Lição 2

Engenharia de Software Orientada a Objetos

Objetivos

Ao final desta lição, o estudante será capaz de:

- Entender os conceitos de orientação a objetos
- Entender os processos gerais de modelagem orientada a objeto
- Ver de uma forma geral as atividades de análise e modelagem orientada a objeto
- Entender a UML (*Unified Modeling Language* - Linguagem de Modelagem Unificada) e entender a atividade de modelagem

Tecnologia de Objetos

- *Pressman*, 1997
  - Tecnologia de Objetos é freqüentemente utilizada para incorporar alguns aspectos de uma visão orientada a objetos. Inclui análise, projeto e teste de métodos; linguagens de programação, bancos de dados e aplicações são criadas utilizando Orientação a Objetos

- *Taylor*, 1997, Tecnologia de Objetos
  - Tecnologia de Objetos é um conjunto de princípios que direcionam a construção de um software através de linguagens, bancos de dados, e outras ferramentas que dão suporte a estes princípios

Benefícios da Tecnologia de Objetos

- Conduz para o reuso
- Conduz a uma maior manutenibilidade dos módulos
- Conduz para um sistema orientado a objetos

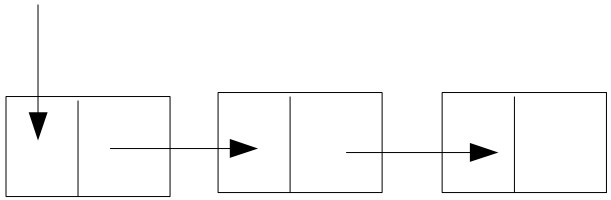
Objeto

- Esta é uma representação de uma entidade física, conceitual, ou abstrata

- Permite que os desenvolvedores de software representem os conceitos do mundo real no projeto do software

Avião

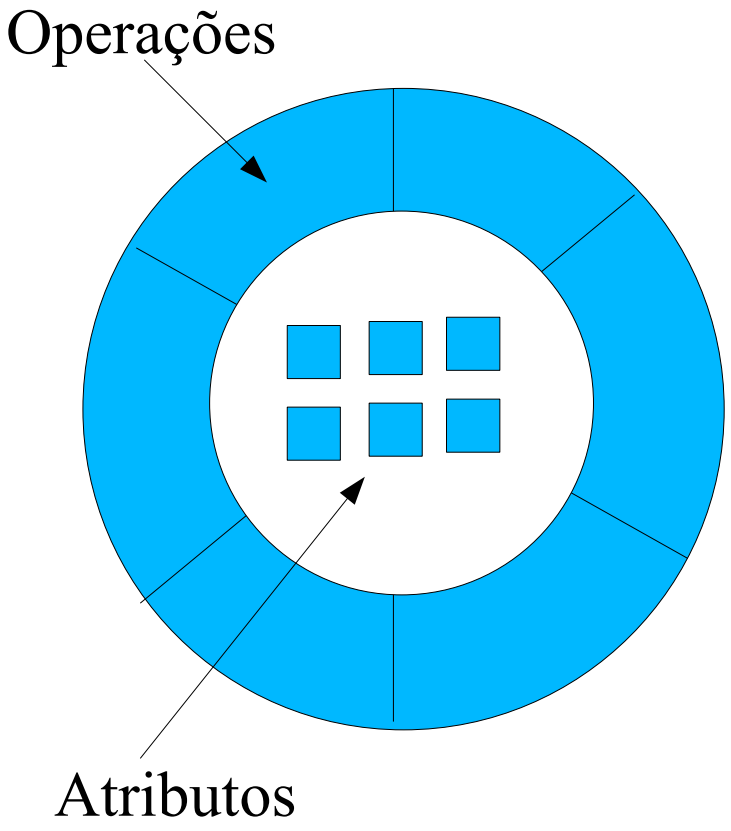
Processo Químico



Lista Sequencial

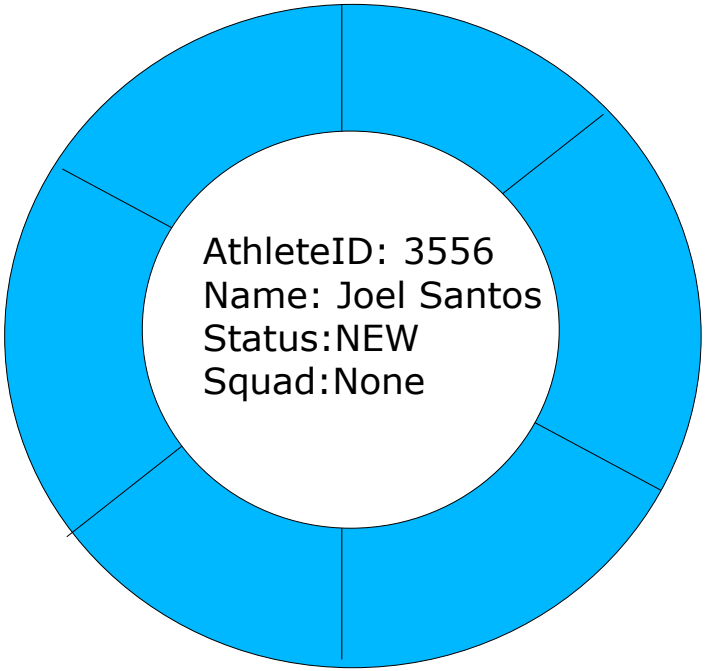
Objeto

- É uma entidade com um limite bem definido e identidade que encapsulam estado e comportamento



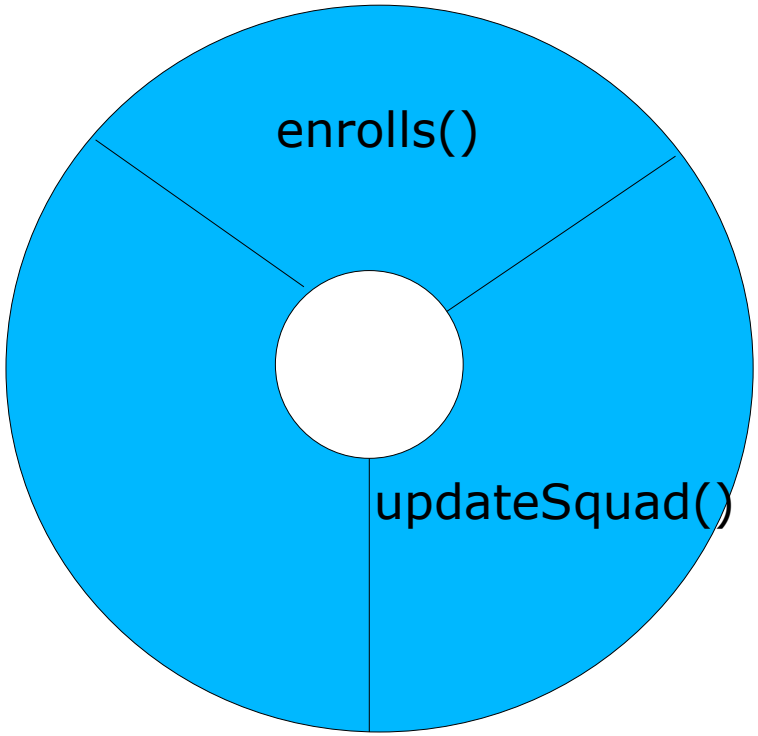
Estado do Objeto

- Uma das possíveis condições para que o objeto possa existir
- É implementado por um conjunto de propriedades chamados de atributos, junto com seus valores e as ligações com outros objetos



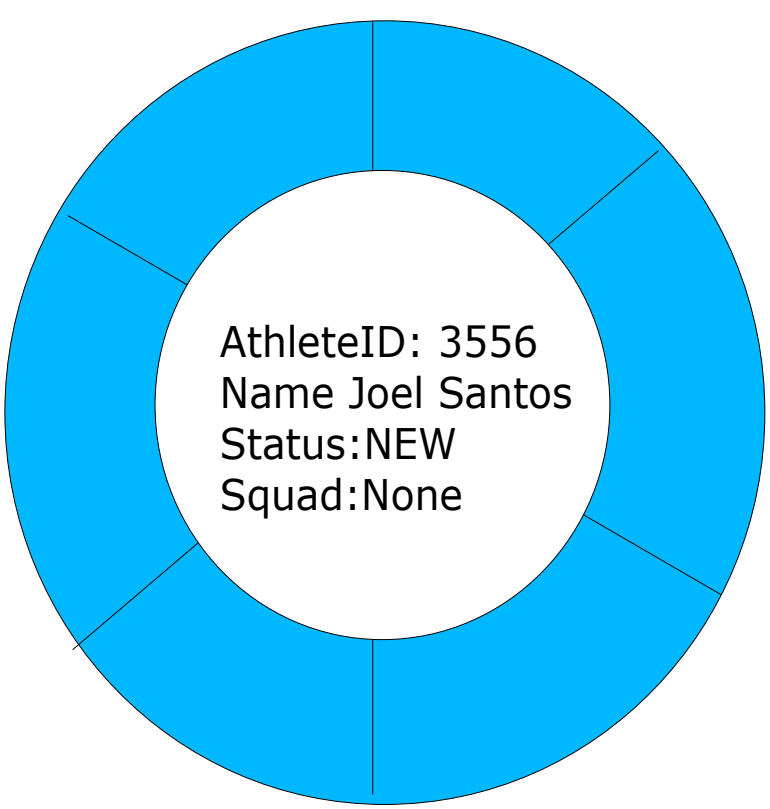
Comportamento do Objeto

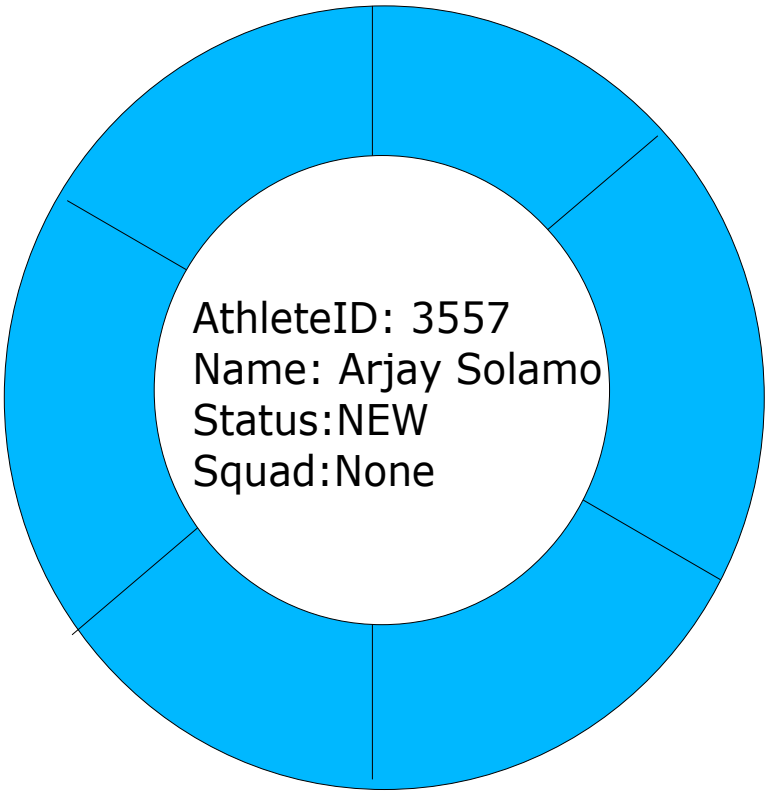
- Determina como um objeto age e reage
- Representado pelas operações que o objeto pode executar



Joel Santos

Identidade do Objecto





Quatro Princípios Básicos da Orientação a Objetos

- Abstração
- Encapsulamento
- Modularidade
- Hierarquia

Abstração

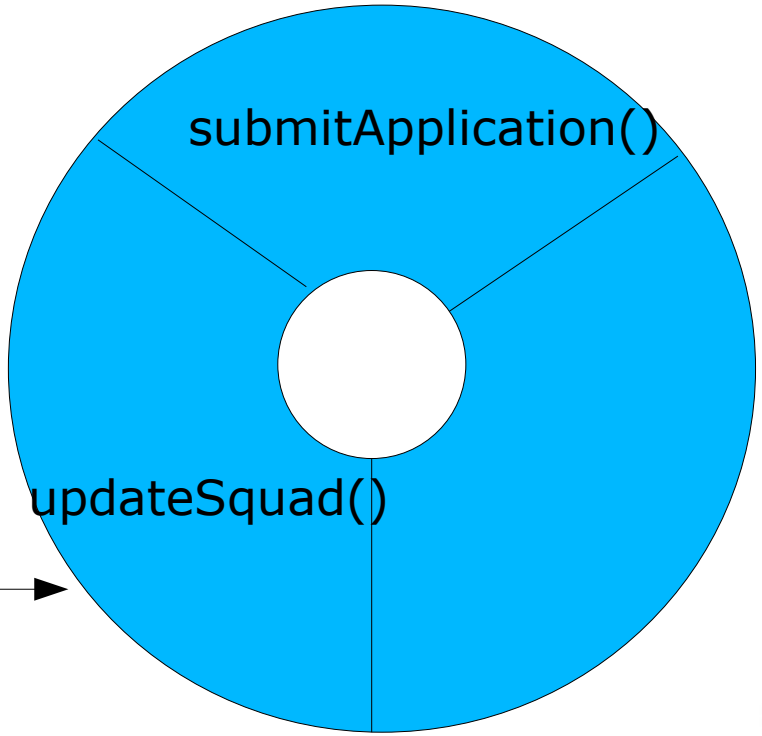
- Tipo de representação que inclui somente as coisas que são realmente importantes ou interessantes de um ponto de vista particular
- Processo de enfatizar o que é conhecido enquanto remove-se as distinções
- Gerenciar sistemas complexos e concentrar-se nas características essenciais que são distintas entre todos os outros tipos de sistemas
- Domínio e dependente de perspectiva

Encapsulamento

- Localiza características de uma entidade dentro de uma "caixa-preta" abstrata, e esconde da implementação estas características da através de uma interface simples
- Também é conhecido como ocultação de informação, isto permite que os usuários utilizem os objetos sem conhecer como a interface foi implementada
- Oferece dois tipos de proteção

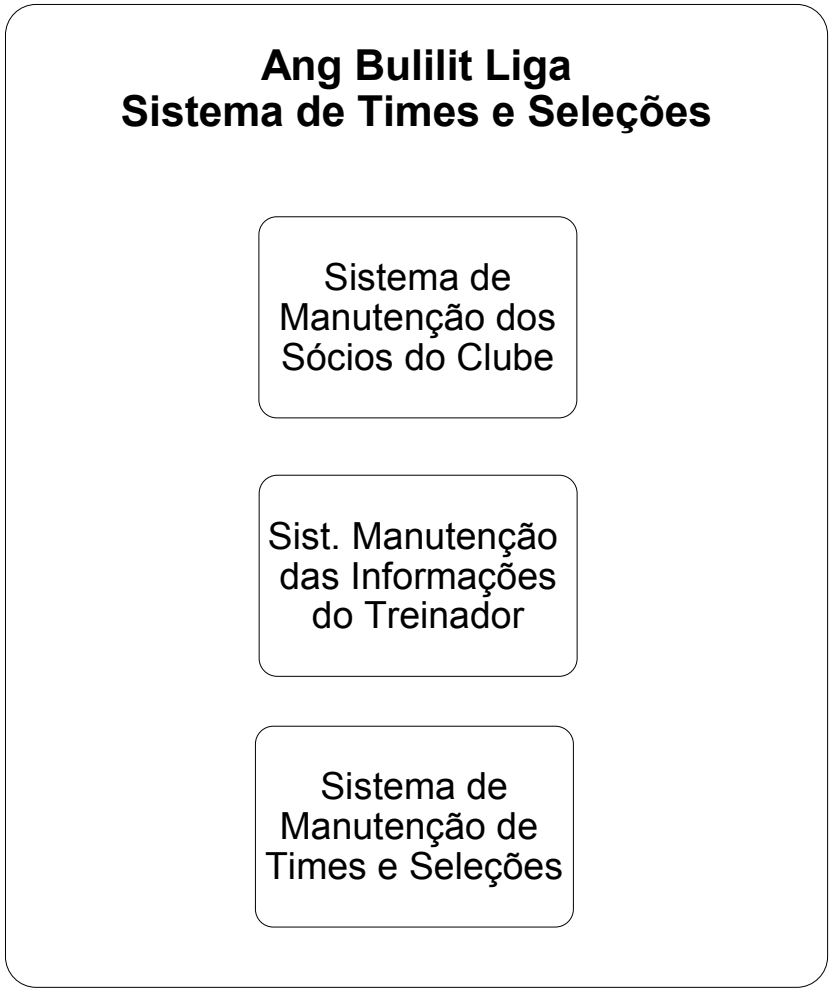
Encapsulamento

- Joel Santos está assinando a "Esquadra de Treinamento"
- A chave é uma **mensagem para a interface**



updateSquad("Treinamento")

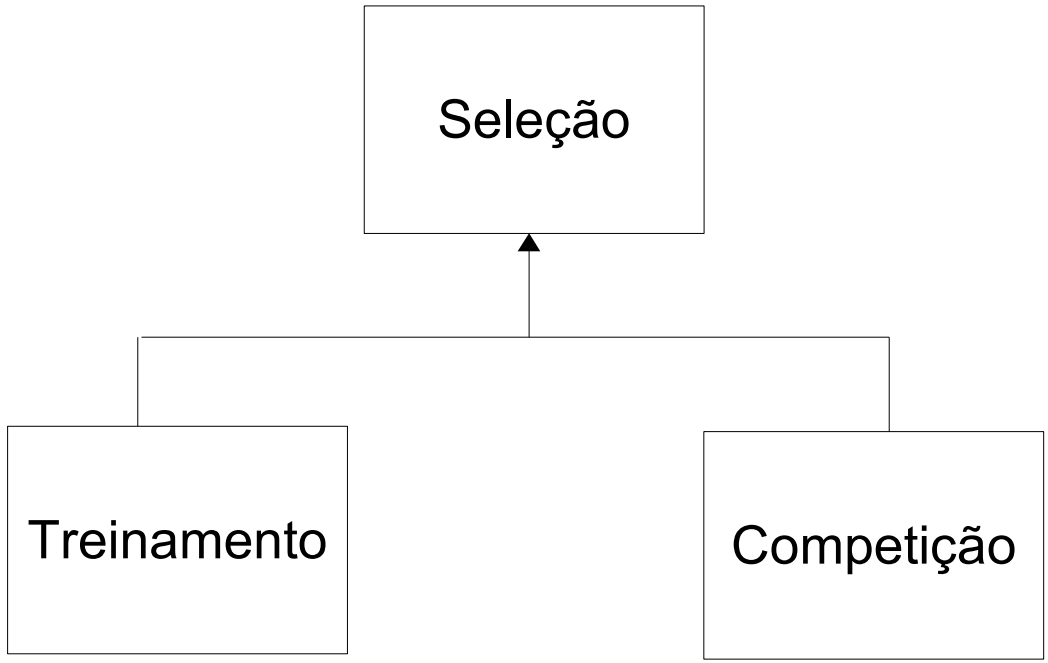
Modularidade



- Decomposição lógica e física de coisas grandes e complexas em componentes menores e manejáveis
- Quebrar um pedaço grande de um sistema em subsistemas pequenos e manejáveis.
- Pode ser desenvolvido independentemente, contanto que as interações sejam estabelecidas e muito bem compreendidas

Hierarquia

- Em qualquer posição ou ordem de abstrações dentro de uma estrutura coerente
- Tipos de Hierarquia
  - Agregação
  - Classe
  - Retenção
  - Herança
  - Partição
  - Especialização
  - Tipo

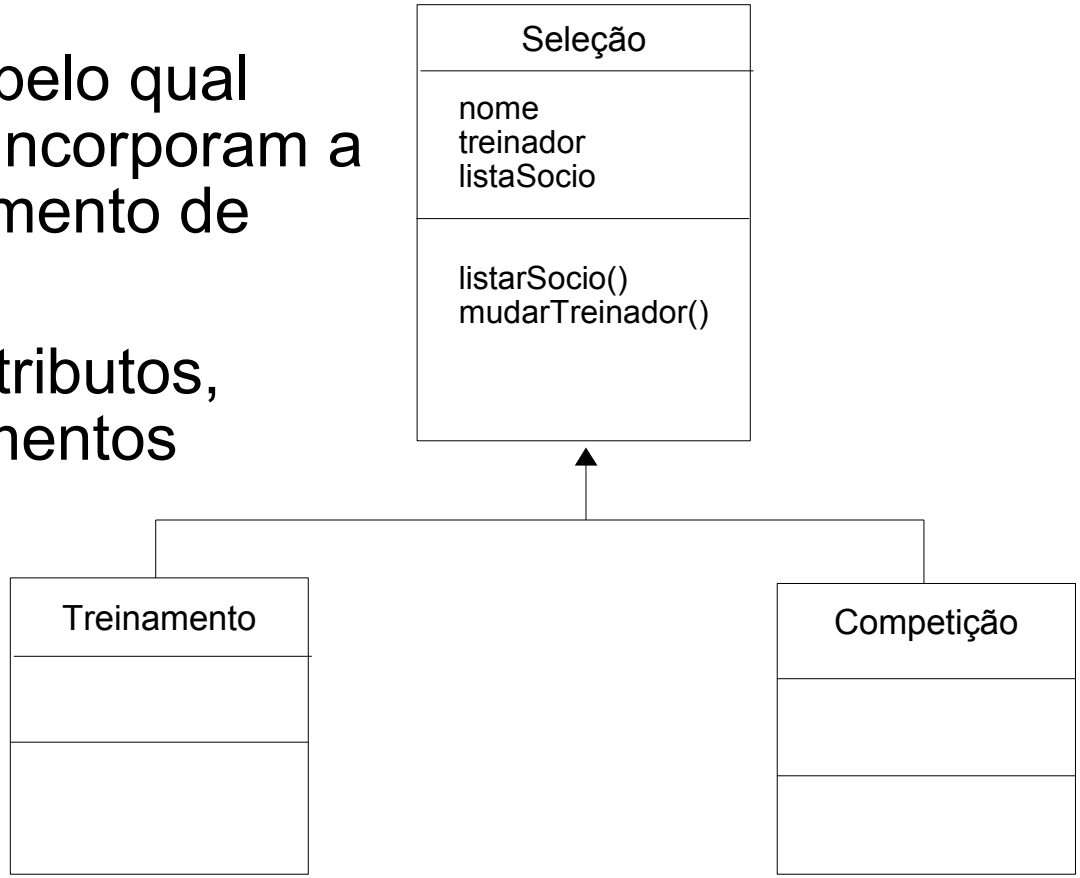


Generalização

- É uma forma de associação com uma classe compartilhando a estrutura e/ou comportamento de uma ou mais classes
- Isto define a hierarquia de abstrações com uma subclasse herdada de uma ou mais superclasses
  - Herança Simples
  - Herança Múltipla
- É um tipo de **relacionamento**

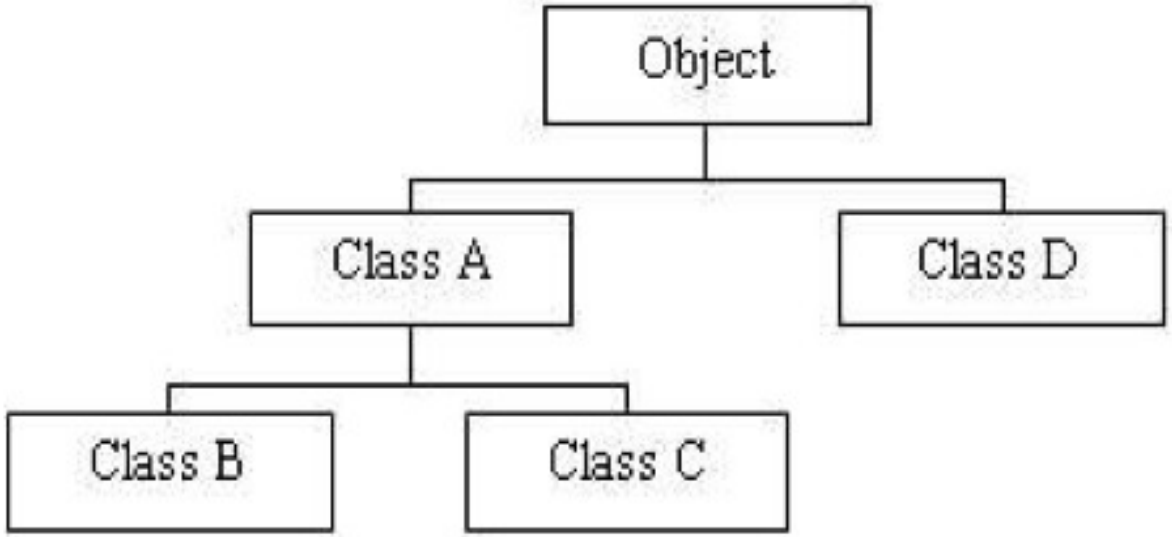
Herança

- Este é um mecanismo pelo qual elementos específicos incorporam a estrutura e o comportamento de elementos mais gerais
- Uma classe herda os atributos, operações e relacionamentos



Herança

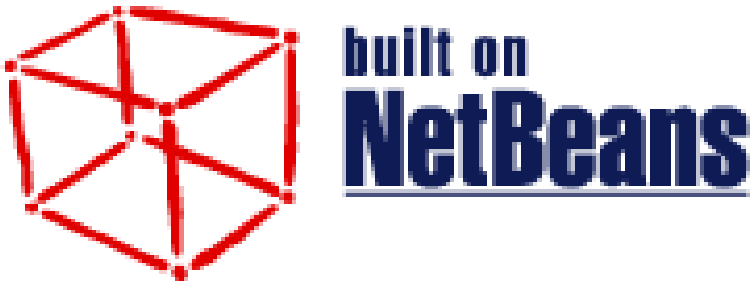
- Em Java, todas as classes, incluindo as classes da Java API, são subclasses da superclasse **Object**
- Uma simples hierarquia de classes pode ser mostrada como:



- Superclasse – qualquer classe acima de uma classe especificada
- Subclasse – qualquer classe abaixo de uma classe especificada

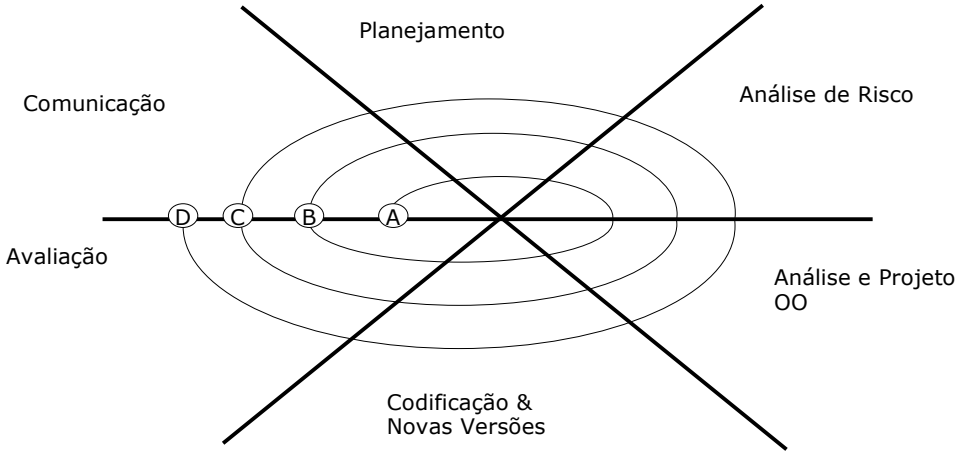
Herança

- Passaremos agora para o NetBeans

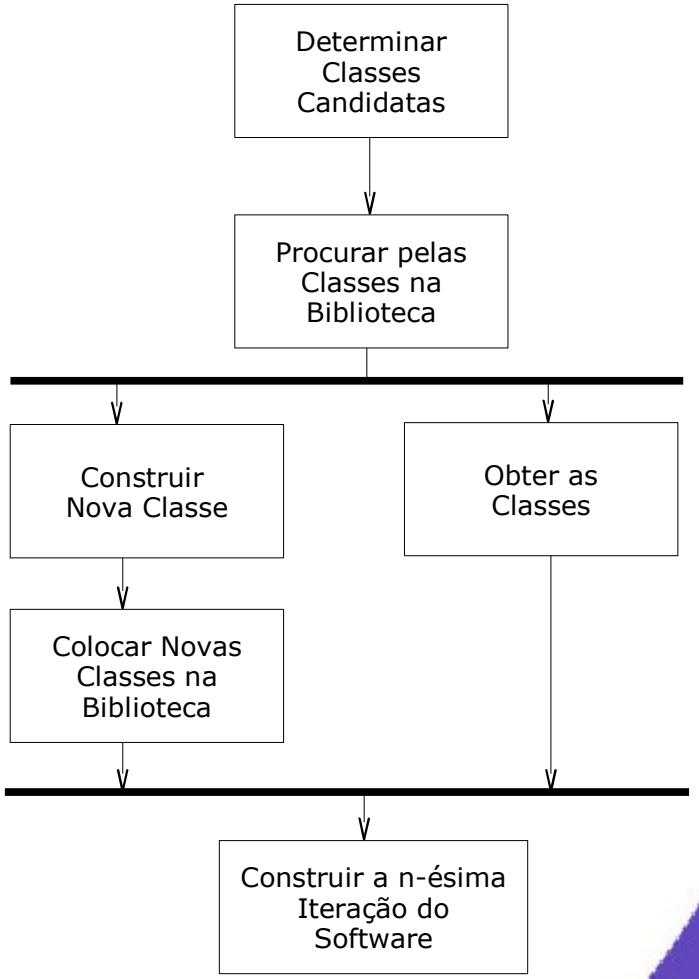


Agregação

Modelo de Processo Orientado a Objeto



A. Projeto Inicial de Software
B. Manutenção do Novo Software
C. Evolução do Software
D. Desenvolvimento de outro sistema inter-relacionado



Características do Modelo de Processo OO

- Segue um caminho de evolução em espiral
- Por natureza, é iterativo
- Suporta o conceito de reutilização por ser um processo de engenharia baseado em componentes
- Focado em uma estratégia de arquitetura centralizada, identificando e usando uma arquitetura com uma linha de tempo bem definida

Arquitetura de Software

- M. Shaw and D.Garlan [SHA95a]
  - Arquitetura de Software refere-se a toda estrutura do software e ao modo como esta estrutura oferece integridade conceitual para o sistema

- Rational Rose
  - Arquitetura de Software é sobre tomar decisões em como o sistema será construído; ela controla o desenvolvimento iterativo e incremental de um sistema durante todo o seu ciclo de vida

Pacote

- É utilizado como um mecanismo de propósito geral para organizar os elementos em grupos
- É um elemento modelo que pode conter outros elementos modelos
- Utilizado para:
  - organizar o modelo em desenvolvimento
  - servir como uma unidade do gerenciamento de configuração

Subsistema

- É um conceito que combina a definição de um pacote, isto é, pode conter outros elementos, e uma classe, isto é, tem comportamento (interage com outros elementos ou sistemas)

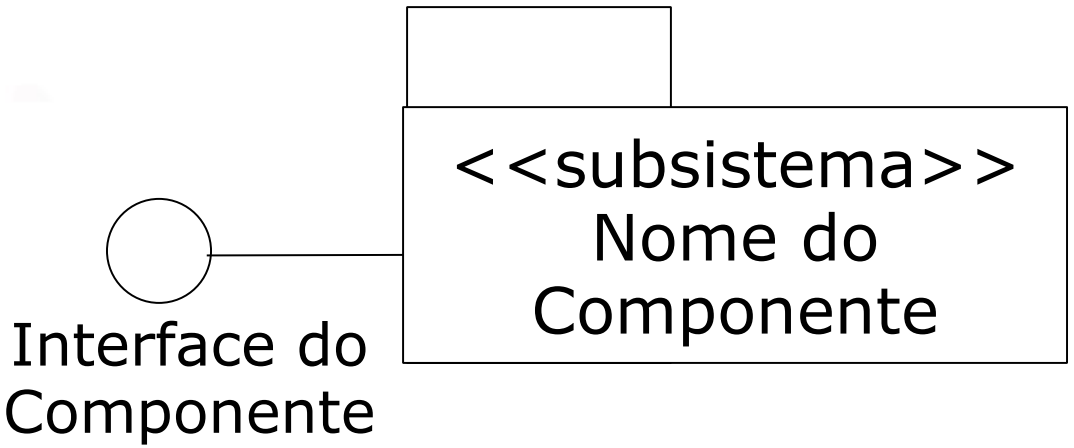
- Realiza uma ou mais interfaces o que define comportamento

Componente

- É uma parte do sistema substituível, não-trivial e quase independente
- Tem uma função bem definida no contexto de arquitetura bem definida
- Pode ser:
  - componente do código-fonte
  - componente em tempo de execução
  - componente executável

Subsistema e Componente

- Componentes são a realização física de um projeto abstrato
- Subsistemas podem ser usados para representar o componente no projeto



Análise Orientada a Objeto

- O objetivo principal da análise orientada a objetos é desenvolver uma série de modelos que descrevam como o software é executado satisfazendo os requisitos definidos pelo cliente

- A intenção da análise orientada a objetos é definir classes, seus relacionamentos e comportamentos que sejam relevantes para o sistema. Devido aos requisitos do cliente influenciarem a criação dos modelos, esta fase ou atividade também pode ser chamada de **Engenharia de Requisitos**

Os Cinco Princípios da Análise

- O domínio da informação é modelado
- A função módulo é descrita
- O modelo comportamental é representado
- Os modelos são particionados para que possam expor maiores detalhes
- Modelos recentes representam a essência do problema, enquanto que modelos tardios proveêm uma implementação detalhada

Projeto Orientado a Objetos

- Ele transforma o modelo criado a partir da análise orientada a objetos no modelo de design que serve como uma guia para a construção do software

- Ele deve descrever a organização dos dados dentro dos atributos e os detalhes individuais de cada operação

Cinco Princípios Básicos de Projetos

- Unidades Linguísticas modulares
- Poucas interfaces
- Interfaces pequenas/ fraca acoplagem
- Interface explícitas
- Ocultamento de informação

Métodos de Projeto Orientado a Objetos

- **Método Booch**
  - Envolve processos de micro-desenvolvimento e macro-desenvolvimento
- **Método de Coad e Yourdon**
  - Ele endereça não só a aplicação, mas também a infra-estrutura para a aplicação
- **Método de Jacobson**
  - Enfatiza o rastreamento do modelo de análise OOSE

Linguagem de Modelagem Unificada

- A linguagem de Modelagem Unificada (UML) é o padrão de linguagem para especificação, visualização, construção, e documentação de todo o trabalho produzido em um sistema

- Ela unifica as notações de Booch, Rambaugh e Jacobson, somadas as contribuições de outros submetidas a OMG

- Ela propõe um padrão técnico para a trocas de modelos e designs

UML NÃO é

- Não é um método ou uma metodologia
- Não indica um processo particular
- **Não é uma linguagem de programação**

Diagrama de Caso de Uso

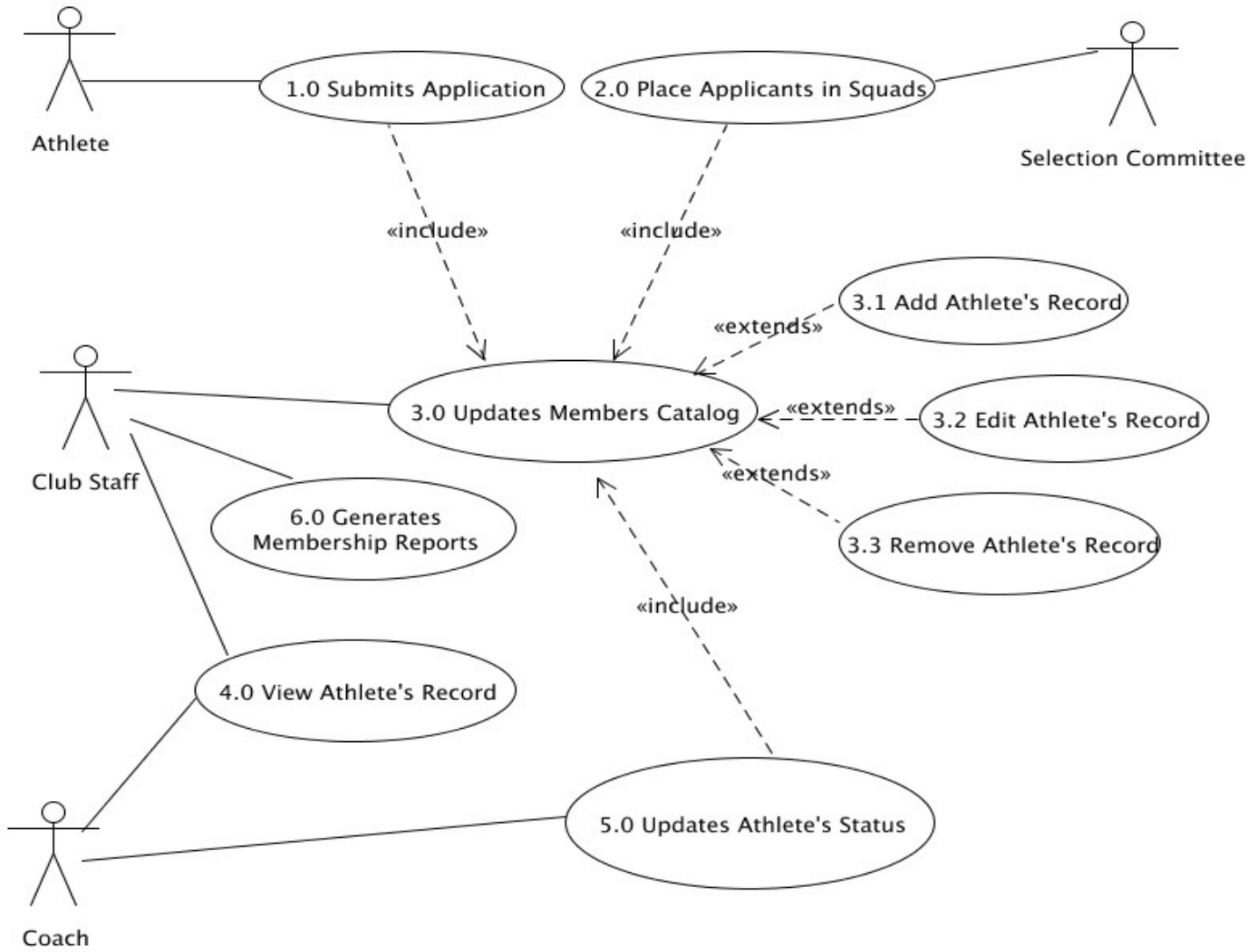


Diagrama de Classe

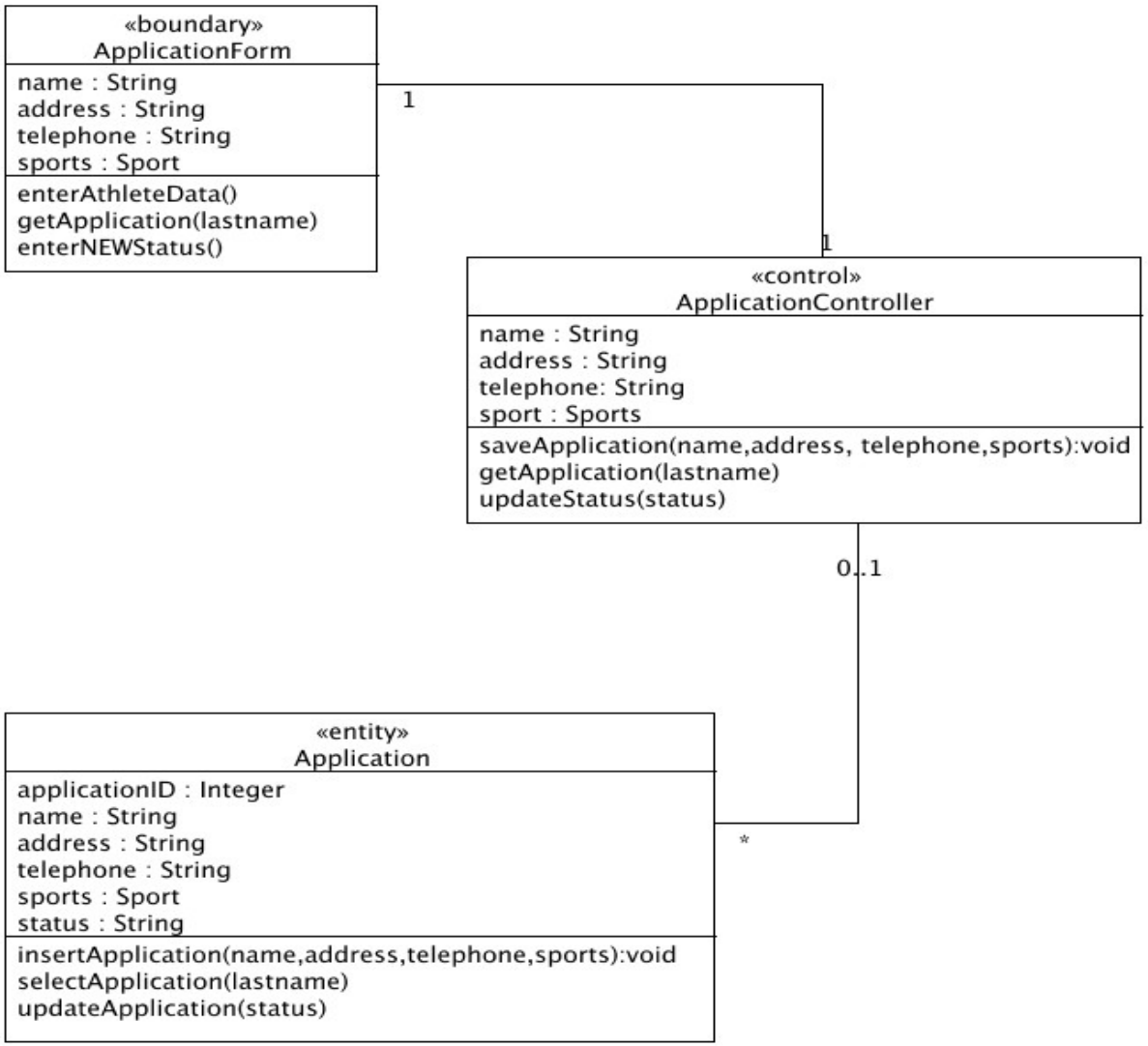


Diagrama de Pacote

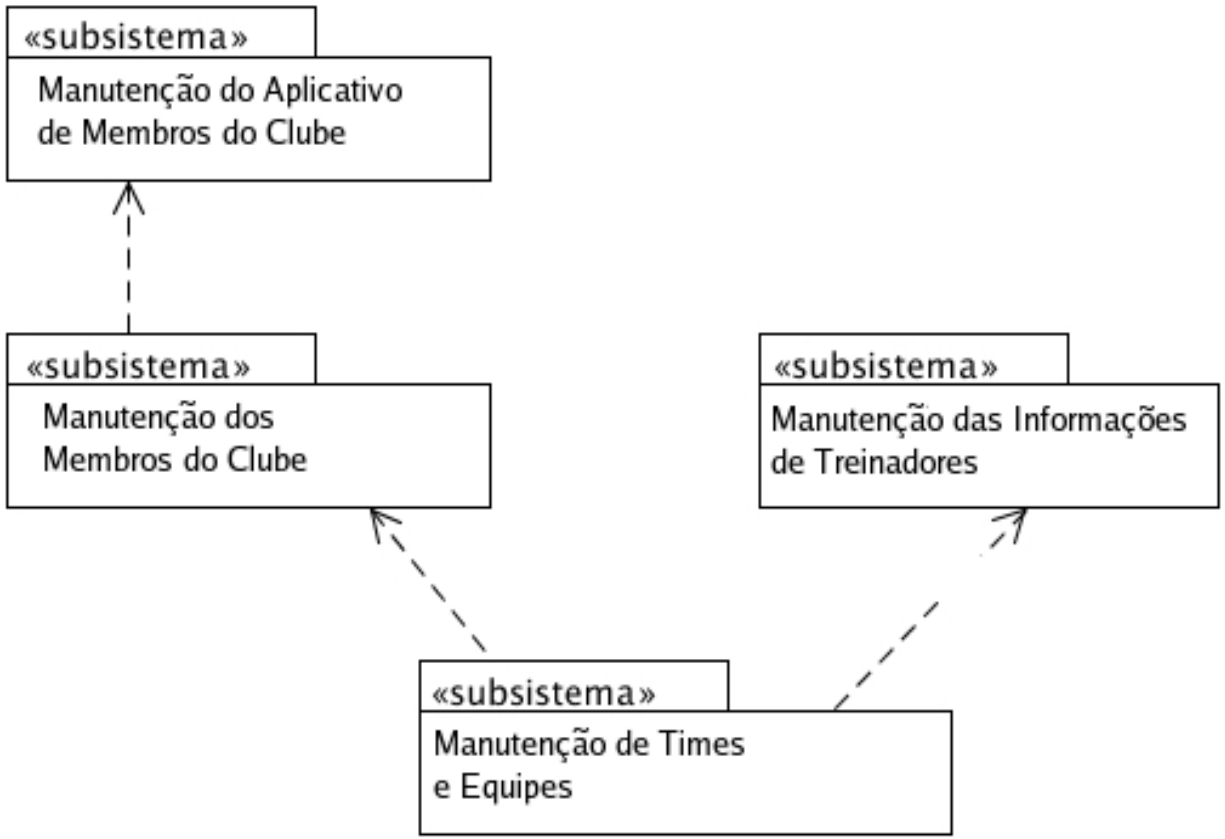


Diagrama de Atividade

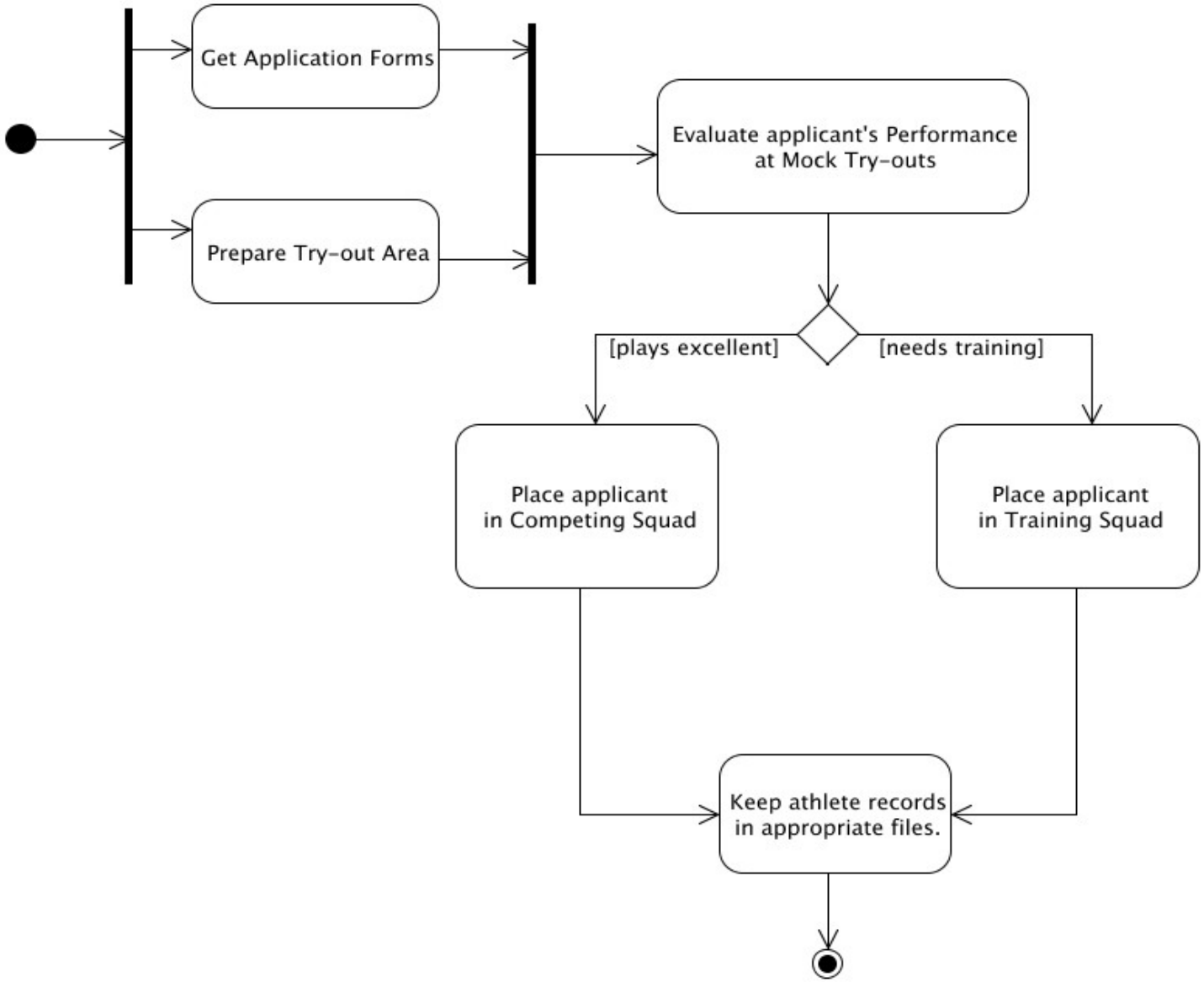


Diagrama de Seqüência

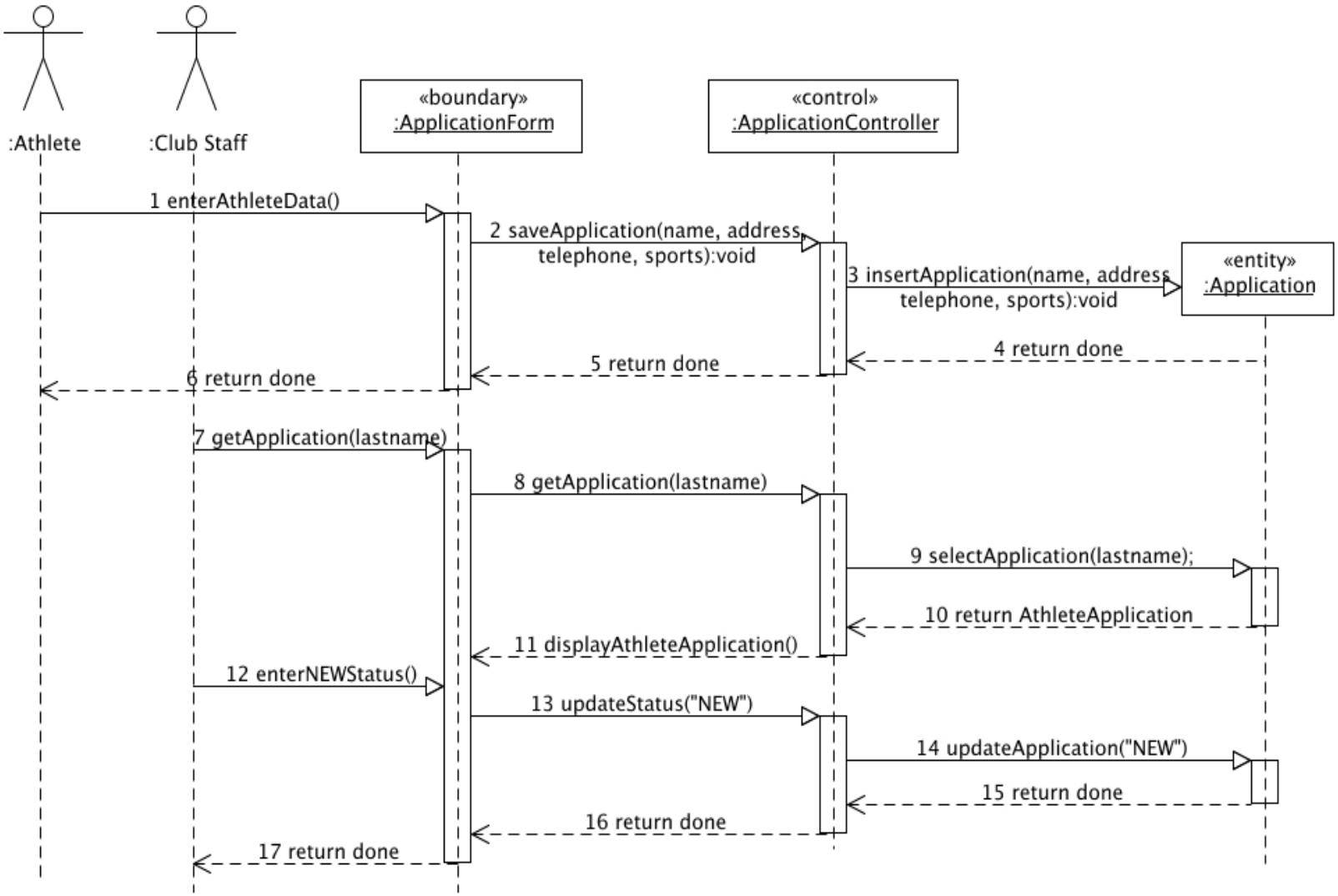


Diagrama de Colaboração

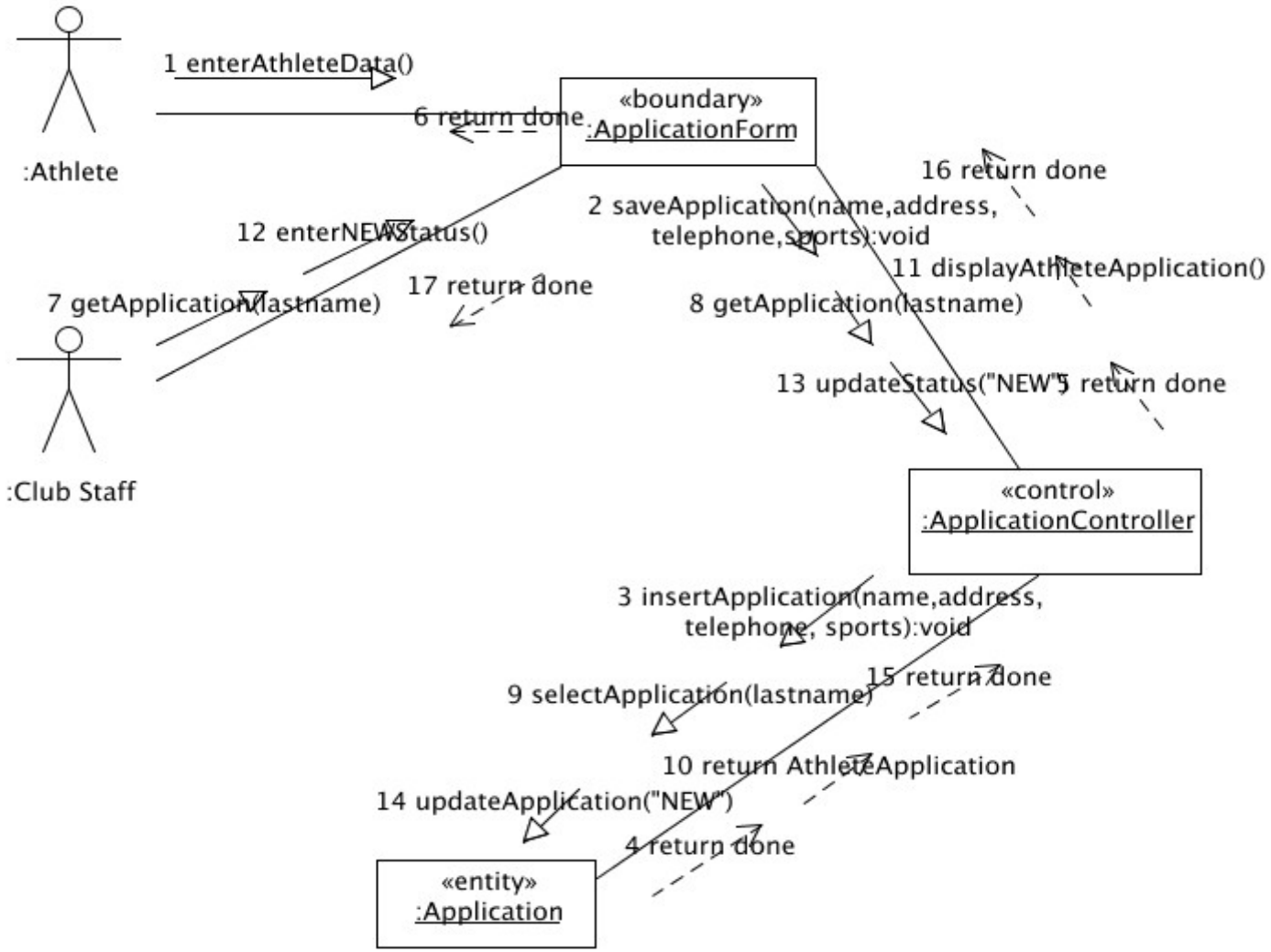


Diagrama de Estado

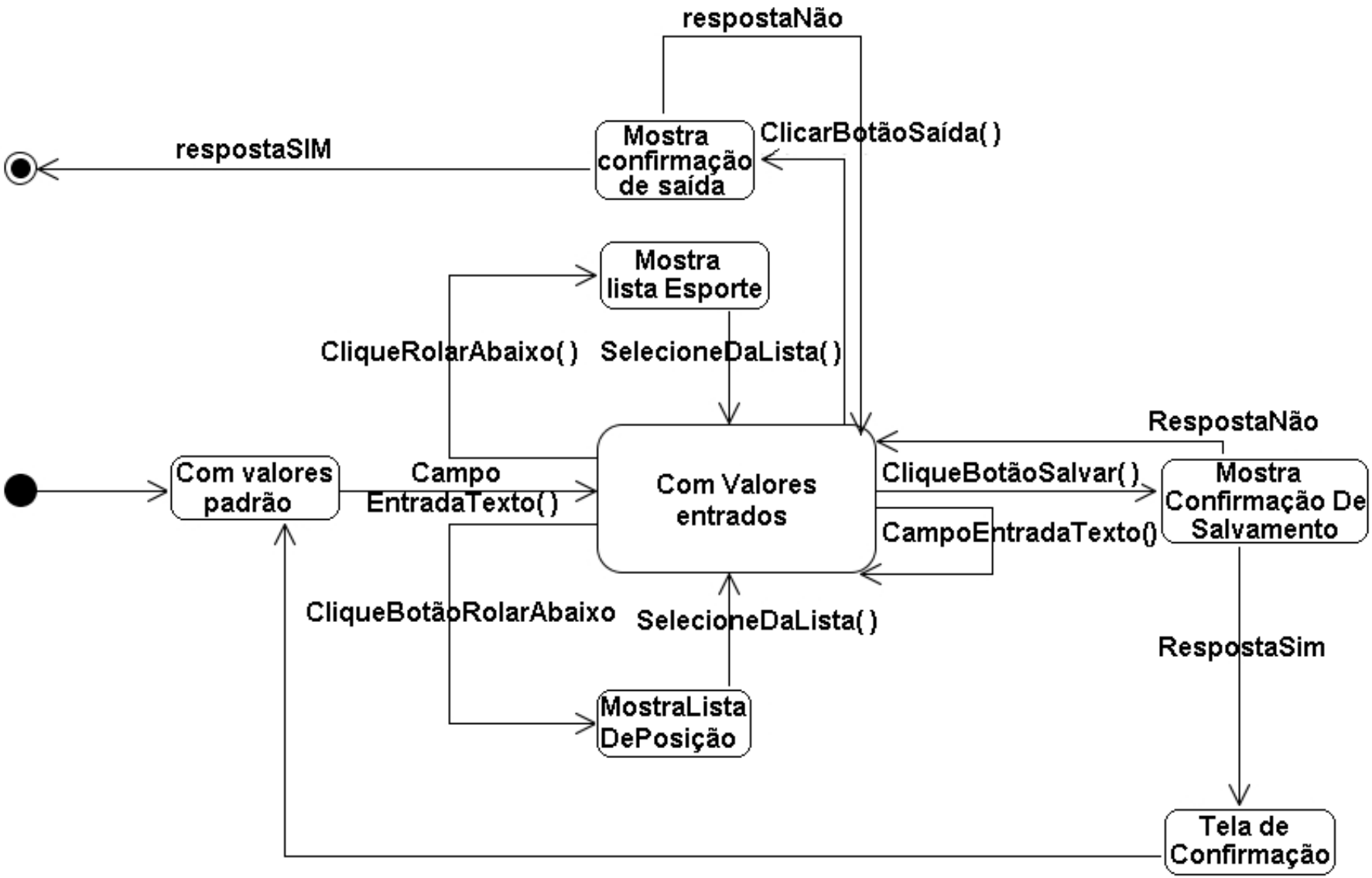


Diagrama de Componentes

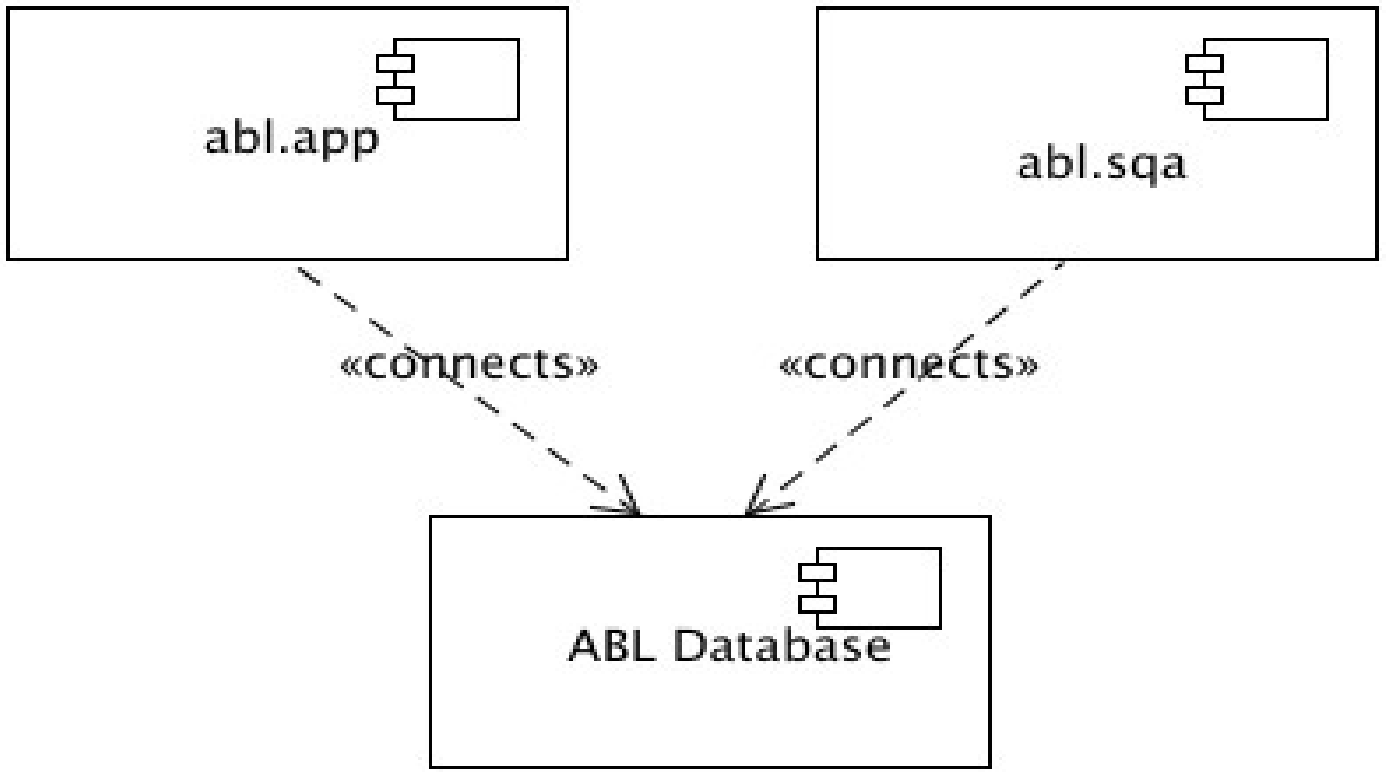
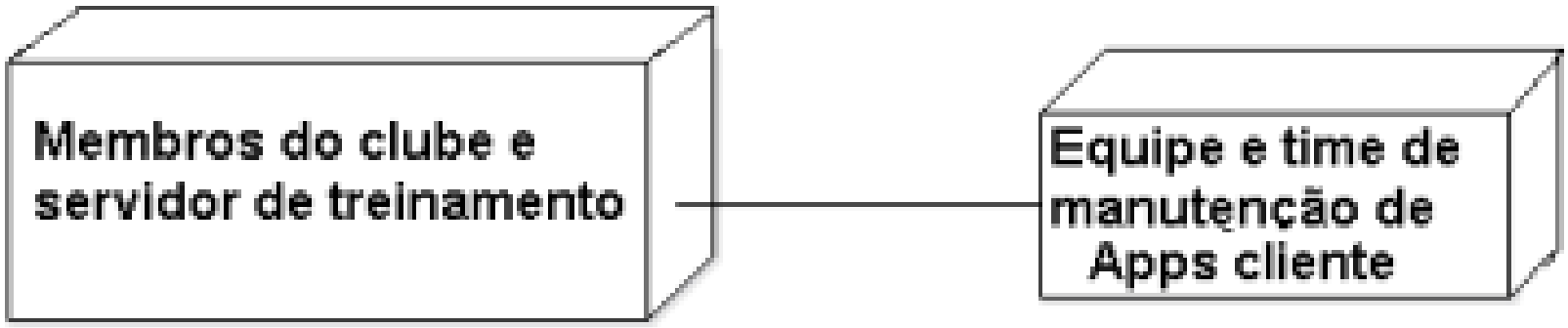


Diagrama de Implementação



Sumário

- Tecnologia de Objetos
- Objetos
- Quatro princípios básicos: Abstração, Encapsulamento, Modularidade e Hierarquia
- Modelo de Processo Orientado a Objeto
- Características do Processo Orientado a Objeto
- Arquitetura de Software
- Pacote
- Subsistema e Componente
- Análise Orientado a Objeto
- Projeto Orientado a Objeto
- UML

Parceiros

- **Os seguintes parceiros tornaram JEDI™ possível em Língua Portuguesa:**