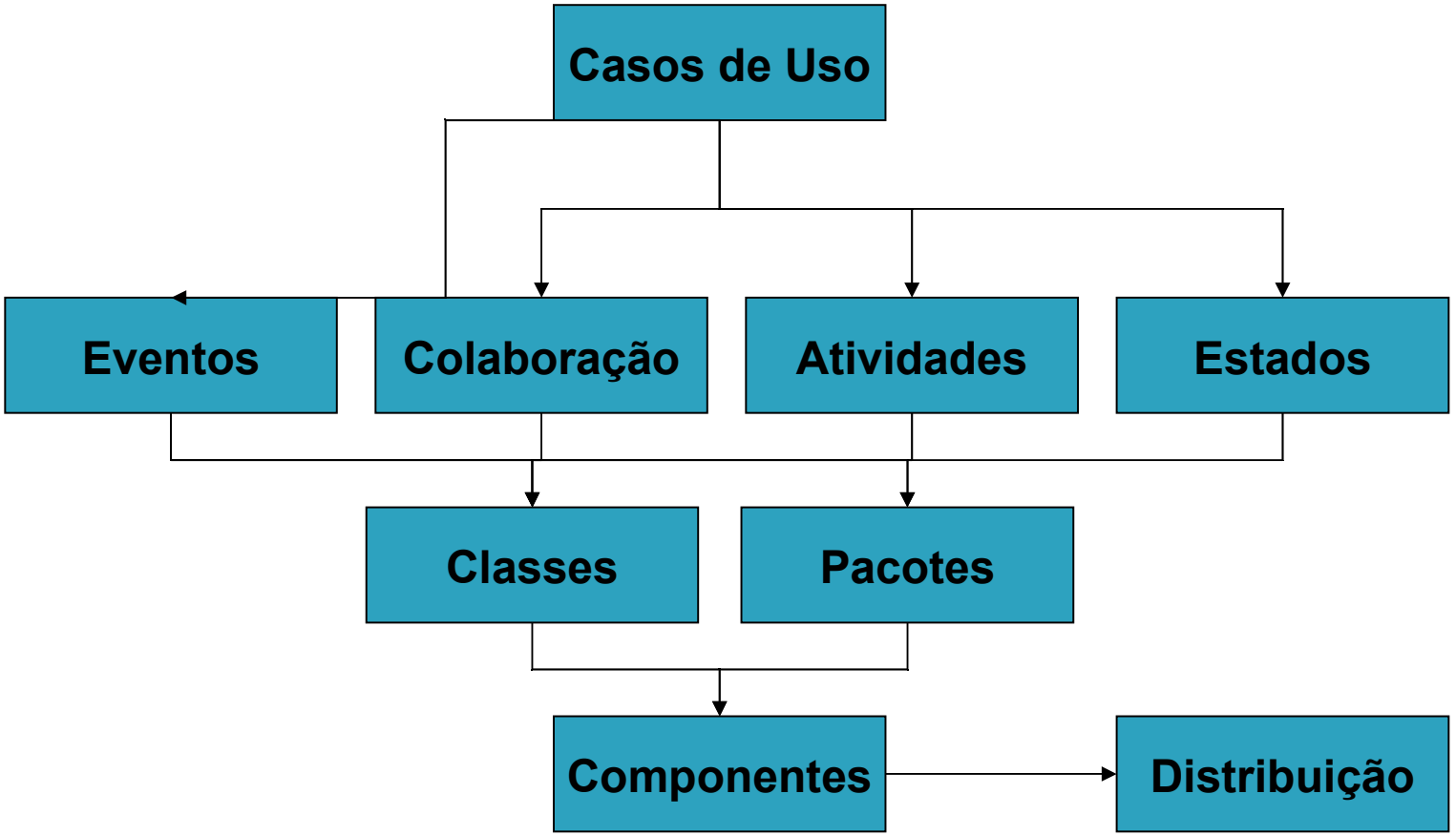
Notação

Comportamento Externo

Comportamento Interno

Estrutura

Implementação



Notação

- **Visões**: Mostram diferentes aspectos do sistema que está sendo modelado. A visão não é um gráfico, mas uma abstração consistindo em uma série de diagramas.
- **Modelos de Elementos**: Os conceitos usados nos diagramas são modelos de elementos que representam definições comuns da orientação a objetos como as classes, objetos, mensagem, relacionamentos entre classes incluindo associações, dependências e heranças.

Notação

- **Mecanismos Gerais**: Provém comentários suplementares, informações, ou semântica sobre os elementos que compõem os modelos.
- **Diagramas**: Os diagramas são os gráficos que descrevem o conteúdo em uma visão. A UML possui nove tipo de diagramas que são usados em combinação para prover todas as visões do sistema.

Visões

- O desenvolvimento de sistemas complexos envolvem vários aspectos, tais como: o aspecto funcional, não funcional e aspectos organizacionais;
- Para isto utiliza-se o conceito de visões, onde cada visão é descrita por um número de diagramas que contém informações que dão ênfase aos aspectos particulares do sistema;
- Os diagramas que compõem as visões contém os modelos dos elementos do sistema;
- As visões que compõem um sistema podem ser enumeradas em: casos de uso, lógica, componentes, concorrência e organização.

Visões

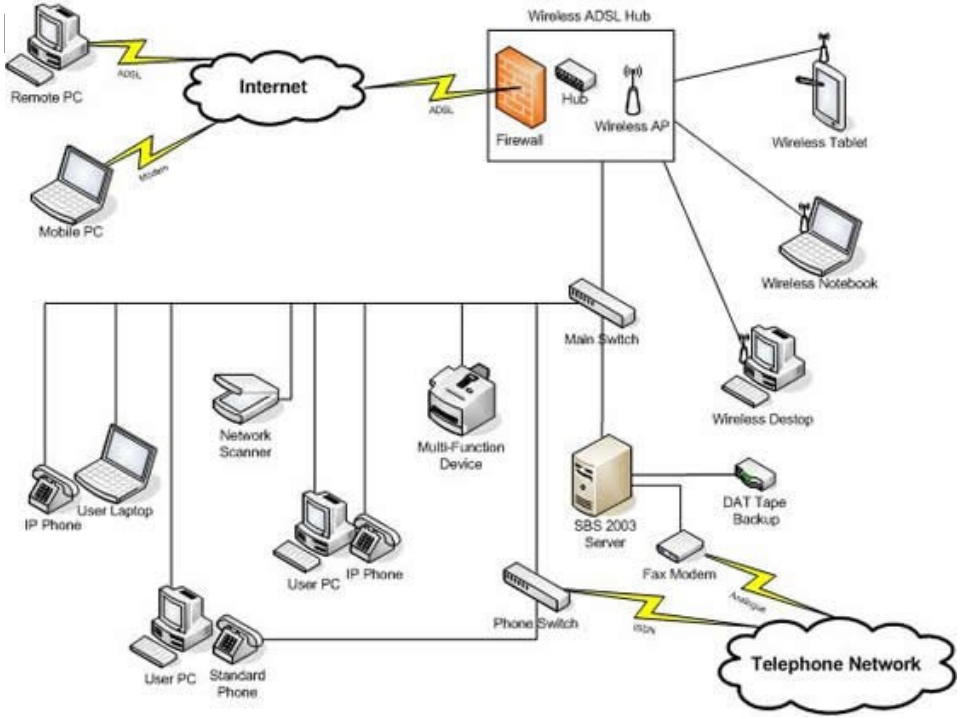
- **Casos de Uso**: Descrevem a funcionalidade do sistema desempenhada pelos atores externos ao sistema como, por exemplo, usuários;
- **Visão Lógica**: Descreve como a funcionalidade do sistema será implementada. É feita principalmente pelos analistas e desenvolvedores. Em contraste com a visão de casos de uso, a visão lógica observa e estuda o sistema internamente. Ela descreve e especifica a estrutura estática do sistema (classes, objetos, e relacionamentos) e as colaborações dinâmicas quando os objetos enviarem mensagens uns para os outros.

Visões

- **Visão de Componentes**: É uma descrição da implementação dos módulos e suas dependências. É principalmente executado por desenvolvedores, e consiste nos componentes dos diagramas;
- **Visão de Concorrência**: Trata a divisão do sistema em processos e processadores. Este aspecto, que é uma propriedade não funcional do sistema, permite uma melhor utilização do ambiente onde o sistema se encontrará, se o mesmo possui execuções paralelas, e se existe dentro do sistema um gerenciamento de eventos assíncronos.

Visões

- **<u>Visão de Organização</u>**: Mostra a organização física do sistema, os computadores, os periféricos e como eles se conecta



Análise de Requisitos

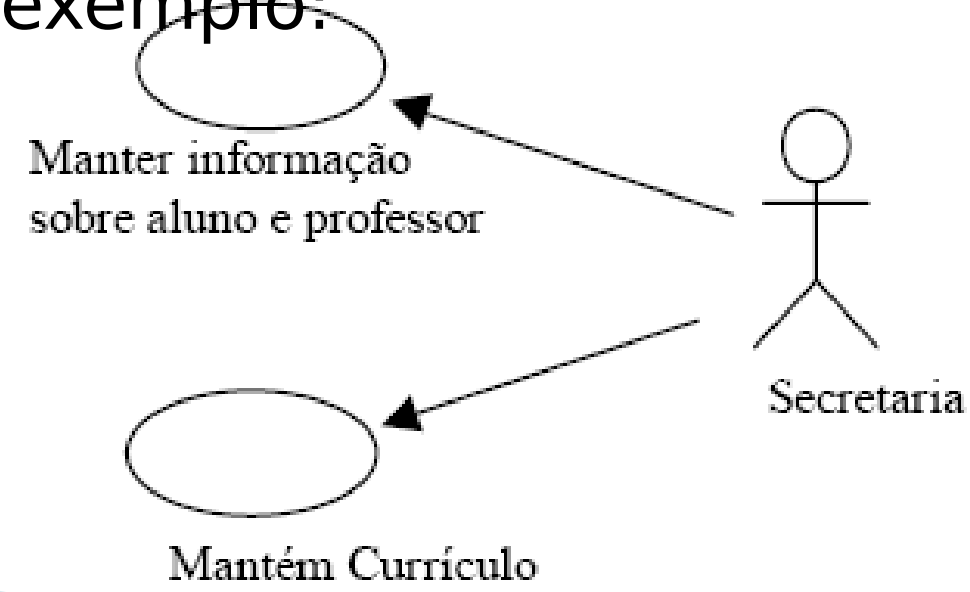
- Esta fase captura as intenções e necessidades dos usuários do sistema a ser desenvolvido através do uso de funções chamadas “casos de uso”;
- Basicamente pode ser entendido como a descrição das necessidades ou desejos de um determinado sistema.;
- O princípio básico da análise de requisitos é identificar e documentar o que é realmente necessário sintetizando de forma clara e acessível, a todos os envolvidos no projeto, o funcionamento desejado do sistema.

Casos de Uso

- É um instrumento utilizado para realizar a descrição das intenções ou requisitos para um sistema computacional;
- A construção do modelo de casos de uso corresponde a uma das fases iniciais de um projeto de software pois envolve a determinação dos usos que o sistema terá, ou seja, do que ele deverá fornecer como funcionalidades.

Casos de Uso

- O diagrama de casos de uso é usado para se identificar como o sistema se comporta nas várias situações que podem ocorrer durante sua operação. Os componentes deste diagrama são os atores e os próprios casos de uso, por exemplo:



Casos de Uso

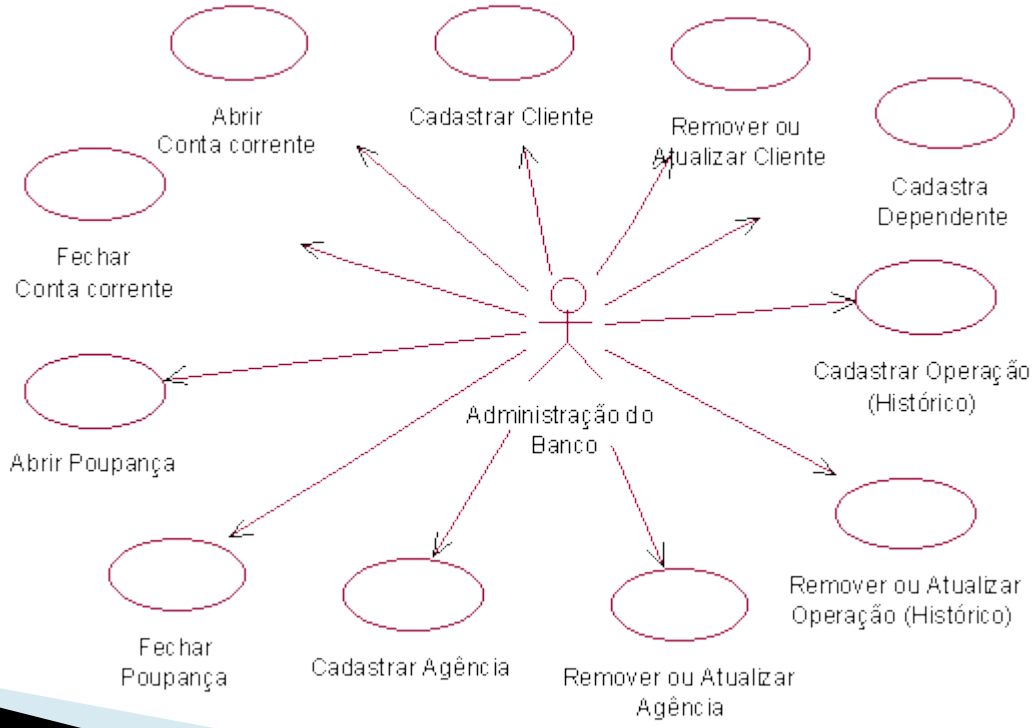
- A notação usada pelo diagrama de casos de uso consiste nos seguintes elementos gráficos:
- **Ator**: Representa qualquer entidade que interage com o sistema. Pode ser, por exemplo, uma pessoa, um dispositivo de hardware ou outro sistema;
- **Casos de Uso**: é uma seqüência de ações que o sistema executa e produz um resultado de valor para o ator.

Casos de Uso

- Neste momento não existe preocupação com a implementação de cada uma das funções descritas no caso de uso, pois este diagrama apenas se resume em determinar que funções deverão ser suportadas pelo sistema;

Casos de Uso

- A diagrama de casos de uso abaixo demonstra as funções de um ator externo em um sistema de controle bancário fictício:



Exercícios

- **1. Identificar os requisitos funcionais para o seguinte problema:**

  Um programa deverá receber, através de digitação, dois números inteiros entre 1 e 100. Quando o primeiro número digitado for maior ou igual o segundo número o programa deverá realizar a soma dos mesmos exibindo o resultado no monitor de vídeo, caso contrário, deverá ser mostrada uma mensagem de erro.

Exercícios

- **2. Elaborar o diagrama de casos de uso para a seguinte situação:**

  Uma vídeo locadora pretende informatizar suas operações. É sabido que os clientes podem realizar a reserva, locação e devolução de filmes, além de efetuar os pagamentos em dinheiro e cartão de crédito. Por outro lado, os atendentes dessa vídeo locadora são responsáveis pela manutenção do cadastro do cliente e também informá-lo

  sobre a ocorrência de atrasos na

  entrega dos filmes locados.

Exercícios

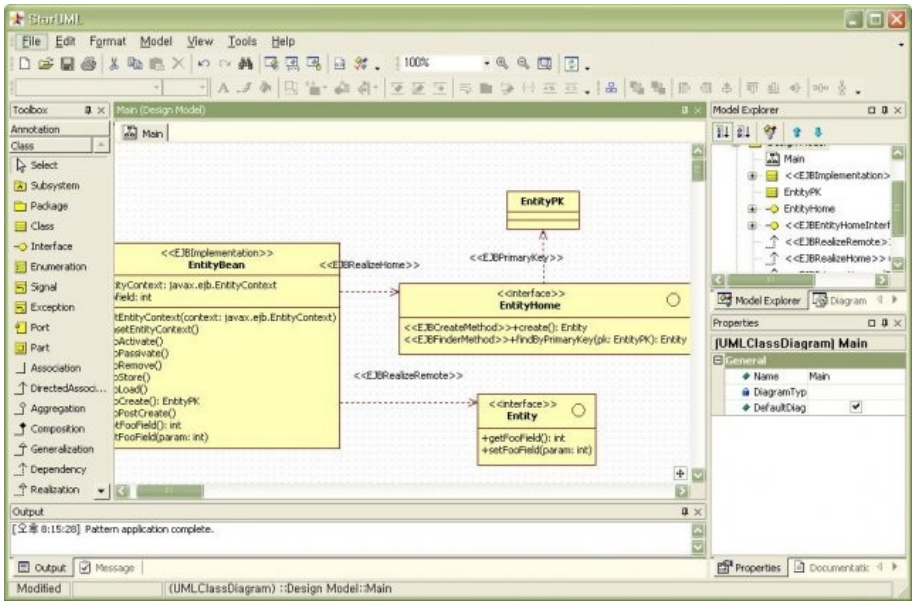
- **3. Elaborar o diagrama de casos de uso para a seguinte situação:**

  Uma nota fiscal é emitida sempre que uma determinada compra é finalizada e os dados referentes ao pagamento são aceitos pelo setor financeiro. Ao realizar a impressão da nota fiscal deve-se verificar se os produtos constantes na mesma estão disponíveis no estoque e também que os dados do cliente estão todos preenchidos no respectivo cadastro.

StarUML

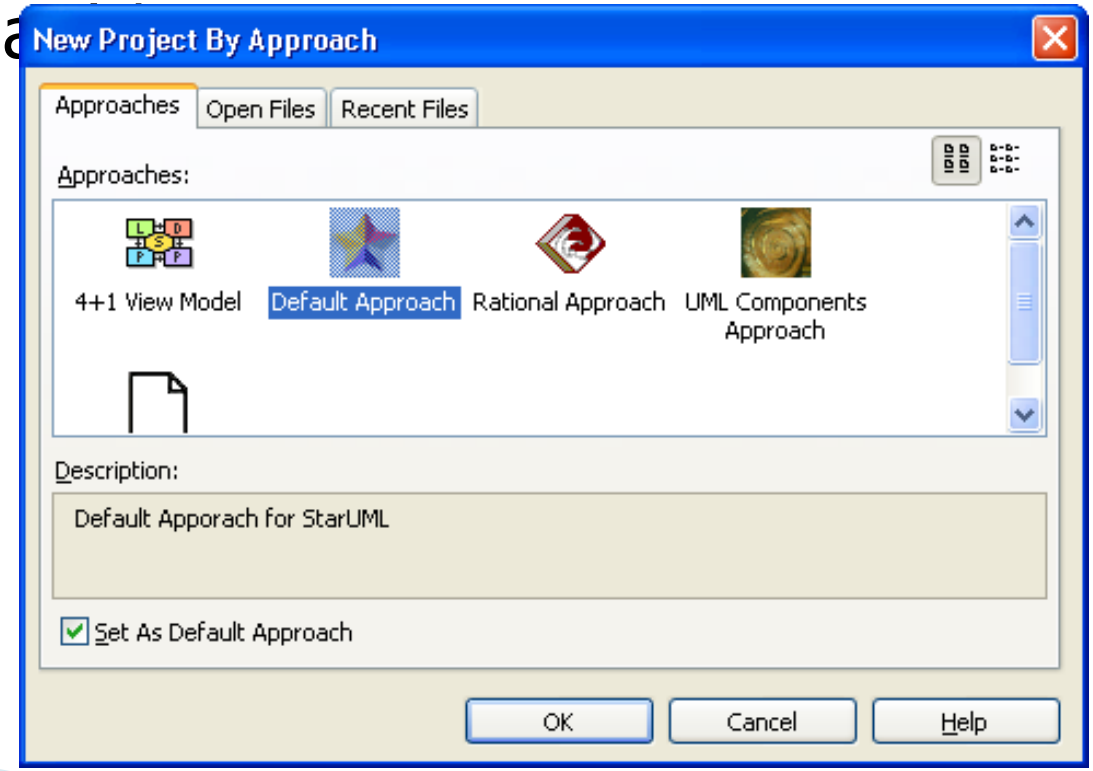
- StarUML é uma ferramenta gratuita e bastante completa para realizar modelagem. Pode ser obtida através do endereço: **http://www.staruml.com**

StarUML™
The Open Source UML/MDA Platform

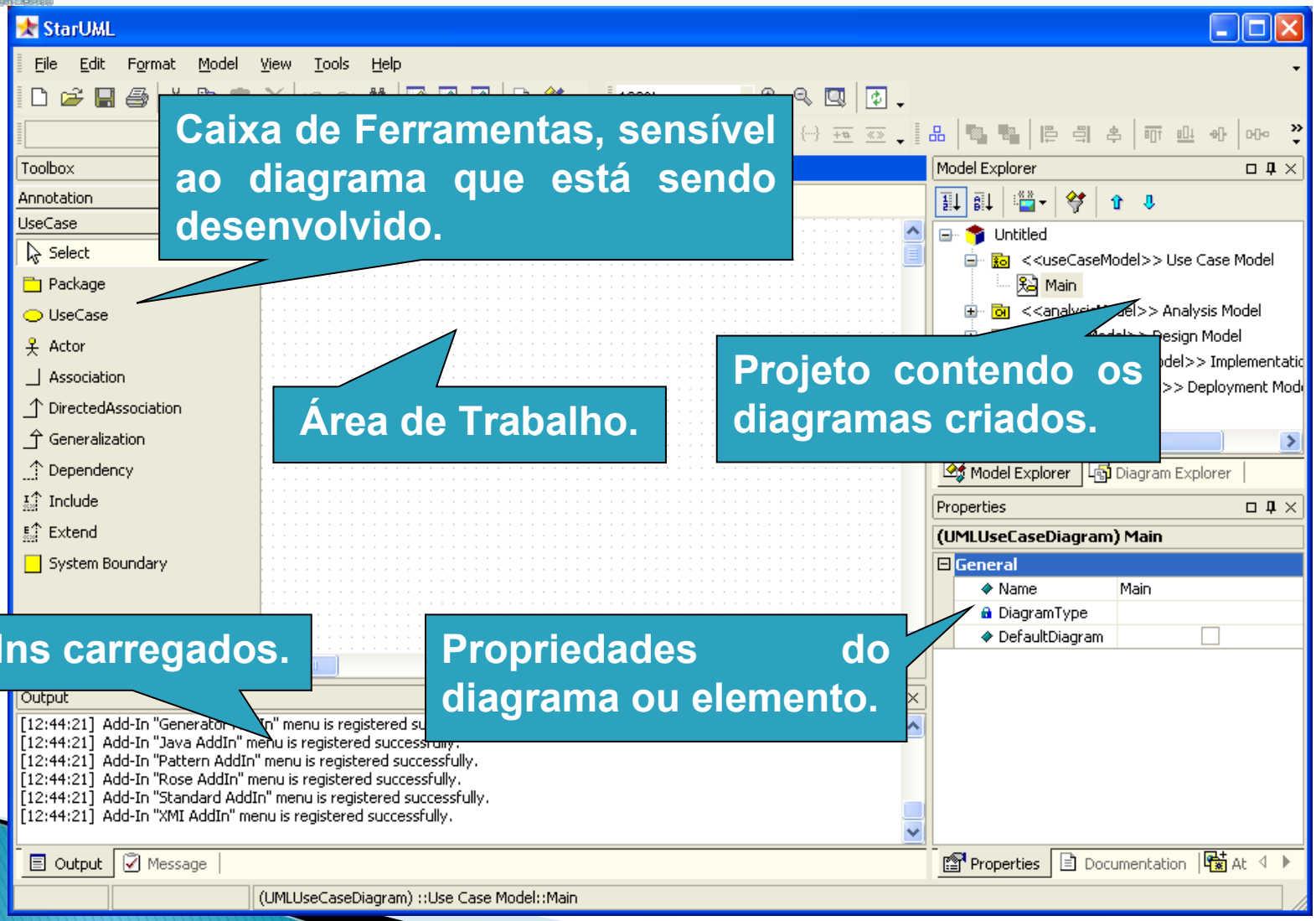


StarUML

- Após carregar o programa é necessário escolher o tipo de abordagem, no nosso caso sempre será utilizada a padrão (Default Approa

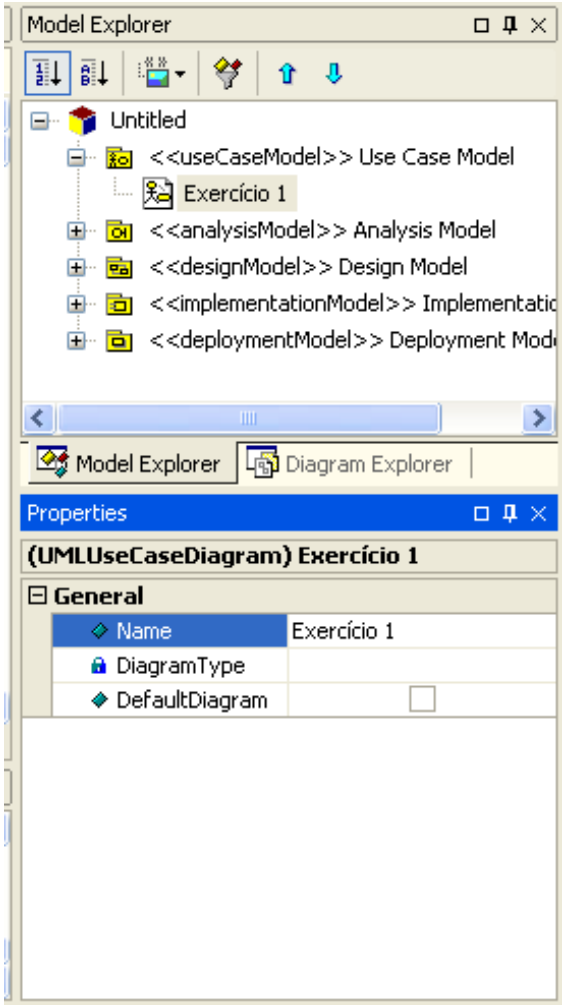


StarUML


StarUML
File Edit Format Model View Tools Help
Caixa de Ferramentas, sensível ao diagrama que está sendo desenvolvido.
Toolbox
Annotation
UseCase
Select
Package
UseCase
Actor
Association
DirectedAssociation
Generalization
Dependency
Include
Extend
System Boundary
Área de Trabalho.
Projeto contendo os diagramas criados.
Model Explorer
Untitled
<<useCaseModel>> Use Case Model
Main
Model Explorer
Diagram Explorer
Properties
(UMLUseCaseDiagram) Main
General
Name Main
DiagramType
DefaultDiagram
Add-Ins carregados.
Propriedades do diagrama ou elemento.
Output
[12:44:21] Add-In "Java AddIn" menu is registered successfully.
[12:44:21] Add-In "Pattern AddIn" menu is registered successfully.
[12:44:21] Add-In "Rose AddIn" menu is registered successfully.
[12:44:21] Add-In "Standard AddIn" menu is registered successfully.
[12:44:21] Add-In "XMI AddIn" menu is registered successfully.
Output
Message
Properties
Documentation
(UMLUseCaseDiagram) ::Use Case Model::Main

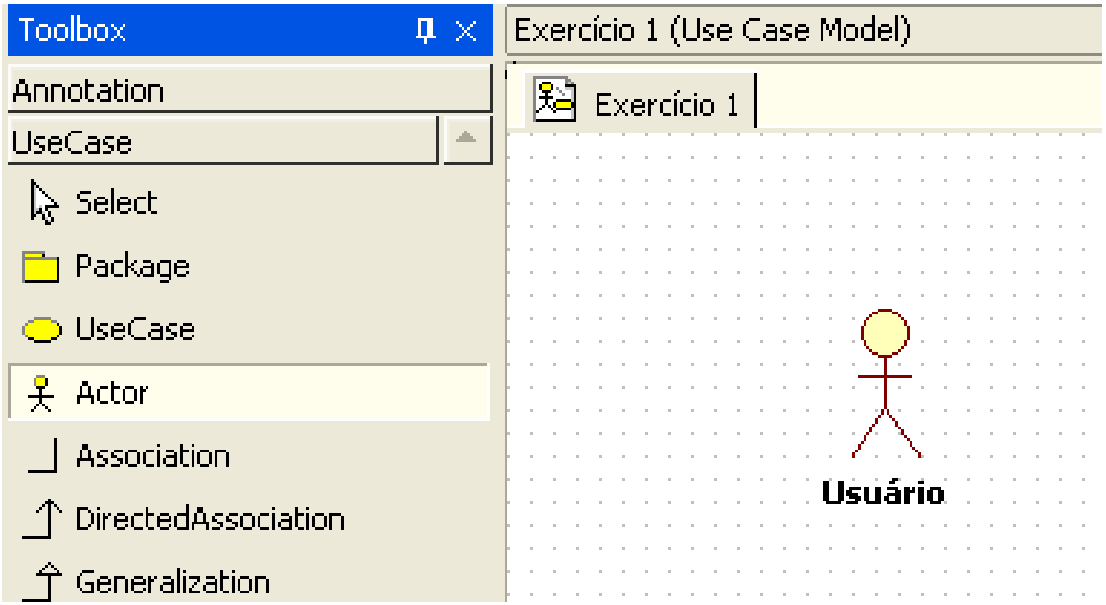
Criando Casos de Uso

- No primeiro exercícios foram identificados os requisitos do cenário que foi apresentado, a partir deles criaremos um caso de uso dentro da ferramenta StarUML;
- O primeiro passo consiste em dar um nome ao diagrama, por exemplo, “Exercício 1”.



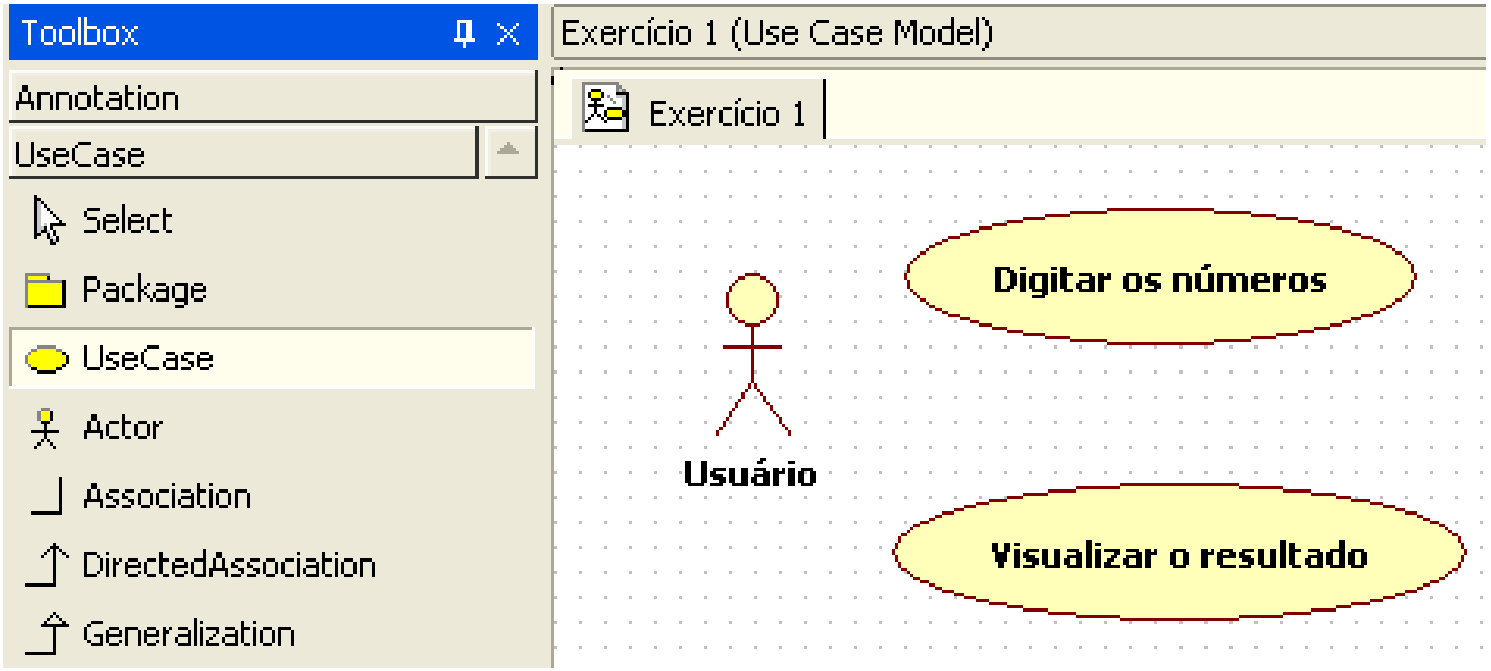
Criando Casos de Uso

- Agora será criado um ator chamado “Usuário”, o qual será responsável pela operação do sistema. Para isso basta clicar no respectivo ícone na caixa de ferramentas e depois clicar na área de trabalho:



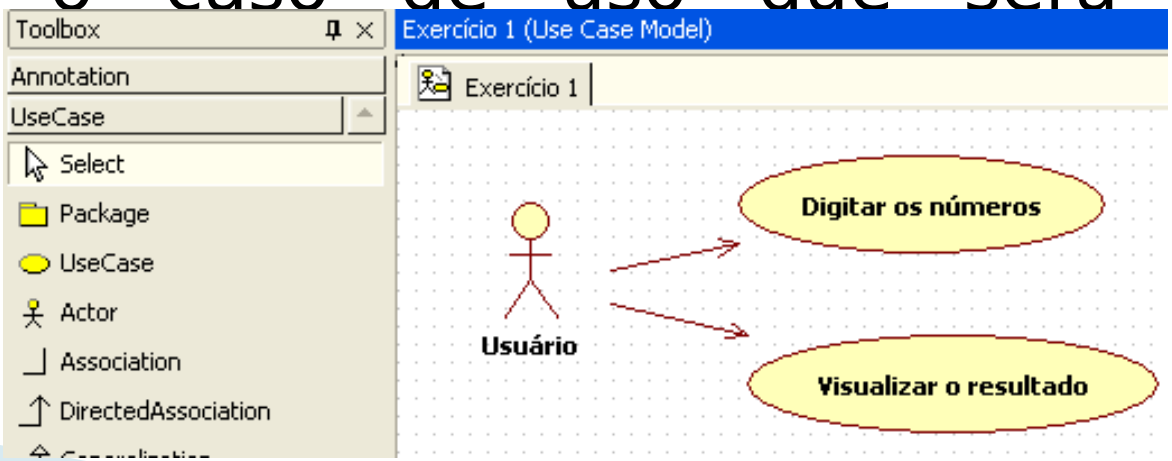
Criando Casos de Uso

- A inserção dos casos de uso deve ser realizada também através da caixa de ferramentas, por exemplo:



Criando Casos de Uso

- A última etapa do processo de criação consiste em realizar a associação entre os atores e respectivos casos de uso.
- Neste caso deve-se escolher o ícone “Directed Association” na caixa de ferramentas, clicar sobre o ator e depois arrastar até o caso de uso que será associado:



Exercícios

- 4. Usando a StarUML finalizar o diagrama de casos de uso referente ao exercício 1.
- 5. Elaborar, usando a ferramenta StarUML, o diagrama de casos de uso para os exercícios 2 e 3.

Orientação a Objetos

Diagrama de Classes

- Considerado um dos mais importantes diagramas da UML, permite demonstrar a estrutura estática das classes de um sistema além dos relacionamentos entre as diversas classes. Também possibilitam ilustrar os atributos e operações de uma determinada classe.

Diagrama de Classes

- Exemplo de um diagrama de classes para uma locadora de veículos fictícia:

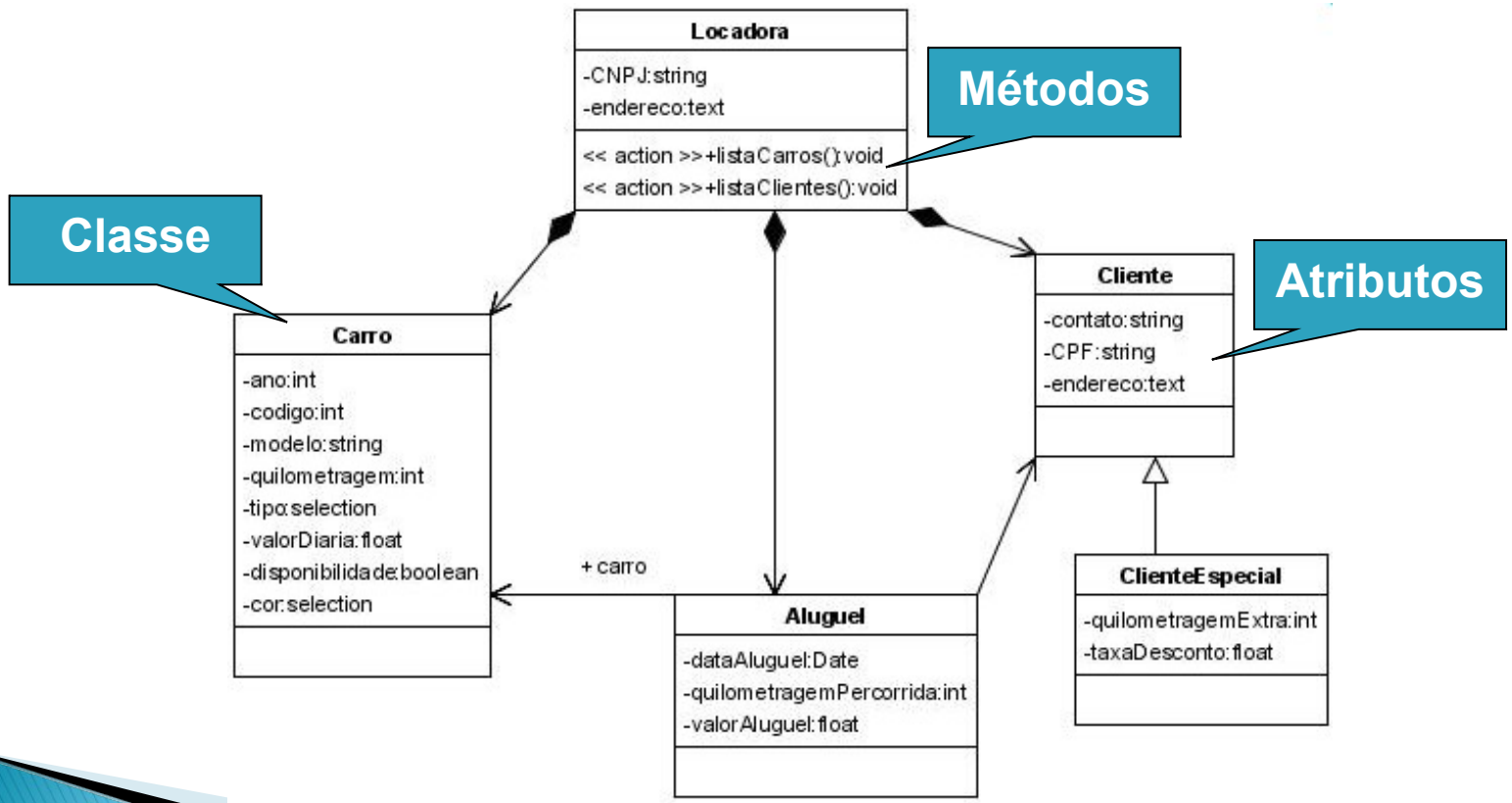


Diagrama de Classes

- Com o intuito de ilustrar o desenvolvimento de um **diagrama de classes** dentro da StarUML, as páginas seguintes irão demonstrar a modelagem do caso de uso de um cliente realizando a locação de um filme dentro de um sistema de vídeo-locadora.

Diagrama de Classes

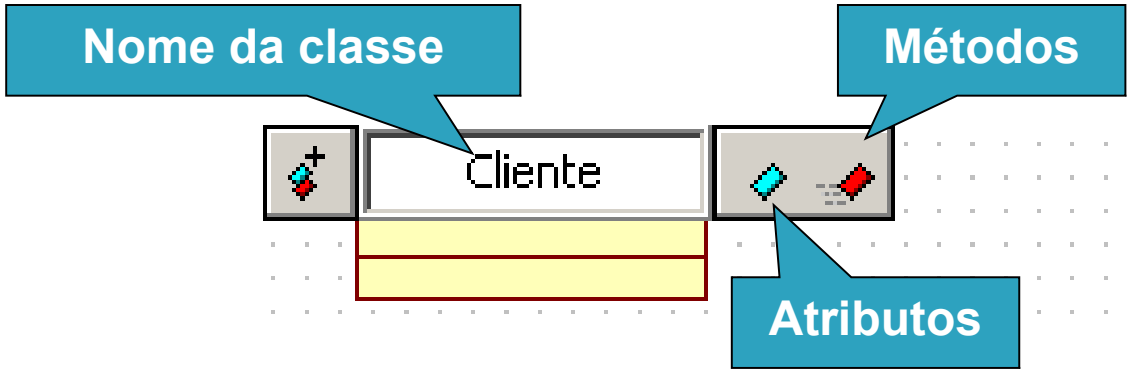
- Considerando o cenário descrito no exercício 2 e as informações adicionais mostradas abaixo, será realizada modelagem o **diagrama de classes**.
  - Um filme deve possuir pelo menos um volume para locação. Para um volume deverá ser armazenado o seu código além de estar devidamente relacionado à um filme.
  - É necessário armazenar o código, título, gênero e duração do cada filme.

Diagrama de Classes

- Um cliente deve ter armazenado o seu código, nome, endereço e telefone. O cliente, após cadastrada, poderá realizar quantas locações desejar;
- Cada locação deve obrigatoriamente referenciar-se a pelo menos um filme e deve conter as datas de locação e devolução.

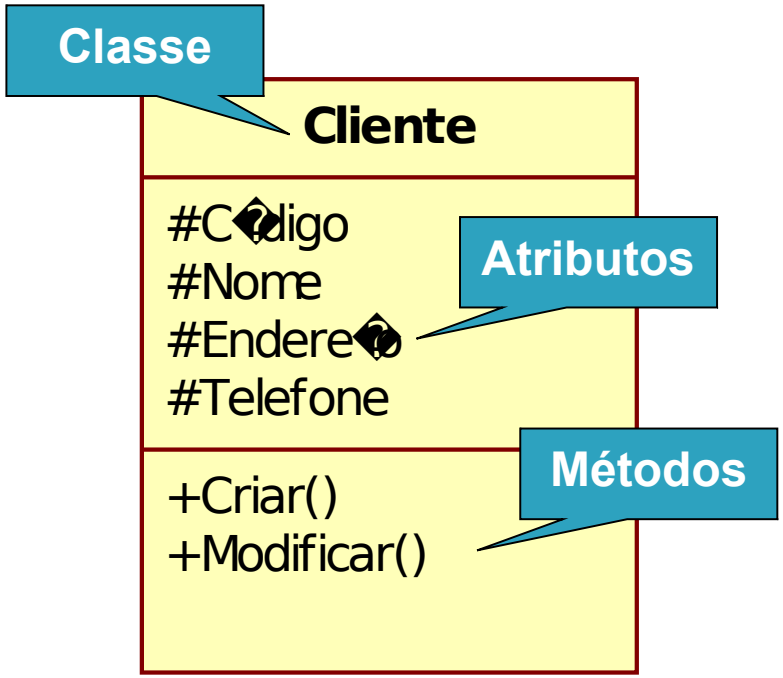
Classes

- A criação de classes na StarUML é realizada através do respectivo ícone dentro da barra de ferramentas;
- Em seguida deve-se definir o nome da classe e os respectivos atributos e métodos.



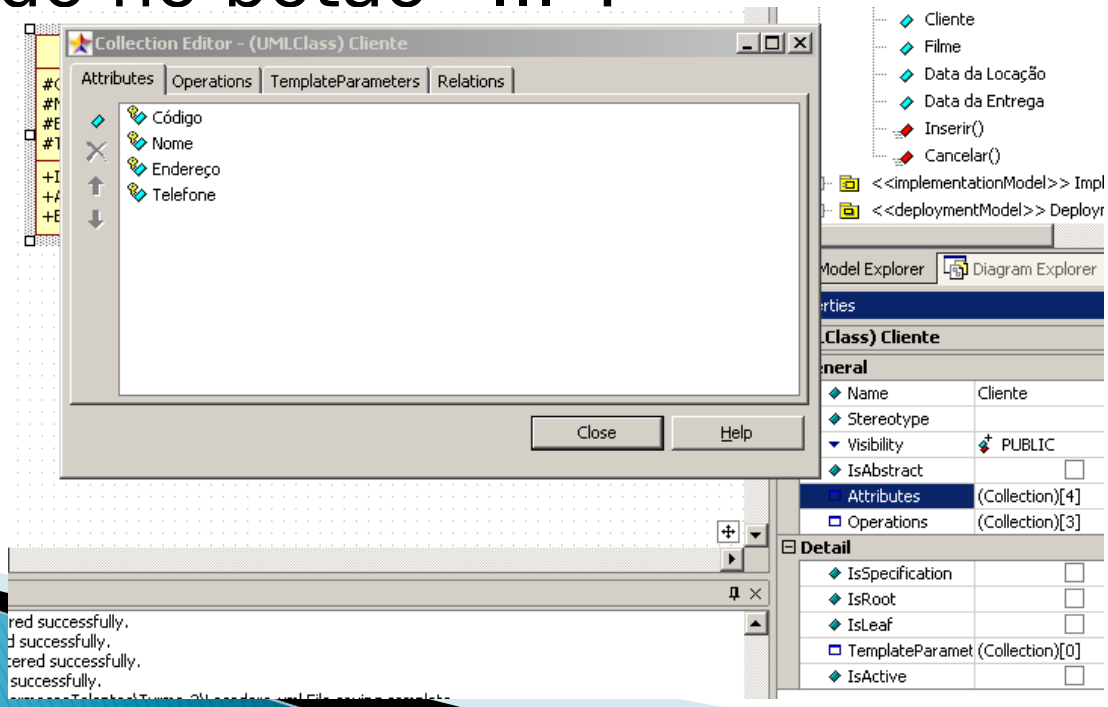
Classes

- Ao final do processo será possível visualizar a classe com seus respectivos atributos e métodos, por exemplo:



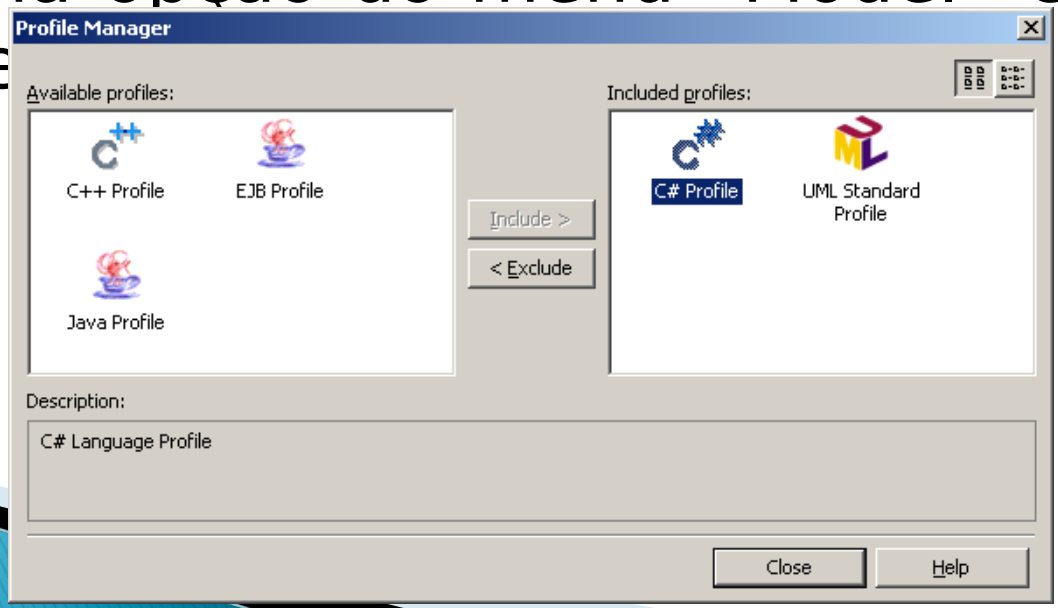
Classes

- A definição da visibilidade dos atributos e métodos da classe deverá ser realizada através da janela de propriedades, dentro das opções de “Attributes” e “Operations” e clicando no botão “...”:



Definição de Perfil

- Antes de iniciar a definição dos tipos de dados para atributos e métodos é necessário definir o “profile” em relação à linguagem de programação que será adotada na implementação. Para fazer isso, entre na opção do menu “Model” e escolha “Profile

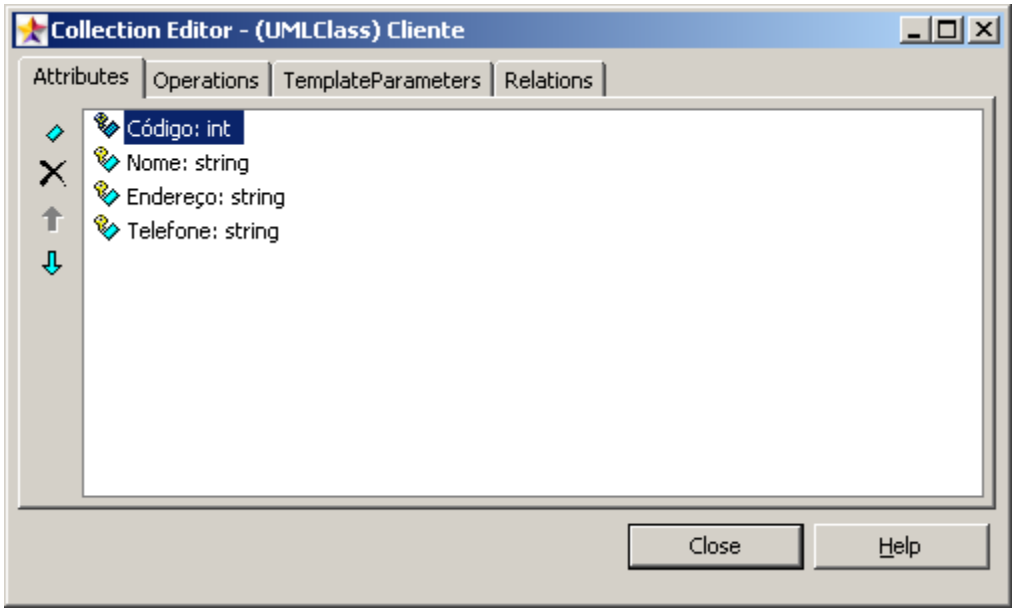


Classes

- É importante, principalmente quando se desejar fazer a criação automática de código-fonte a partir do diagrama de classes, realizar a definição dos tipos de dados de cada atributo e também dos parâmetros dos métodos;
- A definição será realizada através da janela de propriedades, escolhendo as opções “Attributes” ou “Operations” e depois clicando no botão “...”.

Classes

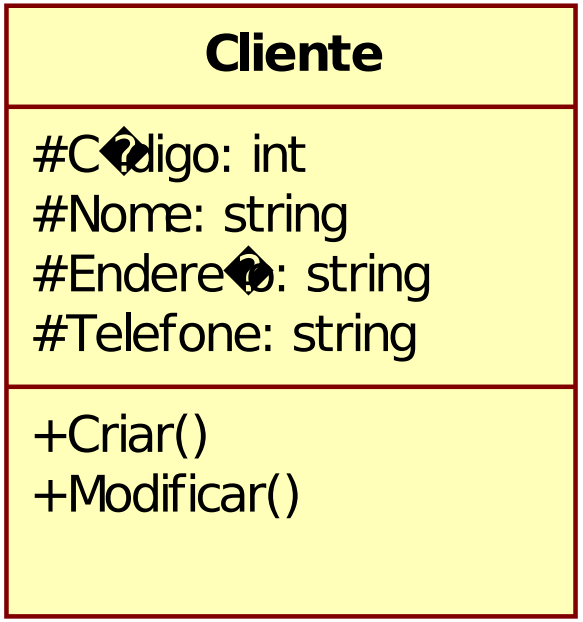
- Em seguida será exibida a janela do “Collection Editor”:



- Clique no atributo desejado e na janela de propriedades escolha na opção “Type” o tipo de dados desejado.

Classes

- Após a definição dos tipos de dados a classe deverá ser exibida da seguinte maneira:



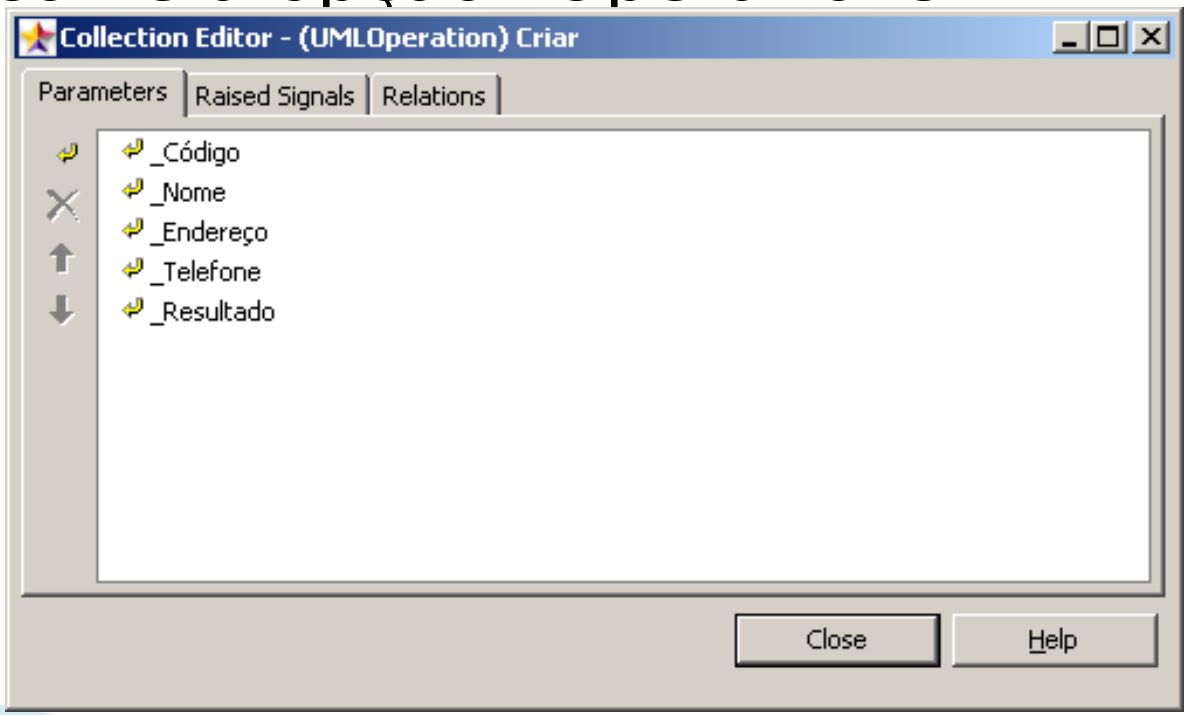
Classes

- A definição do métodos é ligeiramente diferente, pois os mesmos vão apresentar parâmetros de entrada e de saída, os quais precisarão ser especificados como no exemplo abaixo:

Cliente
#C�digo: int #Nome: string #Endere�o: string #Telefone: string
+Criar(_C�digo: int, _Nome: string, _Endere�o: string, _Telefone: string): bool +Modificar(_Nome: int, _Endere�o: string, _Telefone: string): bool

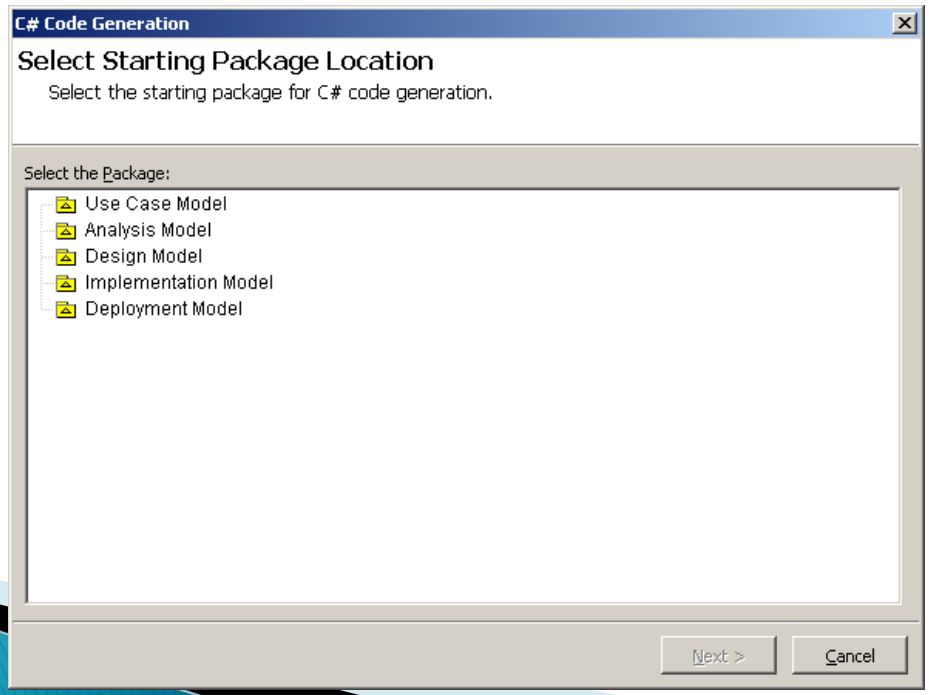
Classes

- Os parâmetros podem ser criados e alterados através da opção “Parameters” disponível na janela de propriedades, quando se escolhe a opção “Operations”:



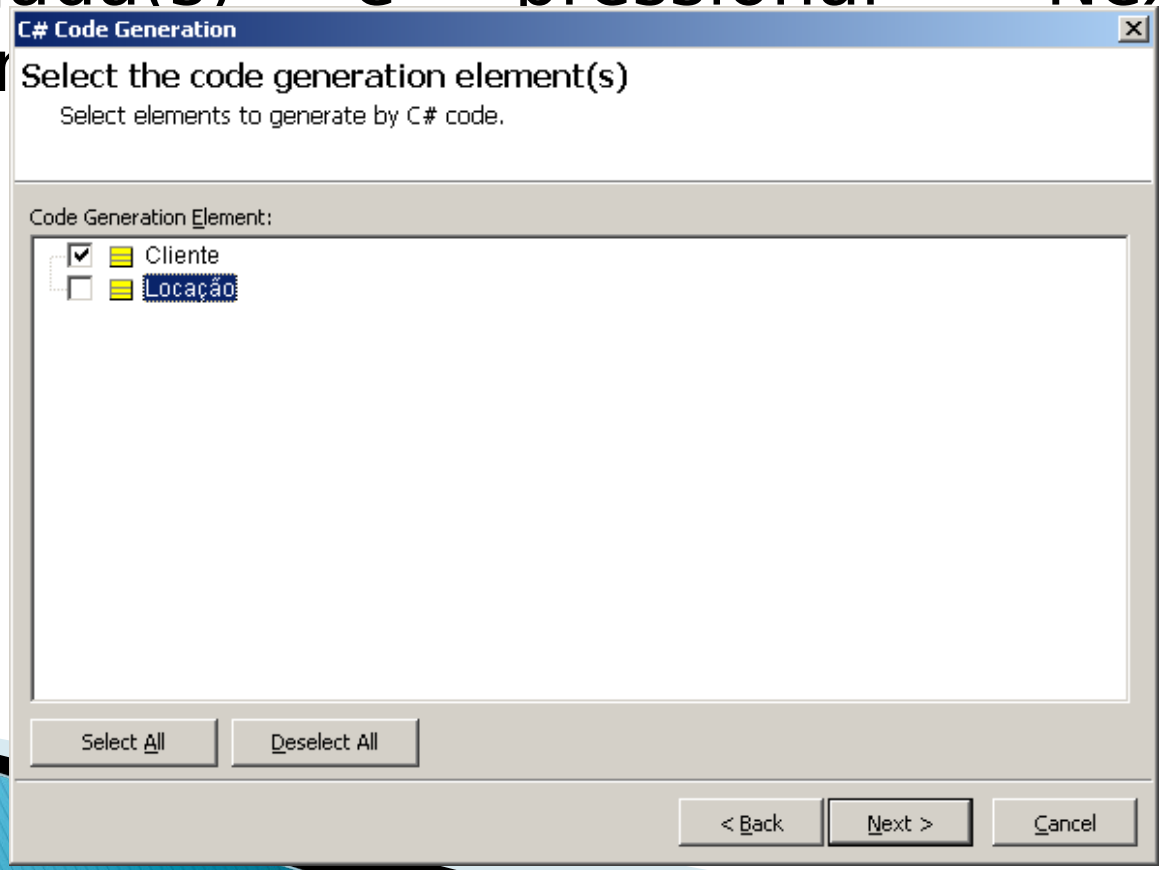
Geração de Código-fonte

- A StarUML permite a geração automática de código-fonte. Através do menu escolher a opção “Tools”, em seguida a linguagem desejada, neste exemplo será a C# e clicar em “Generate Code...”:



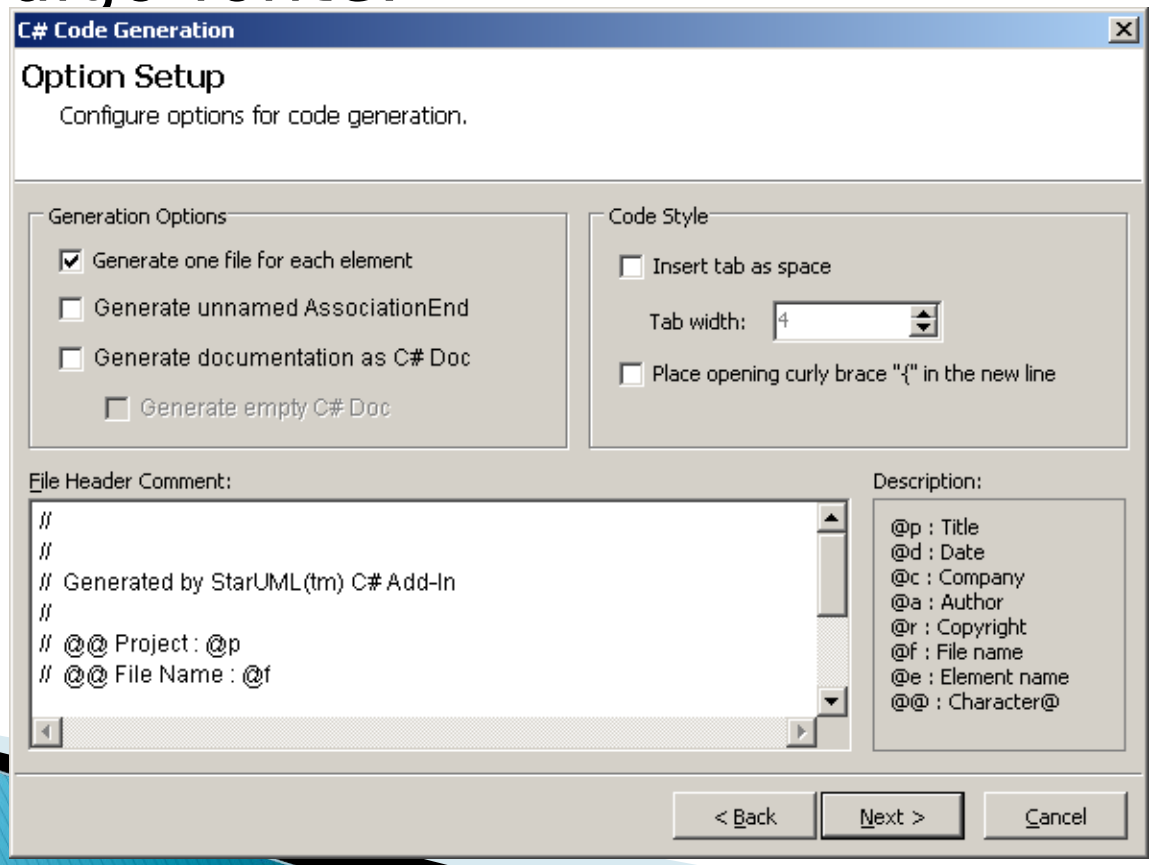
Geração de Código-fonte

- Selecionar “Design Model” e clicar no botão “Next >”, escolher a(s) classe(s) desejada(s) e pressionar “Next >” novamente



Geração de Código-fonte

- Escolher a pasta na qual serão criados os arquivos e depois a opções de formatação do código-fonte:



Exercícios

- 6. Adicionar um método chamado Consultar à classe Cliente. Este método não possui parâmetros de entrada e deverá apresentar como valor de retorno uma string;

- 7. Especificar as classes, com respectivos métodos e atributos, para os demais elementos do caso de uso da locação.

Associação

- **Associação:** pode ser entendida com um relacionamento que descreve uma série de ligações, onde a ligação é definida como a semântica entre as duplas de elementos que estão sendo ligados;
- **Associação Direta:** uma associação é considerada direta quando a mesma pode ser percorrida apenas em um único sentido;
- **Associação Recursiva:** permite conectar um elemento a ele mesmo através de uma associação e que, ainda assim, irá representar semanticamente a conexão entre dois objetos.

Associação

- Tipos de Associação:
  - **Agregação:** é um tipo de associação onde o objeto parte é um atributo do todo porém pode existir sem o mesmo. Por exemplo, em um carro, você pode tirar as rodas antes de destruí-lo e elas podem ser colocadas em outro carro.
  - **Composição:** descreve o relacionamento entre um elemento (o todo) e outros elementos (as partes), onde as partes só podem pertencer ao todo e são criadas e destruídas com ele. Por exemplo, uma Nota Fiscal é composta por itens.

Generalização

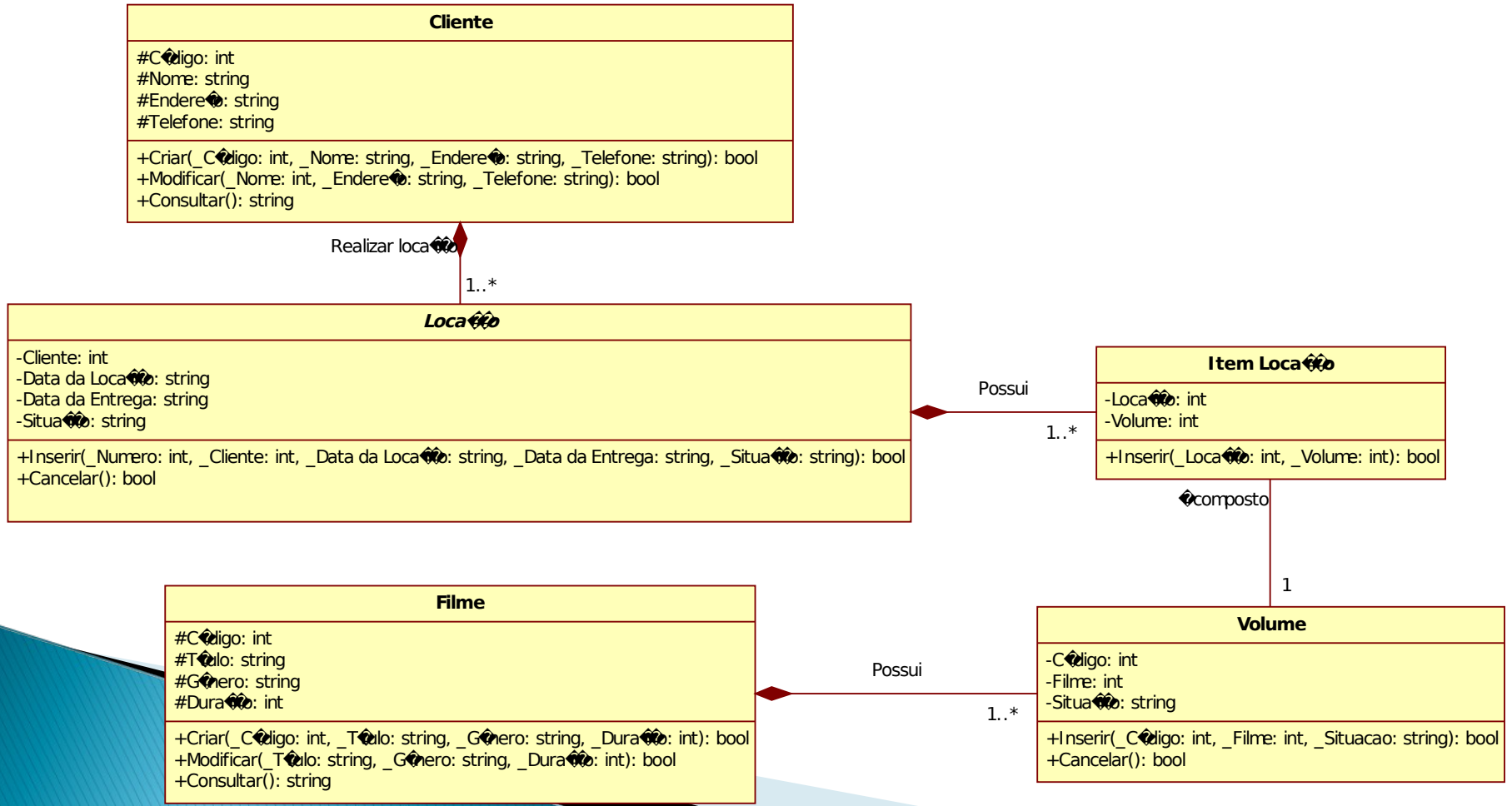
- A **generalização** ou **herança** é um relacionamento entre um elemento geral e um outro mais específico. O elemento mais específico possui todas as características do elemento geral e contém ainda mais particularidades.

Relacionamentos

- As classes, por sua vez, podem se relacionar com outras através de diversas maneiras:
  - **Associação:** classes conectadas entre si;
  - **Dependência:** quando uma classe depende ou necessita de outra classe;
  - **Especialização:** nas situações em que uma classe mais específica é detalhada a partir de classes genéricas;
  - **Pacotes:** classes que se encontram agrupadas porque apresentam características similares.

Relacionamentos

- Diagrama de classes completo para o caso de uso locação:



Exercícios

- 8. Criar o **diagrama de caso de uso** e o **diagrama de classes** para um sistema de clínica veterinária, levando em consideração as seguintes informações:
  - Um cliente pode possuir muitos animais, mas um animal pertence exclusivamente à um cliente. A clínica precisa de informações a respeito de cada cliente, como nome, endereço e telefone além de saber os animais que o mesmo possui.
  - Um animal pertence sempre a uma única espécie;

Exercícios

- É necessário manter informações sobre cada animal, como nome, sexo, idade e a espécie à qual pertence;
- Um animal pode receber muitos tratamentos, mas um tratamento sempre é realizado exclusivamente para um animal;
- Cada tratamento deve possuir ao menos uma consulta. Uma consulta é exclusiva de um tratamento. Cada consulta deve armazenar informações sobre a data que foi realizada, o veterinário responsável e um resumo sobre a consulta.

Exercícios

- Um médico veterinário pode realizar muitas consultas, porém uma consulta deve ser realizada sempre por um único veterinário;
- Em uma consulta pode ocorrer a necessidade de serem marcados exames para o animal, o número de exames possíveis em uma consulta é ilimitado, mas todos devem registrar o nome do exame, a data, o veterinário, o motivo da solicitação do exame e um resumo sobre o resultado.

Diagrama de Seqüência

- O diagrama de seqüência permite modelar os processos através da troca de mensagens (eventos) entre os objetos que compõem o sistema;
- Neste diagrama os objetos são representados por linhas verticais e as mensagens como setas que partem do objeto que invoca um outro objeto;
- As setas pode ser cheias para indicar uma mensagem de chamada ou tracejadas para indicar uma mensagem de retorno;
- Devem ser desenhados tantos diagramas de seqüência quantos cenários foram levantados no diagrama de casos de uso.

Diagrama de Seqüência

- Na ferramenta StarUML clicar com o botão da direita em Design Model, escolher a opção “Add Diagram” e depois “Sequence Diagram”;
- Através dos ícones disponíveis na caixa de ferramentas desenhar os ob
os estímulos (métodos).

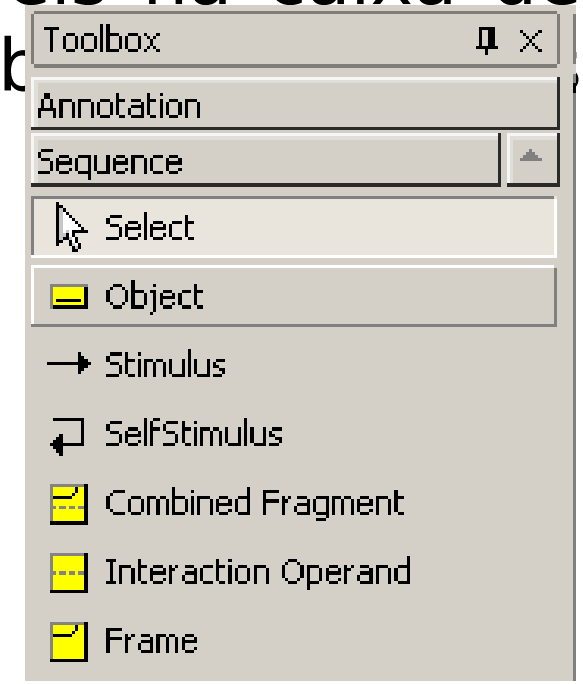


Diagrama de Seqüência

- Abaixo um exemplo de diagrama de seqüência criado na StarUML para um processo de acender e apagar uma lâ a:

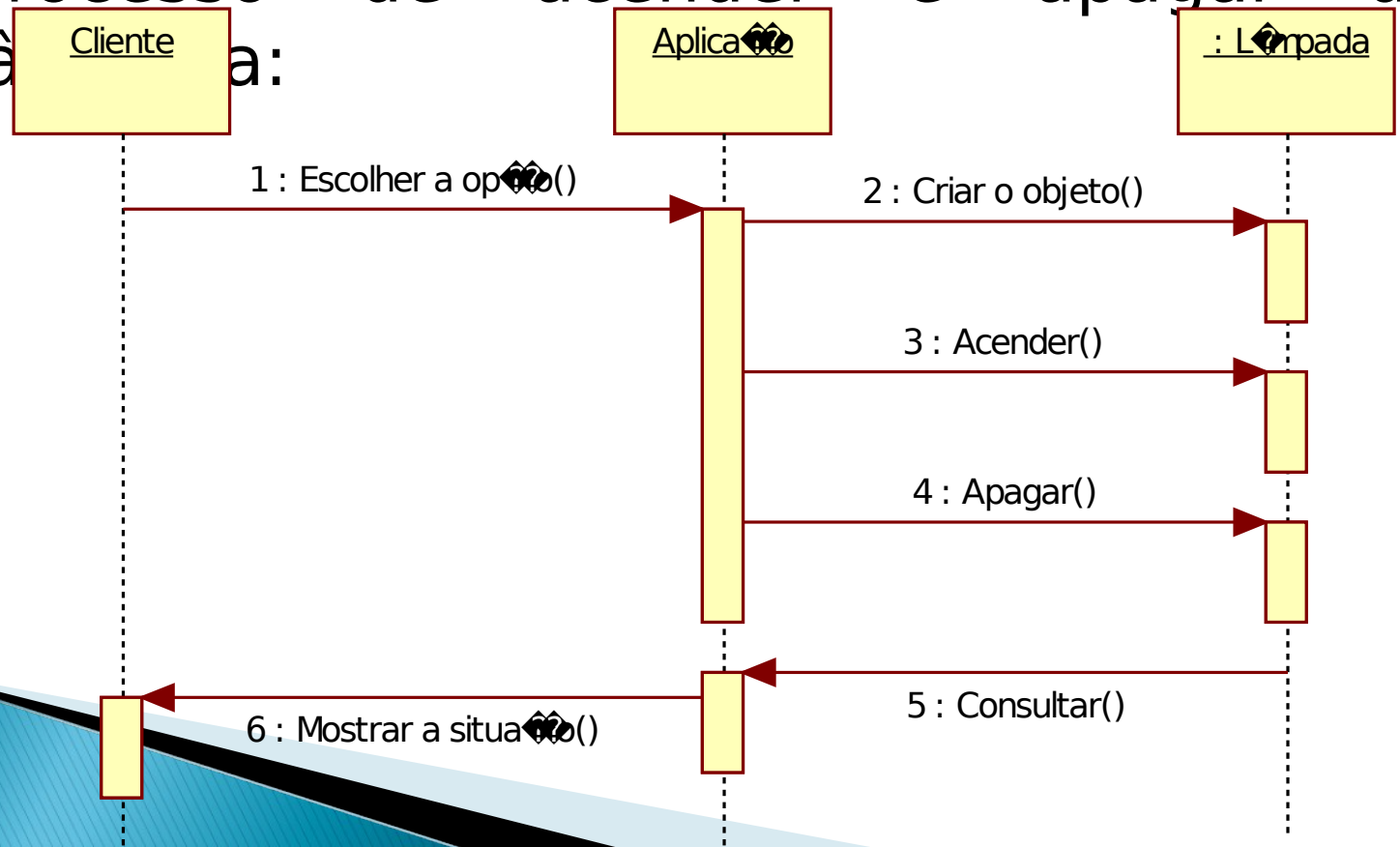


Diagrama de Estados

- Os diagramas de transição de estados mostra a dinâmica interna de uma classe, sendo que apenas os eventos e estados de uma única classe são apresentados neste diagrama;
- Podemos entender como eventos os fatos que ocorrem em uma classe, provocados por elementos externos (mensagens) ou internos como condições internas da classe e que provocam uma troca de estado.

Diagrama de Estados

- Uma classe pode ter vários estados, caracterizados por situações em que a classe se encontra. O diagrama de estados podem possuir ainda estados especiais como o estado inicial e o estado final e outros estados necessários ao controle interno.

Diagrama de Estados

- Na ferramenta StarUML clicar com o botão da direita em Design Model, escolher a opção “Add Diagram” e depois “Statechart Diagram”;
- Através da barra de ferramentas escolha os elementos que serão utilizados no diagrama de estado

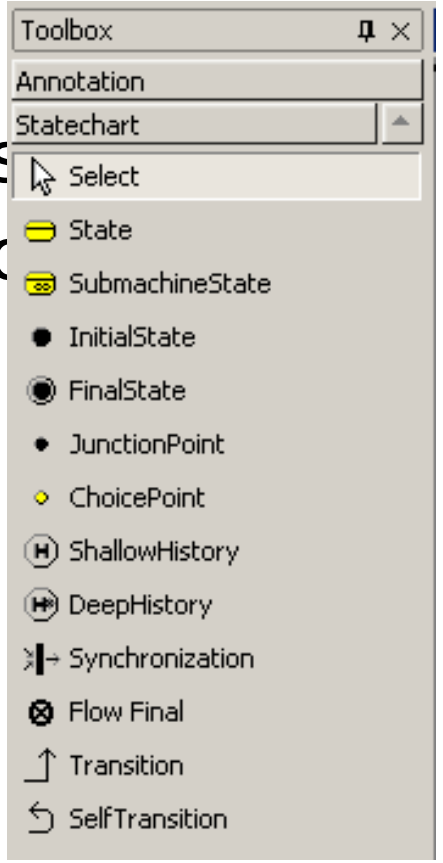


Diagrama de Estados

- O exemplo abaixo mostra o diagrama para um determinado objeto “Lâmpada”:

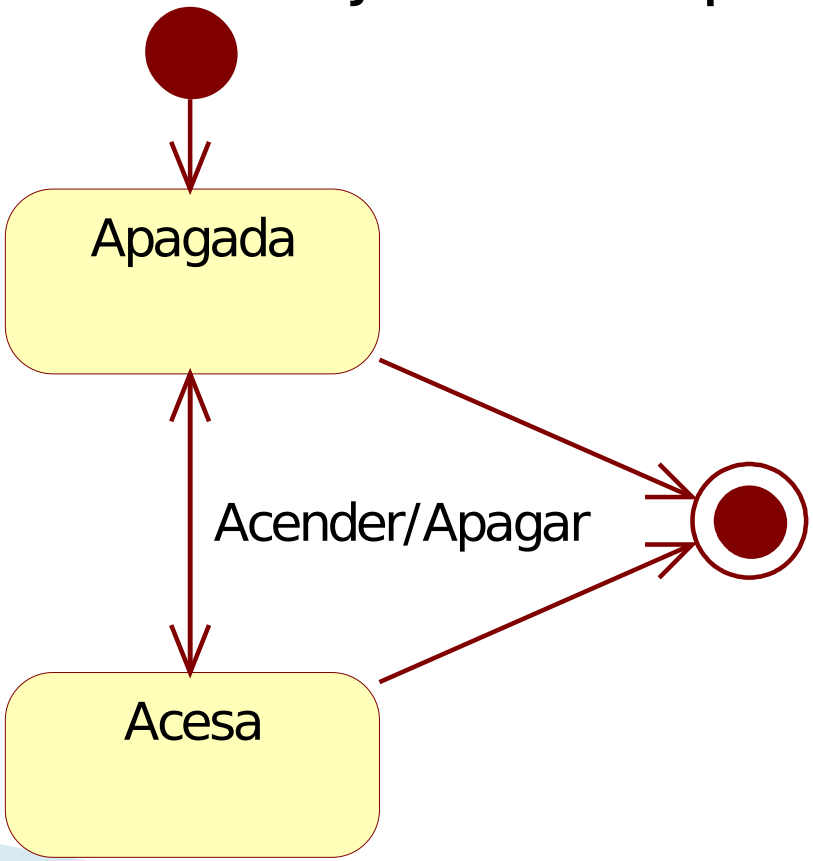


Diagrama de Componentes

- Os diagramas de componentes tem como finalidade mostrar os elementos reutilizáveis de uma aplicação e as suas interdependências;
- Um componente é composto por um conjunto de classes que, por sua vez, podem depender funcionalmente das classes de um outro componente. O diagrama de componentes mostra esta dependência.

Diagrama de Componentes

- Na ferramenta StarUML o diagrama de componentes está disponível dentro do “Implementation Model”;
- Na caixa de ferramentas estão disponíveis os elementos que podem ser usados no referido diagrama:

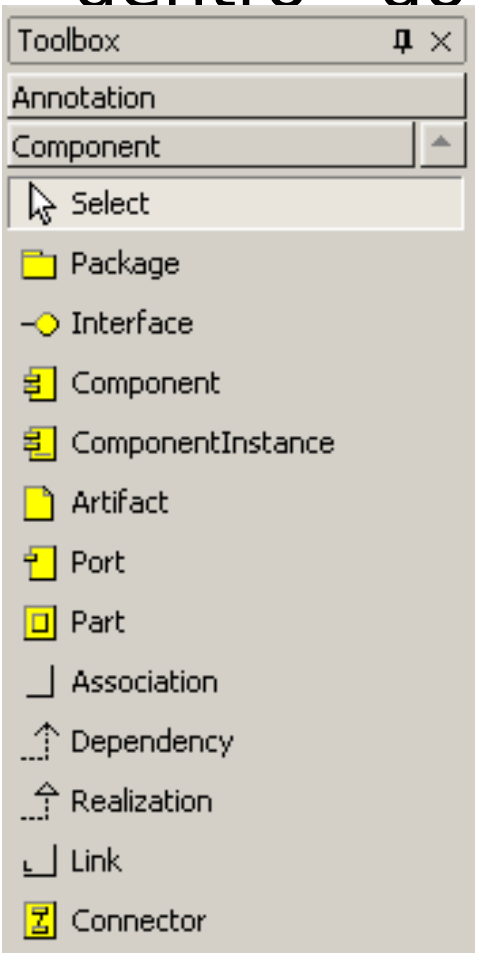


Diagrama de Componentes

- Abaixo é mostrado o diagrama de componentes de uma aplicação hipotética que permite controlar o funcionamento (acender/apagar) de uma determinada lâmpada:

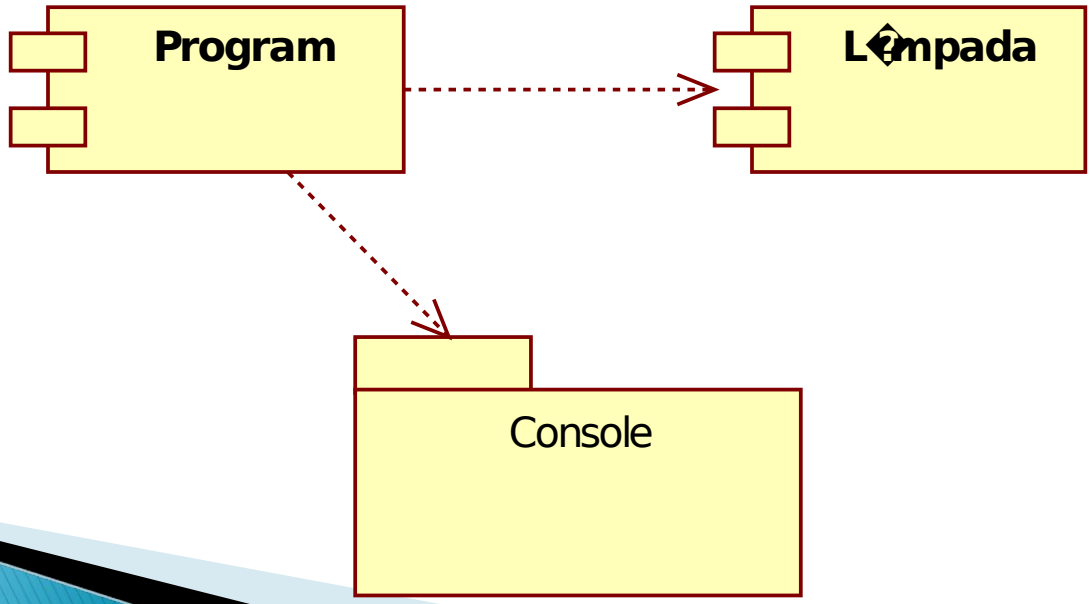


Diagrama de Distribuição

- Os diagramas de distribuição mostram a distribuição dos dispositivos de hardware necessários ao sistema;
- Permite identificar os servidores como nós do diagrama além da rede que relaciona os nós;
- Os componentes de software, por sua vez, vão estar mapeados nestes nós.

Diagrama de Distribuição

- Na ferramenta StarUML o diagrama de componentes está disponível dentro do "Deployment Model";
- A caixa de ferramentas mostra os elementos que podem ser utilizados no diagrama:

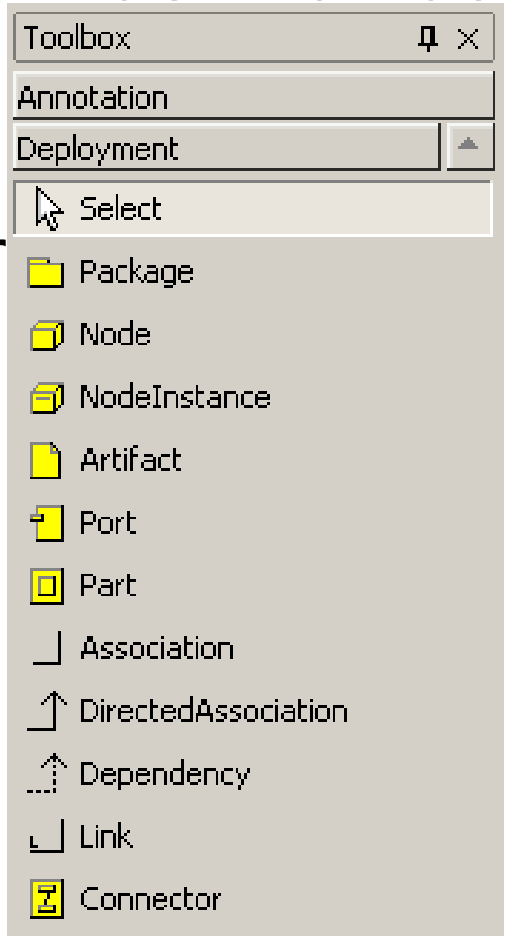


Diagrama de Distribuição

- Abaixo um exemplo de diagrama de distribuição para uma determinada aplicação que deverá ser executada em modo console em um computador com Windows XP: