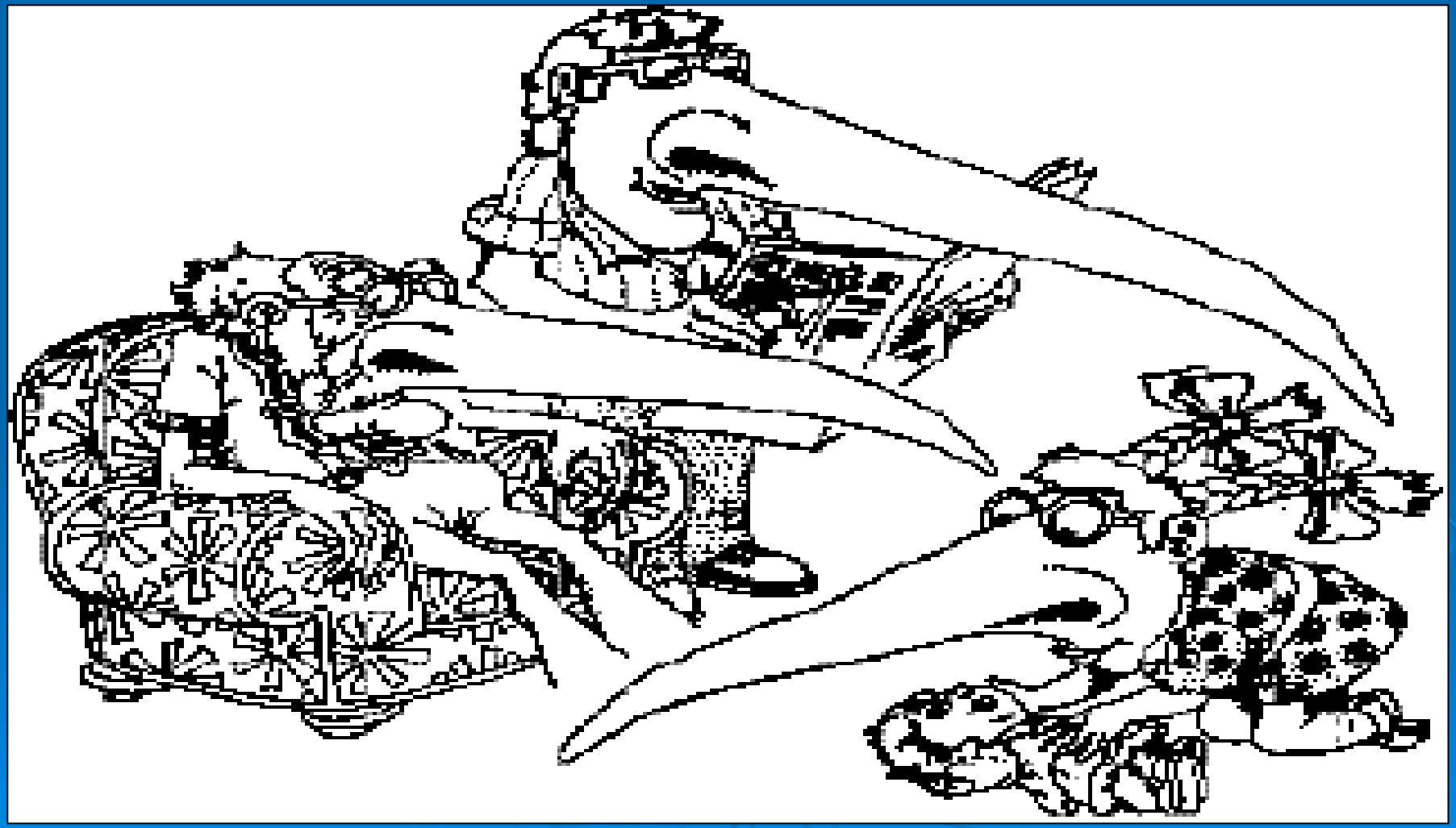
Conceitos Básicos

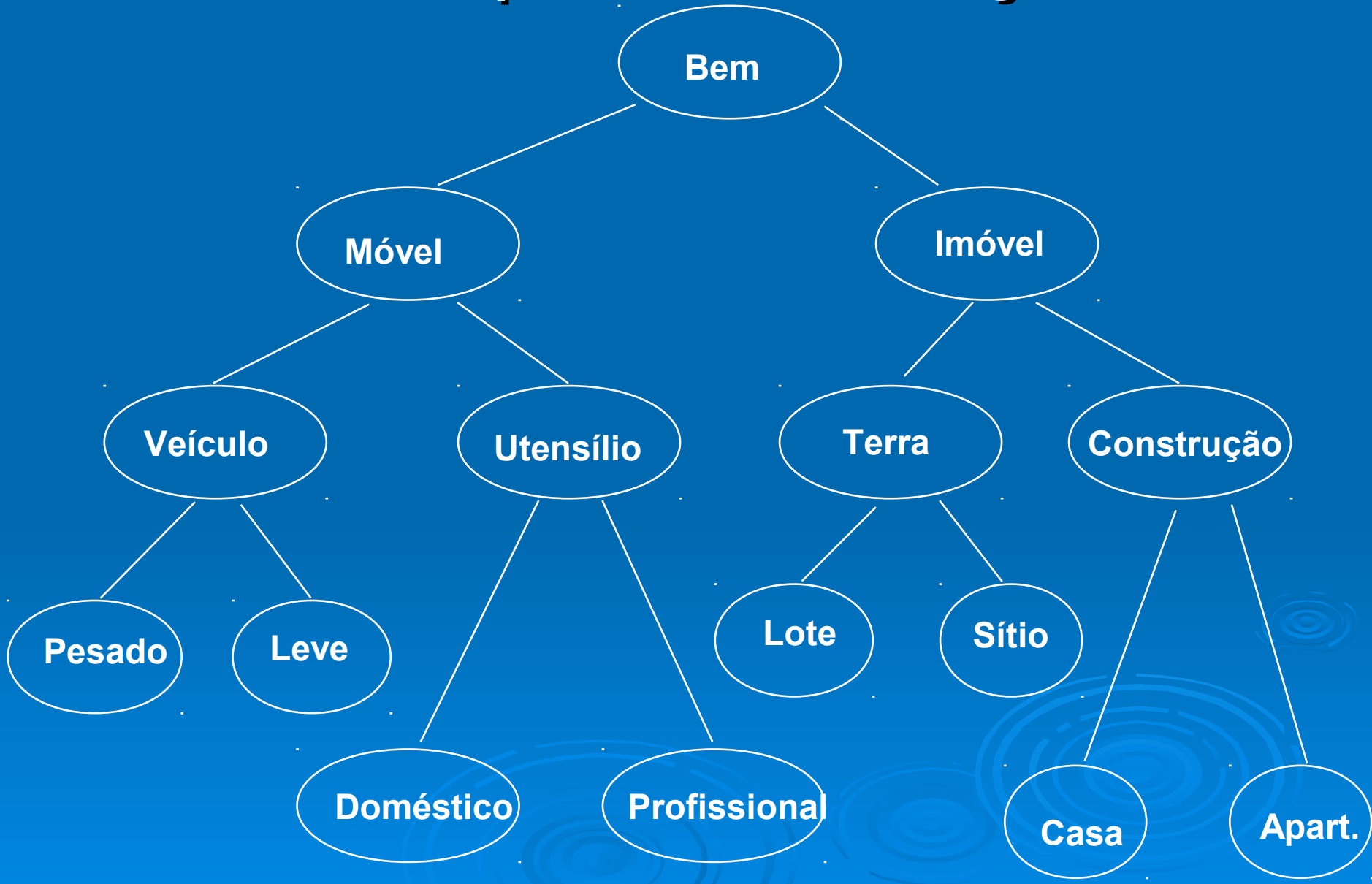
- Herança: “Filho de peixe, peixinho é!”



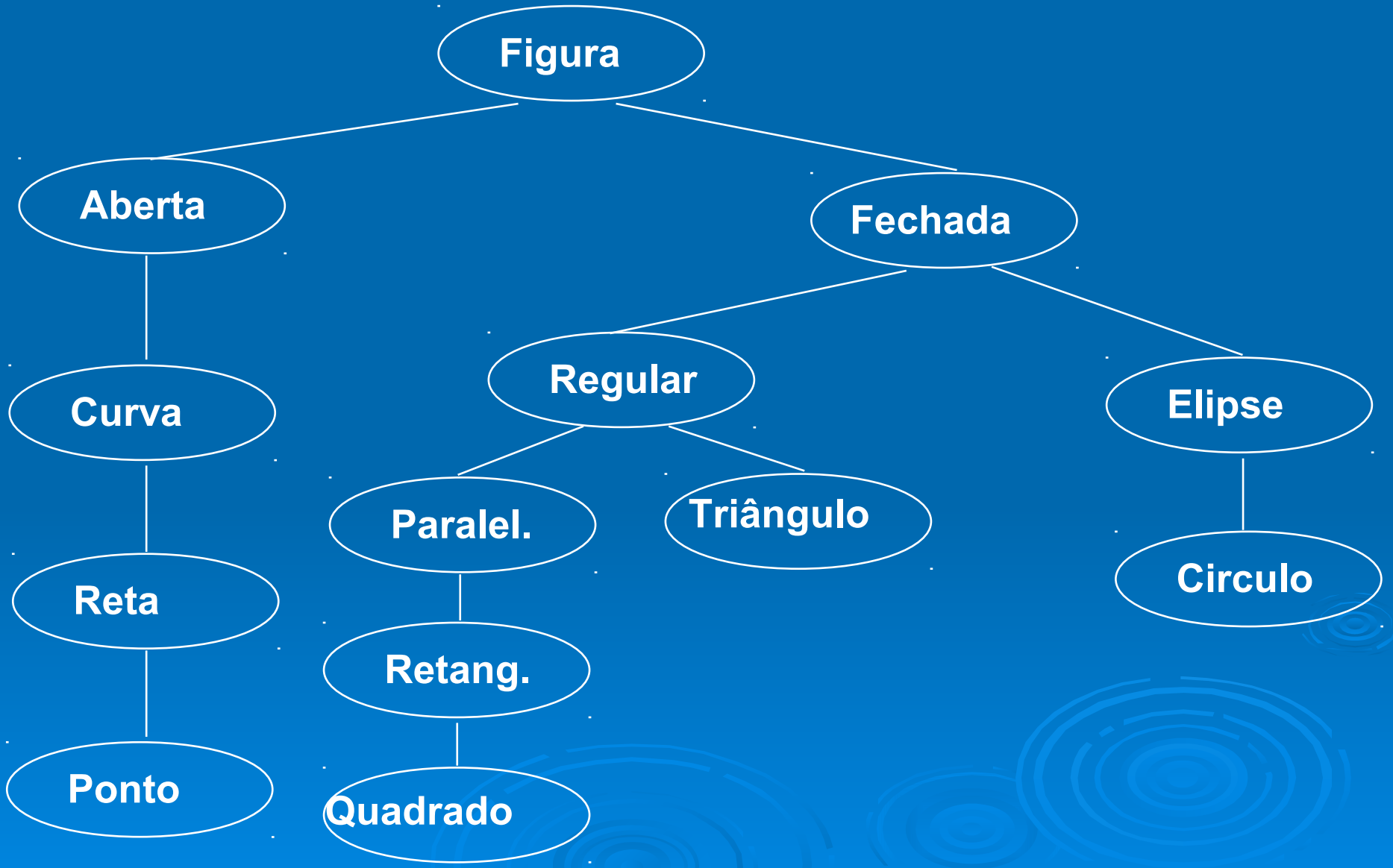
Conceitos Básicos

- Herança
  - Capacidade de uma classe de definir o seu comportamento e sua estrutura aproveitando definições de outra classe.
  - Funciona como um mecanismo de Generalização / Especialização
  - Classes genéricas contêm definições comuns a um grande número de objetos.

Exemplo de Herança



Exemplo de Herança



Herança

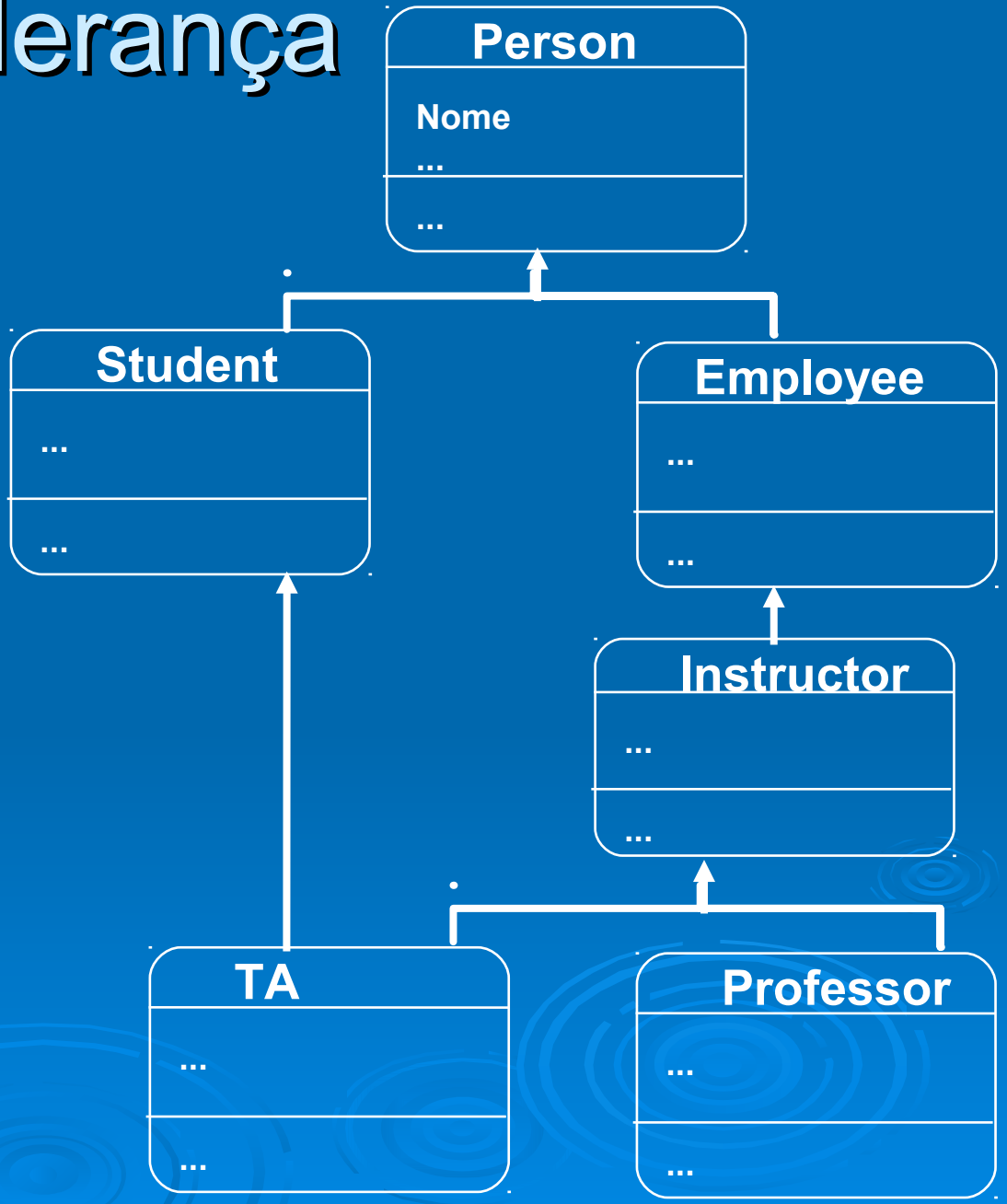
- Conceitos Básicos
  - Subclasse
    - Classe que herda definições de uma outra classe.
    - Uma subclasse, em geral, acrescenta novas características e/ou comportamentos às definições herdadas.
  - Superclasse
    - Classe cujas características e comportamentos são herdados por outras.

Herança

- Conceitos básicos
  - Classes Abstratas
    - Classes que não produzem instâncias.
    - Agrupam características e comportamentos que serão herdados por outras classes.
    - Podem prover implementações completas de determinados comportamentos.
    - Podem fornecer padrões de comportamentos que serão implementados nas subclasses.

Herança

- Dois tipos:
  - Herança Simples
  - Herança Múltipla

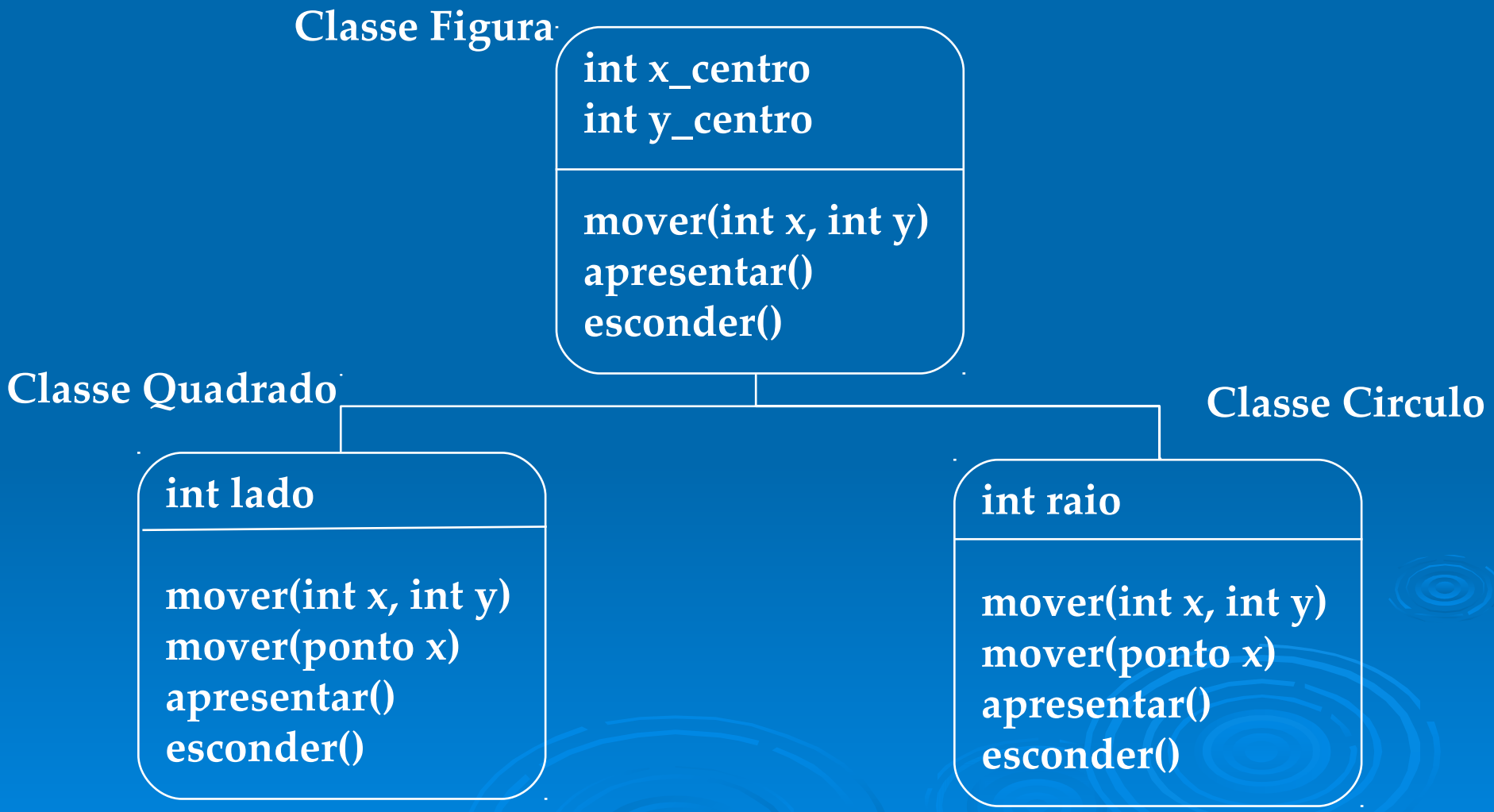


Conceitos Básicos

- Polimorfismo
  - Capacidade de dois ou mais tipos de objetos de responderem à mesma mensagem de maneira particular.

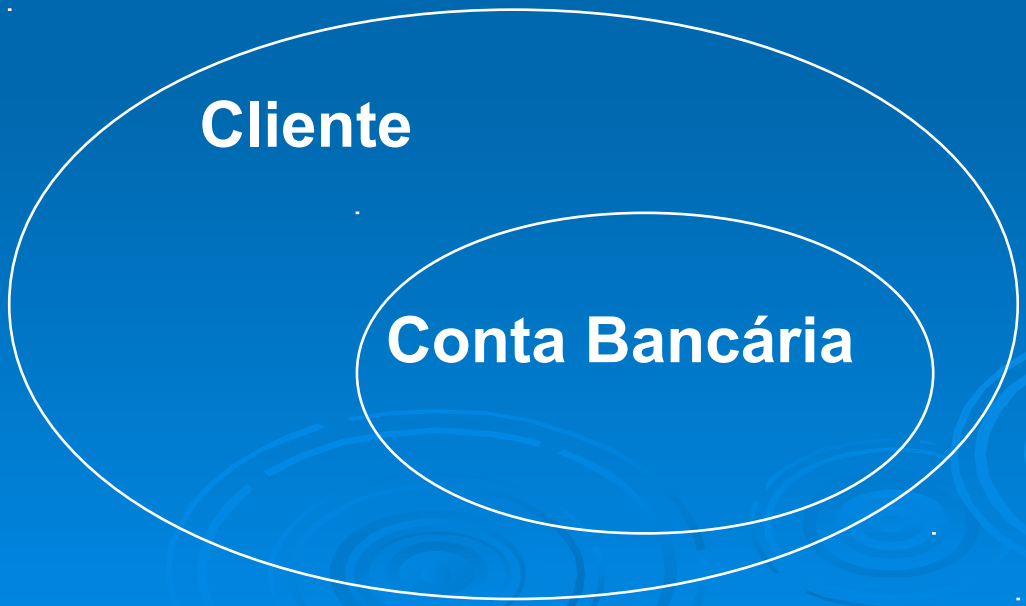
    Ou
  - Capacidade de um objeto responder à mesma mensagem, com métodos distintos, a depender:
    - da classe que esteja sendo requisitada
    - dos tipos dos parâmetros recebidos.
  - Sobrecarga
    - O objeto que envia a mensagem não precisa conhecer especificamente o tipo do objeto receptor.
    - Facilidade de extensão de programas OO

Exemplo de Polimorfismo



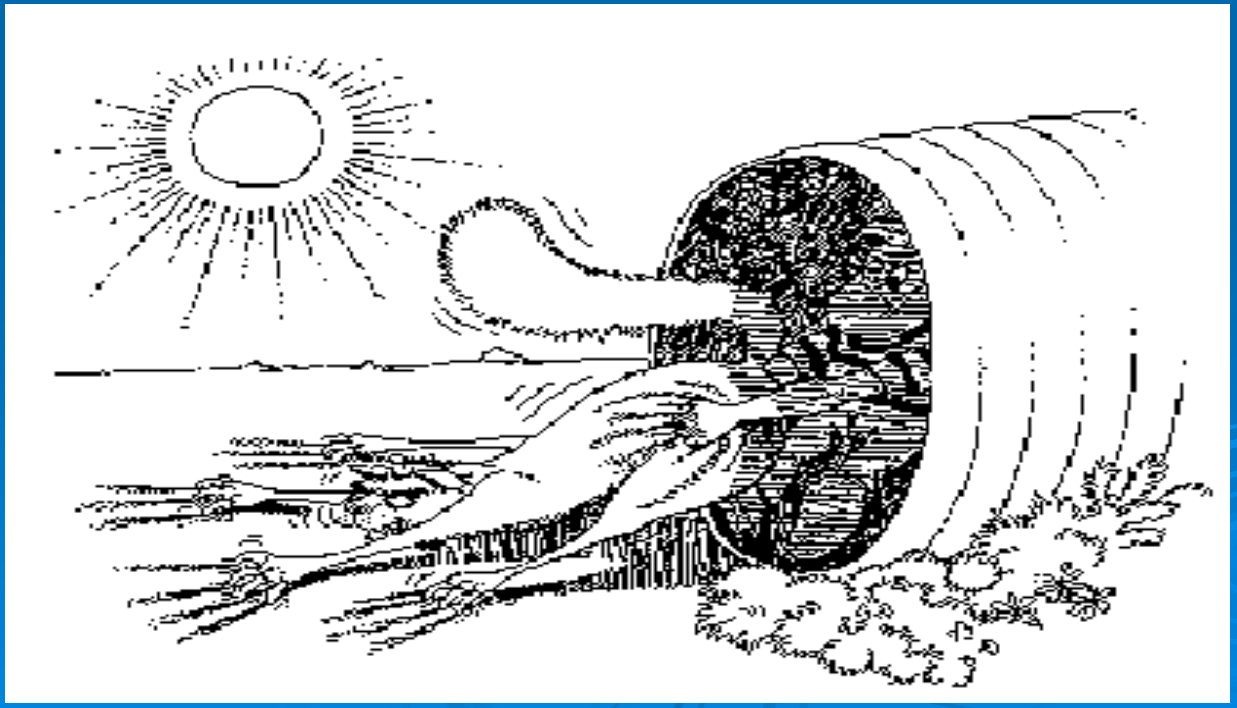
Orientação a Objetos

- Objeto Complexo
  - Objeto que possui outros objetos como parte da sua estrutura.
  - Objetos compostos por outros objetos.



Orientação a Objetos

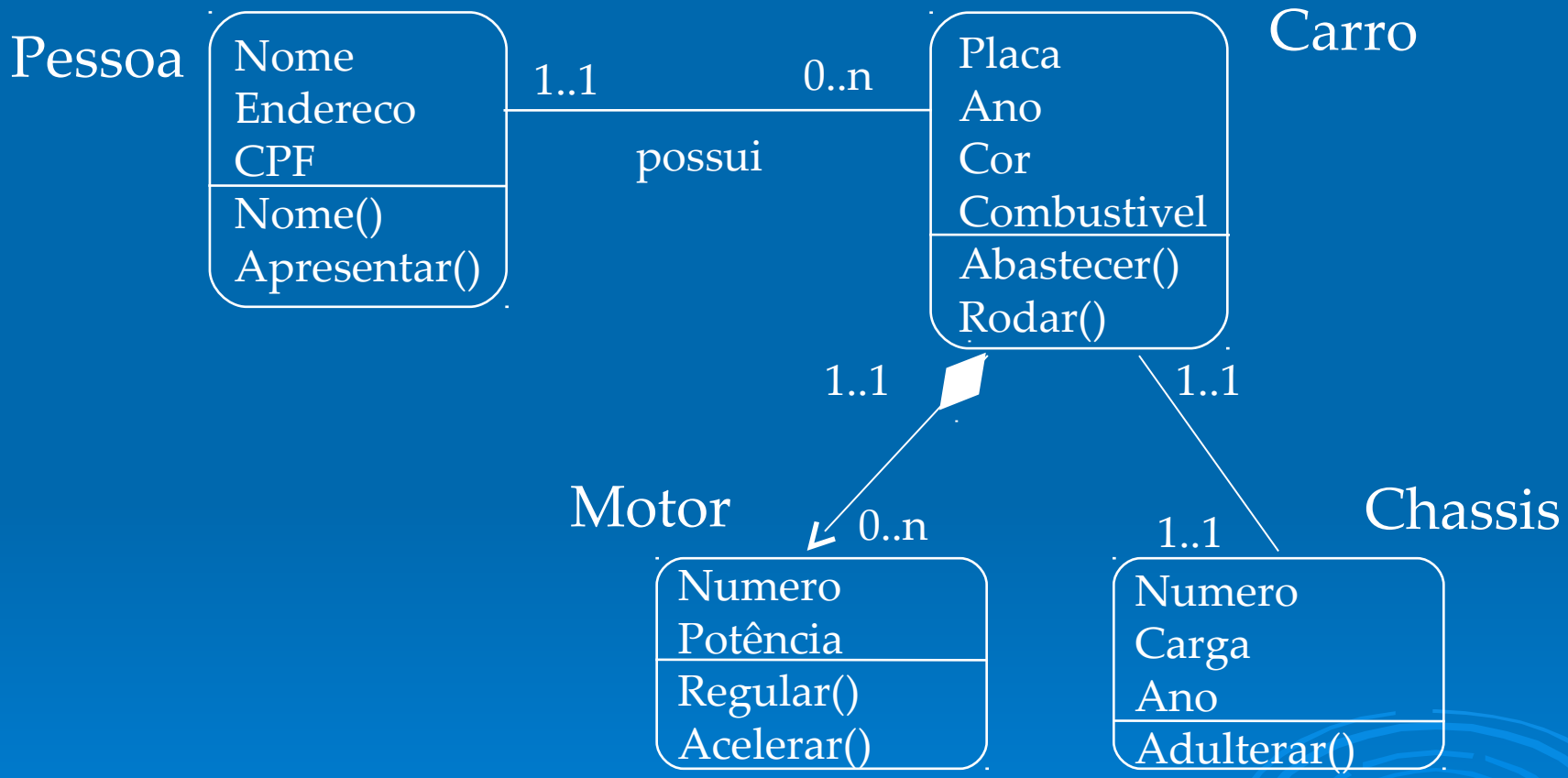
- Objeto Persistente
  - Objeto cuja existência transcende a execução de uma aplicação (programa).
  - Objetos armazenados em memória não volátil.
  - Banco de Dados Orientado a Objetos



Modelos de Representação

- Diagrama de Classes
  - Classes
  - Atributos
  - Métodos
  - Hierarquias de Generalização/Especialização
  - Agregações
  - Relacionamentos

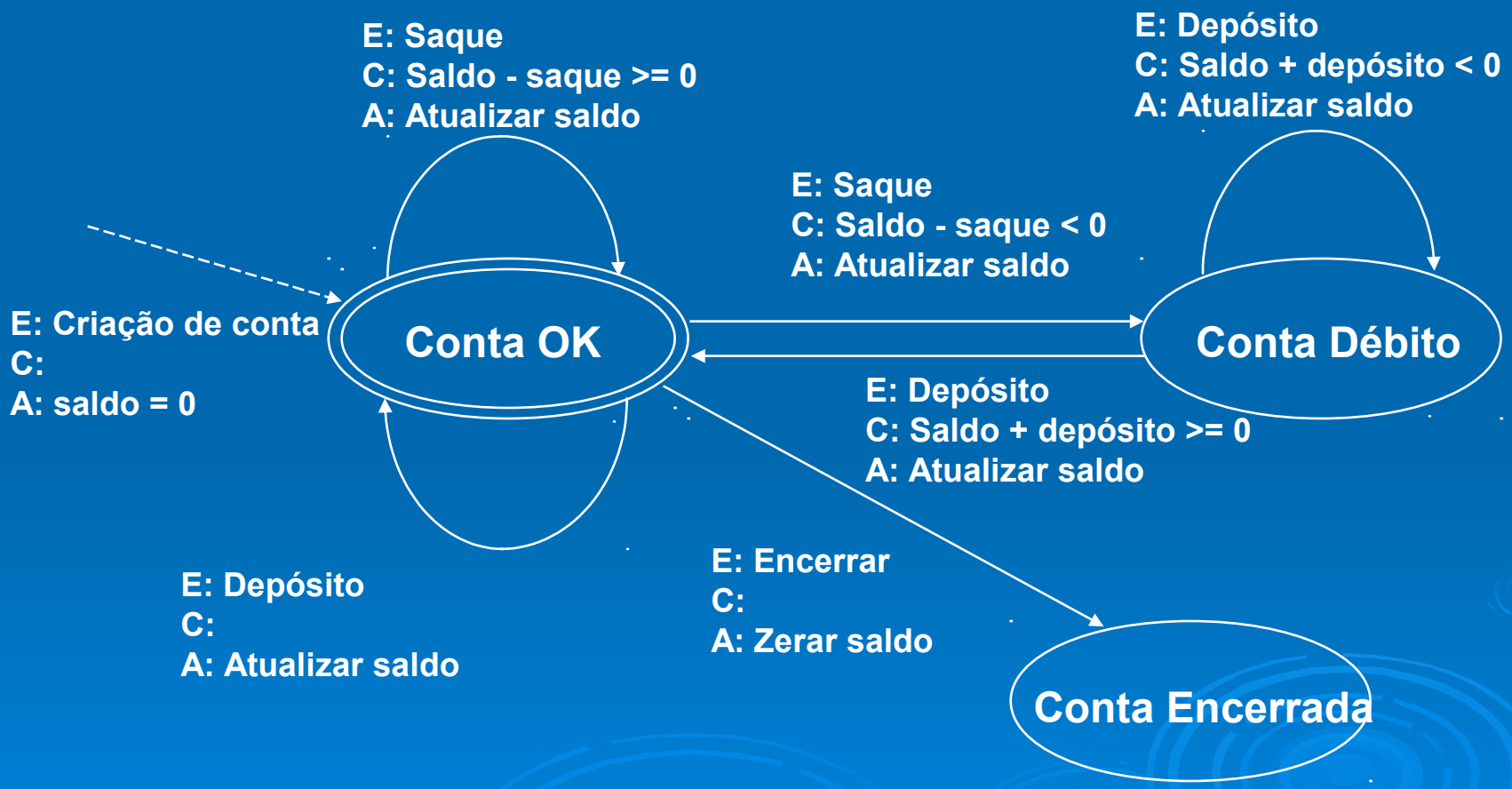
Modelos de Representação



Modelos de Representação

- Diagramas de Estados
  - 1 tipo de objeto (classe)
  - Estados
  - Eventos
  - Condições
  - Ações
  - Transições de Estado

Modelos de Representação



Identificadores de Objetos

- Abordagem Relacional
  - Chave Primária:
    - Baseada naturalmente nos dados da entidade

    Ou
    - Criação de chave artificial por tabela

- Abordagem OO
  - OID (Object Identifier):
    - Criada artificialmente para cada instância de objeto
      - Não possui relevância semântica para o negócio
    - Seu escopo é global (toda a base de dados)

Vantagens da Orientação a Objetos

- Uniformidade do modelo
- Alto nível de abstração
  - Classificação / Instanciação
  - Generalização / Especialização
  - Herança
  - Agregação
  - Tipos Genéricos

Vantagens da Orientação a Objetos

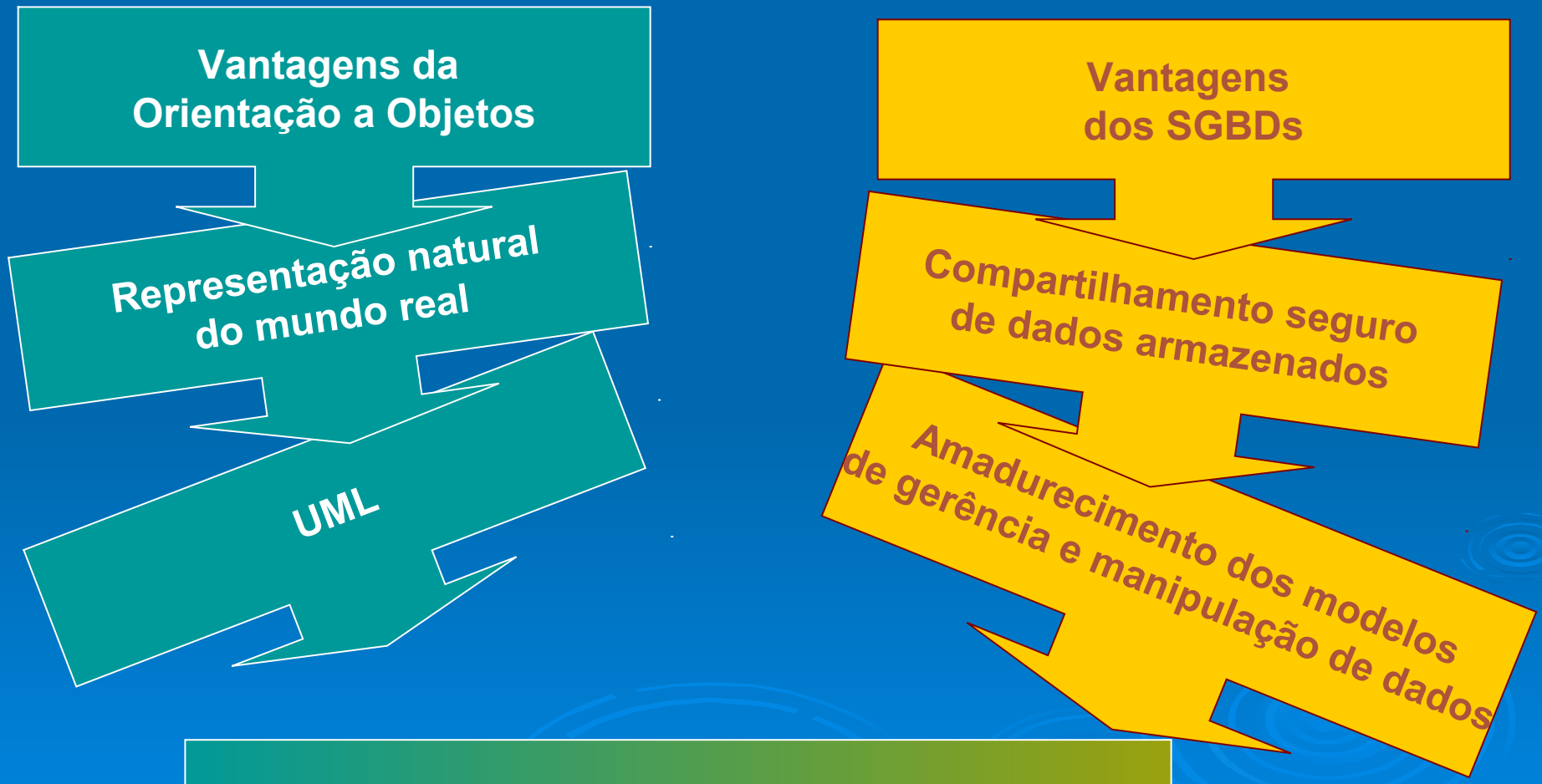
- Acesso controlado às informações
- Reutilização de especificações
- Proximidade com a realidade
  - Objetos como entidades ativas
  - Estrutura e Comportamento
  - Comunicação entre objetos

Aplicações

- Aplicação da O. O. em outras áreas
  - Bancos de Dados
  - Ferramentas CASE
  - Computer Aided Design
  - Interfaces Homem-Máquina
  - Sistemas Distribuídos

SGBD OO

- Motivação



Histórico de Evolução

- 1º Manifesto (ATKINSON, M. et al., 1989, 'The Object-Oriented Database System Manifesto" Proc. 1st DOOD Conf. Japão.)
  - The Golden rules

Complex objects - Object identity
Encapsulation - Types and Classes
Class or Type Hierarchies
Overloading and late binding
Computational completeness
Extensibility - Persistence
Secondary storage management
Concurrency - Recovery
Ad Hoc Query Facility

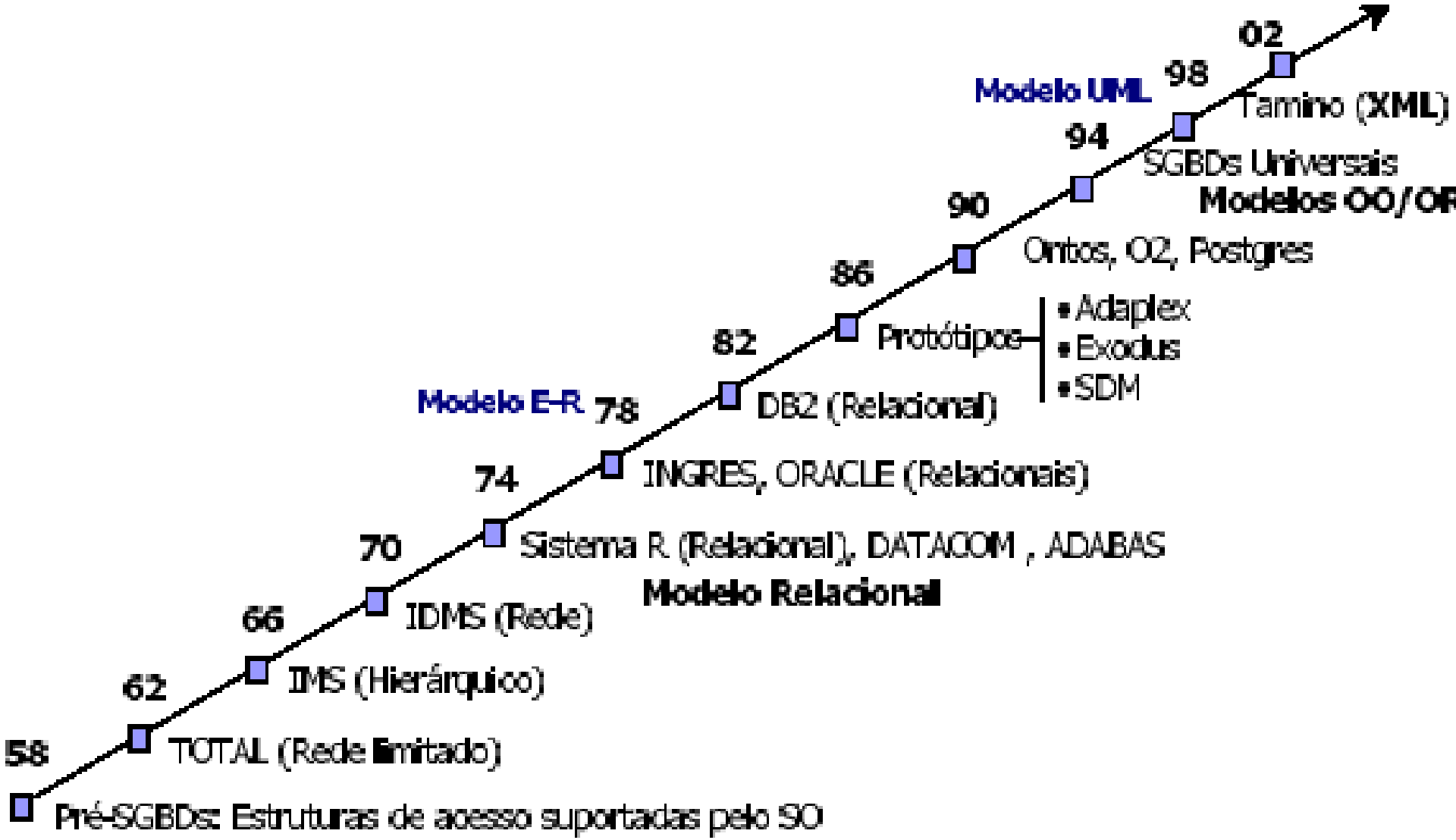
Histórico de Evolução

- 1º Manifesto (ATKINSON, M. et al., 1989, 'The Object-Oriented Database System Manifesto" Proc. 1st DOOD Conf. Japão.)
  - The Goodies
    - **Multiple inheritance**
    - **Type checking and type inferencing**
    - **Distribution**
    - **Design transactions**
    - **Versions**

Histórico de Evolução

- Contra-Manifesto (COMMITTEE for Advanced DBMS Function, 1990, “Third Generation Database System Manifesto” SIGMOD Record 19)
  - Relational-Object Databases
    - incorporar as regras mais ricas do conceito de orientação a objetos;
    - preocupação de como as funções deveriam ser escritas e como elas acessariam os dados;
    - preocupação em oferecer um padrão ser aberto e com a API.

Histórico de Evolução



SGBD OO

Caché

db4o

Javera

Jasmine - CA

JDBCStore

Jodad

Jevan - W3Apps

JYD

Objectivity

ObjectStore - eXcelon

UniObjects - Ardent

Poet

Versant

Vortex

O2

Padrões

- Entidades:
  - OMG
  - ODMG
  - OODBMS Task Group
  - ISO
  - ANSI
- Elementos da padronização
  - Object Request Broker
  - CORBA (Commom ORB Architecture)
  - Interface Definition Language (IDL - independente de SGBD ou LP)

ODMG – Object Database Management Group

- Formado pelos principais fabricantes de banco de dados OO e empresas interessadas na padronização.
- Trabalho em constante evolução
  - Padrão ODMG-93 (1.0)
  - Padrão ODMG-97 (2.0)
  - Padrão ODMG-2000 (3.0)
- http://www.odmg.org

ODMG – Object Database Management Group

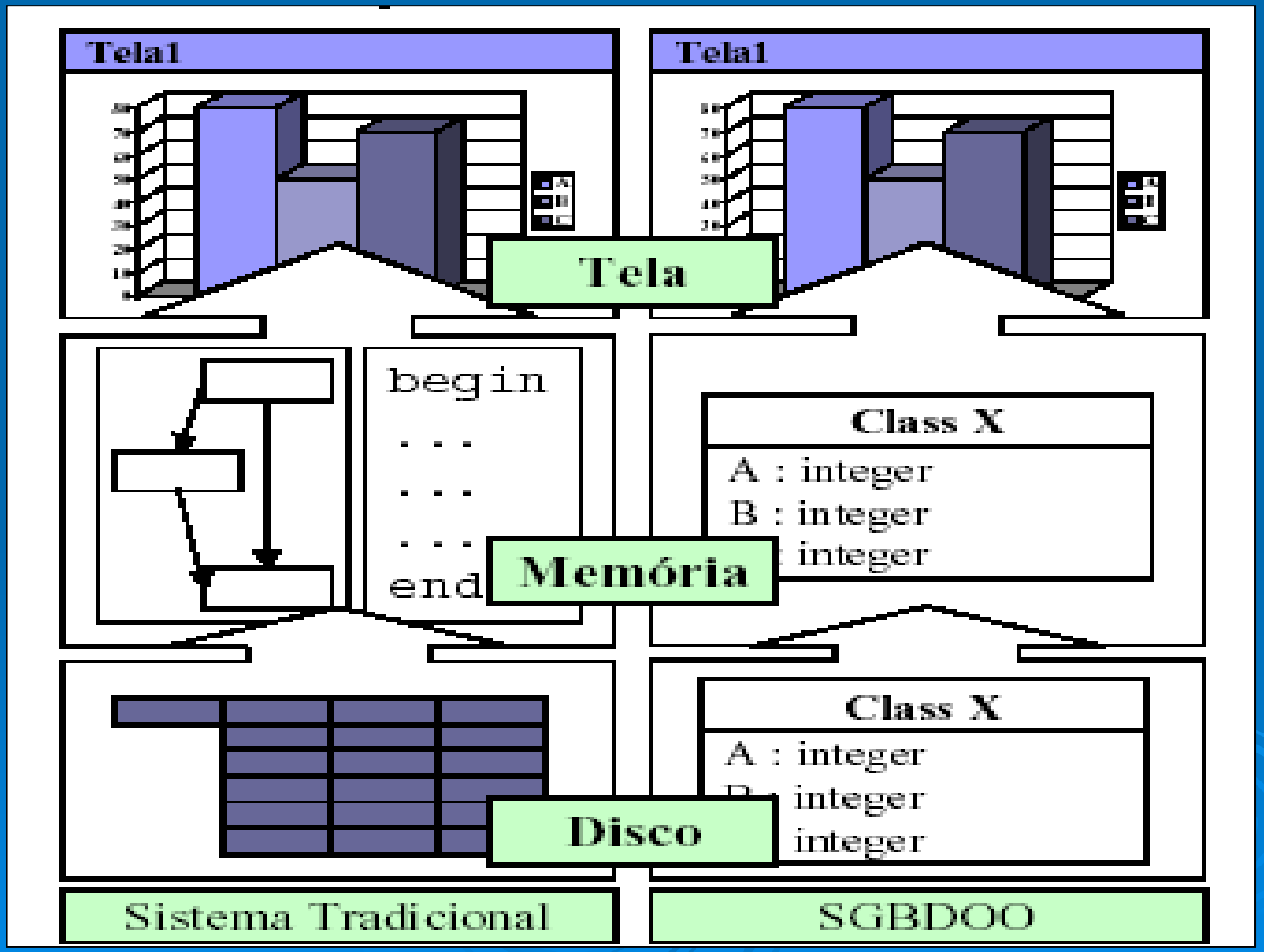
- Principais atividades
  - Desenvolvimento de uma arquitetura para SGBDOO
  - Compatibilizar os padrões OMG e ODMG
  - Modelo de Dados (que servirá como modelo lógico)
  - ODL (*Object Definition Language*) – Ling. de Definição de Objetos
  - OQL (*Object Query Language*) - Ling. de Consulta de Objetos
  - Associações com linguagens de programação como
    - C++
    - SmallTalk
    - Java (Java Binding (JDO))
  - Sintonia com outros padrões (SQL99)

Padronização

- Modelo Relacional e SQL
  - Independência do SGBD:
    portabilidade e interoperabilidade entre SGBDs (?)
- ODMG
  - Independência do SGBD

    +
  - Harmonia entre o modelo da LP e da LMD
    - Engloba os dados e operações da aplicação
  - Aplicações portáveis

Arquitetura



Padrões ODMG

- Modelo de Objetos
- Linguagem de Definição de Objetos - ODL
- Linguagem de Consulta - OQL
- Ligações com LPOO
- Metadados
- Controle de Concorrência e transações
- Controle de Armazenamento

Modelo de Objetos - ODL

- Uma interface (type) pode ter várias implementações (classes)
  - opcionais: extensão de classe, chaves
- Objetos (classes) x Literais (valores)
- Atributos
  - Simples (Atômico)
  - Coleções (set, bag, list, array, struct)
- Relacionamentos
  - herança múltipla (IS-A, EXTENDS)

Interface

- Nome
- SuperClasse
- Chaves
- Indicador de Persistência (indica se os objetos são persistentes ou transientes)
- Atributos da Classe
- Relacionamentos da Classe
- Interfaces de Operações (métodos)
- Exceções
- Constantes

Atributos

- Simples
  - Tipos básicos (integer, string, LOB´s, ...)
  - Tipos definidos pelo usuário
    - CPF, CEP ...
- Chave
- Complexos
  - Referência
  - Coleção
  - Derivados

Department
attribute short dno
attribute string name
attribute chair Professor
attribute components list [Professor]
...

Atributos Complexos

- Referências (relacionamento)
  - Não podem ser corrompidas. Invalidada automaticamente quando o objeto referenciado é apagado.
  - Independente dos valores do objeto referenciado (). (Identidade)
  - Analogia:
    - Ponteiros
    - Chaves estrangeiras

```
Department
        attribute short dno
        attribute string name
        attribute chair Professor
        attribute components list [Professor]
        ...
```

Atributos Complexos

- Coleções
  - Listas (Lists)
  - Conjuntos (Sets)
  - Vetores (Vectors)
  - Bags
- Violação da Primeira forma normal
- Possível ordenação entre os elementos

```
Class Person
        attribute string name
        attribute list<string> kidNames
```

Atributos Complexos

- Atributos derivados (Procedimento)
  - Preço total = Quantidade * Preço Unitário

Relacionamentos

- Nome
- Grau (binário, n-ário)
- Cardinalidade
  - 1 x 1
  - 1 x n
  - n x m
- Direção
  - uni
  - bi-direcional

Relacionamentos

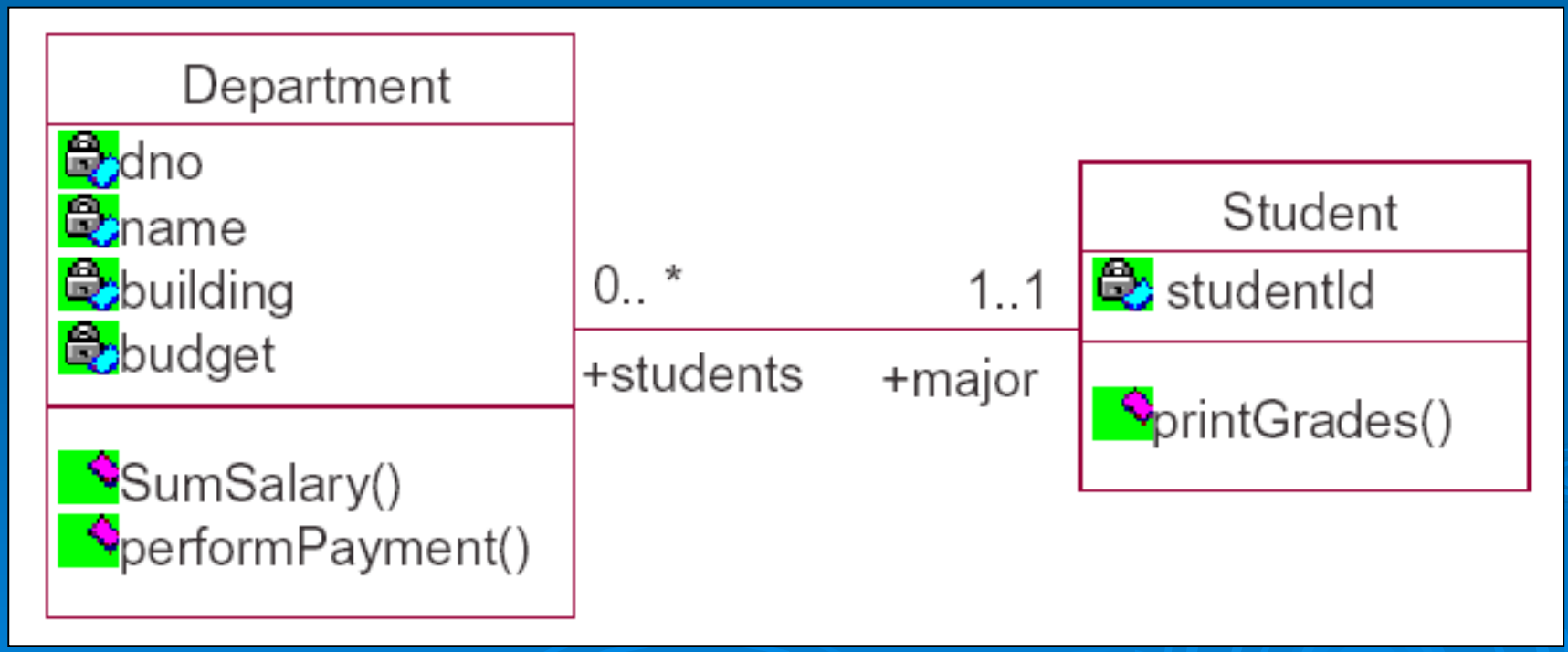
- Binária Unidirecional
  - atributo de referência
- Binária bi-direcional
  - relacionamento
- Binária com atributo, N-ária, Classe
  - classe relacionamento

Relacionamentos

- Binária Unidirecional
  - atributo de referência

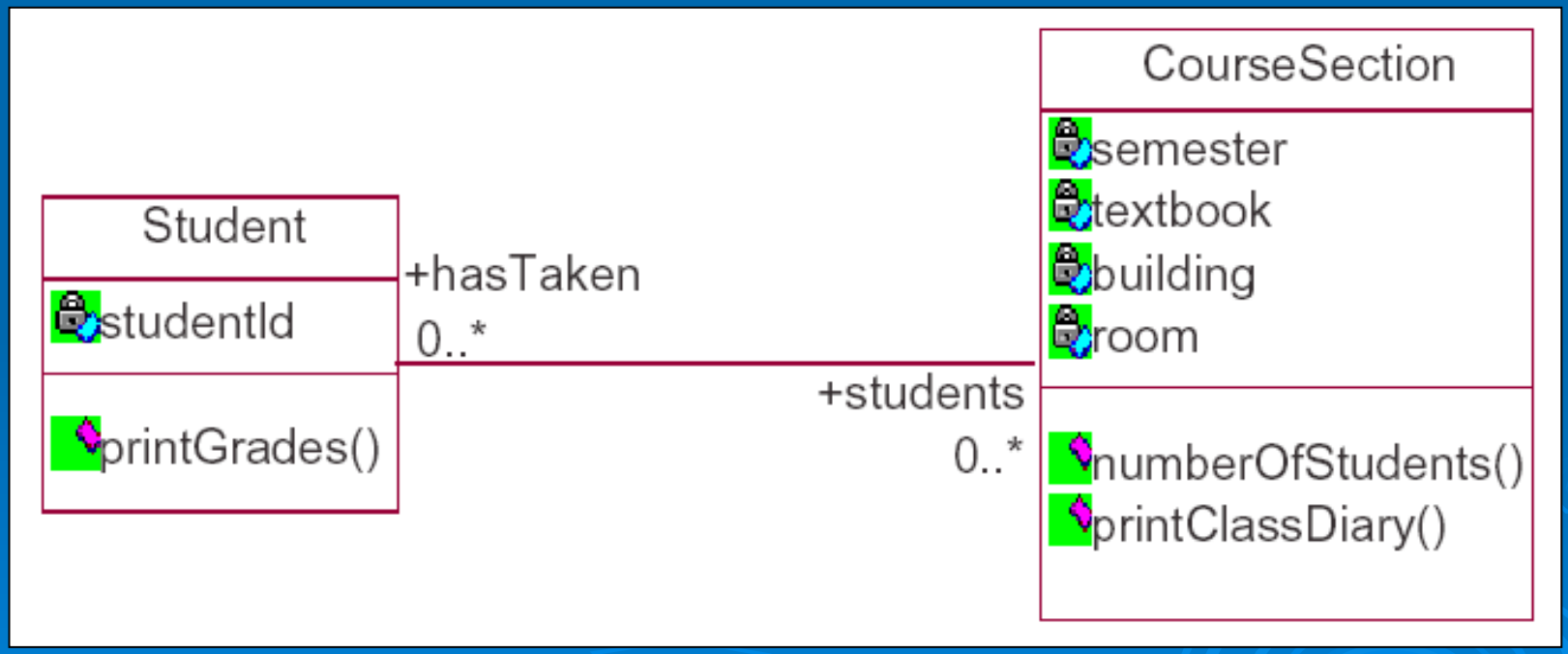
Relacionamentos

- Binária bi-direcional



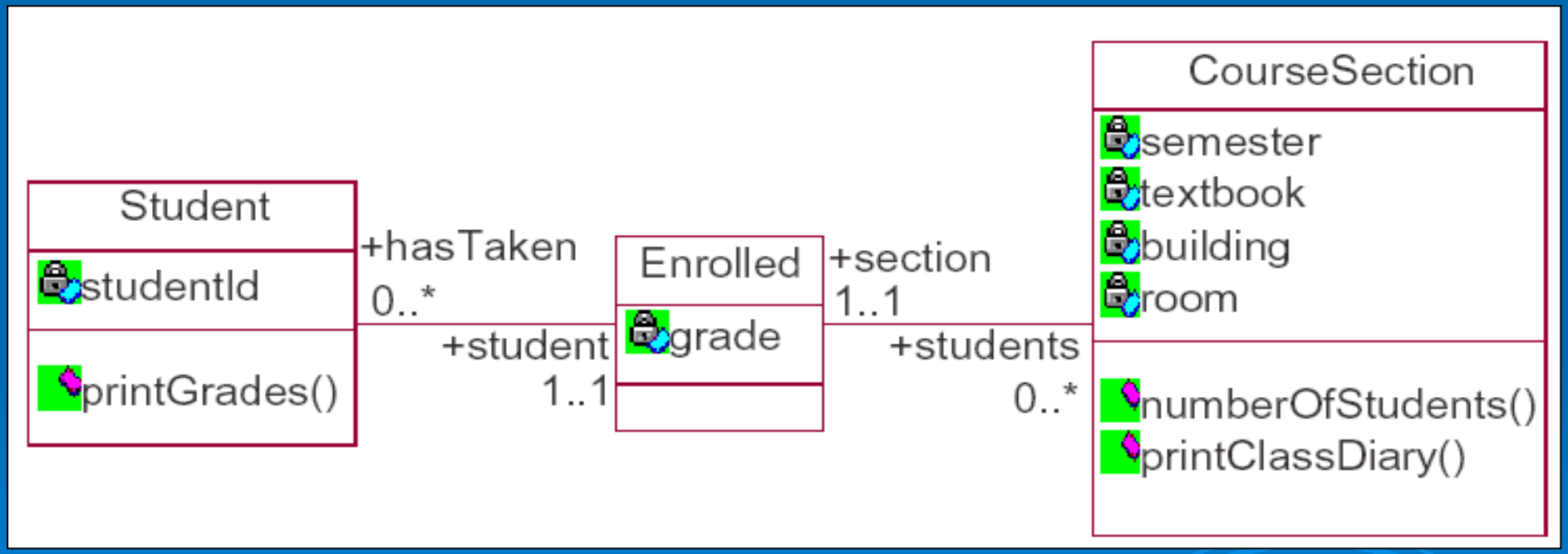
Relacionamentos

- Relacionamento Binário N x M



Relacionamentos

- Relacionamento Binário N x M com atributo



Relacionamentos

- Implementação em disco:
  - Co-alocação: armazenamento em seqüência
  - Indexação: índice *Hash* ou *B-tree*
  - Ligações: owner-member

Herança - ODMG

- Herança simples
- Herança simples com interface
- Herança Múltipla

Questões de implementação

- Armazenamento e recuperação de Objetos
  - Integridade dos dados entre as camadas
  - Armazenamento temporário de Objetos
  - Objetos compostos
  - Integração com linguagens de programação
  - Etc...
- Processamento de consultas, concorrência e recuperação
- Distribuição

Comportamento de Objetos

- Operações
- Exceções
- Transações
  - Requisitos ACID
  - *Controle pessimista de concorrência*
    - *Begin*
    - *Commit*
    - *Abort*
    - *CheckPoint*
    - *Abort-to-top-level*

Questões de implementação

- Armazenamento temporário de objetos
  - Páginas em buffer
  - Mapeamento e conversão de atributos
  - Processos distintos
  - Integridade dos dados entre as camadas
  - Informação em tempo de execução

Questões de implementação

- *Dispatching*
  - *"Dispatching"* é o processo de chamada a uma implementação particular de um método.
  - Como podemos ter vários métodos como o mesmo nome, é necessário especificar o objeto;
  - O método pode não estar diretamente associado ao objeto, pode estar por exemplo, no seu ancestral;

Questões de implementação

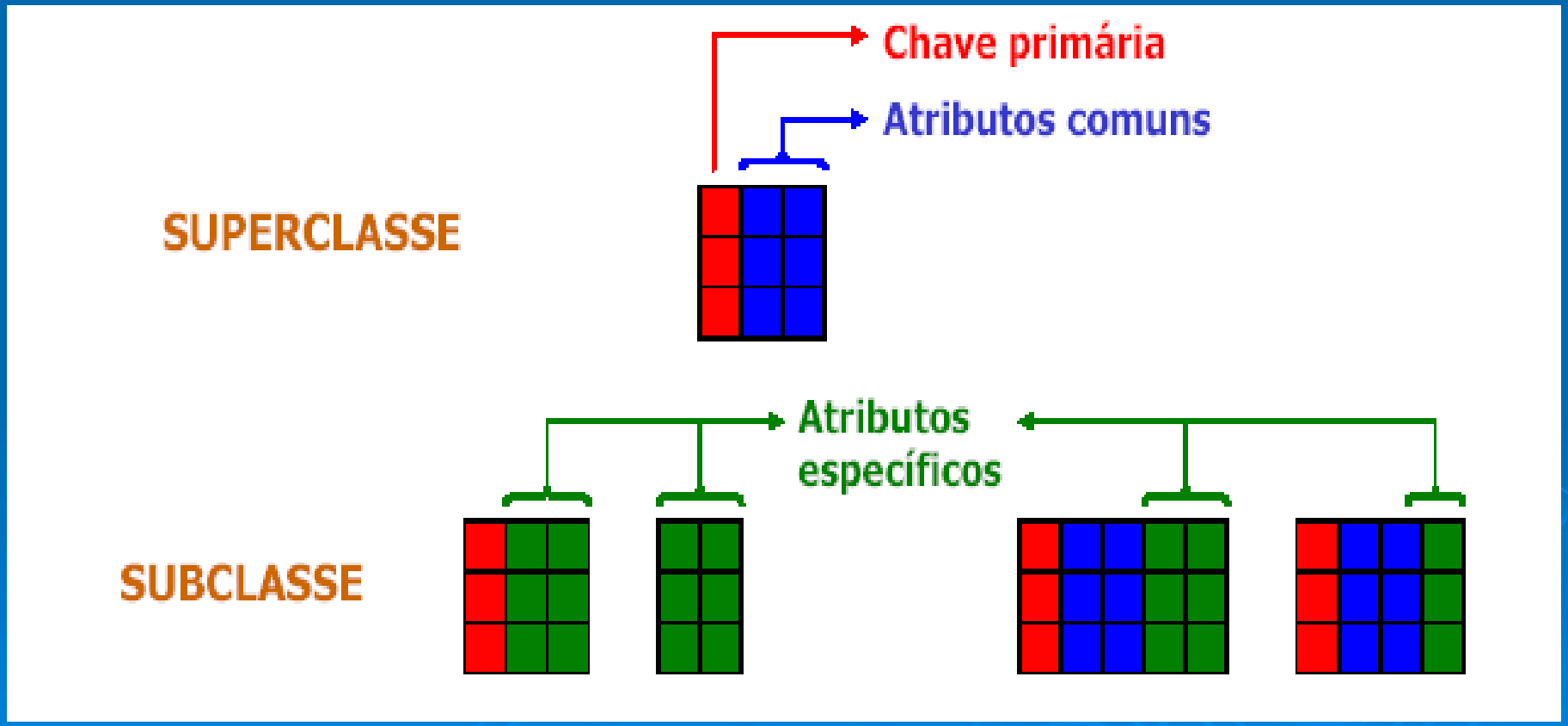
- Amarração Dinâmica (Dynamic Binding)
  - O tipo preciso ou semânticas de um símbolo pode não ser conhecido até o tempo de execução.
  - *Dispatching* em tempo real de métodos
  - Mapeamento em tempo de execução de nomes de atributos e variáveis a seus tipos e valores
    - Tipos ANY
  - Conveniência para o programador.
  - Torna possível escrever rotinas genéricas com maior variedade de tipos de dados como argumento

Questões de implementação

- Armazenamento x Agrupamento
  - Diversas abordagens:
    - Objetos compostos: segue o relacionamento de agregação
    - Referência: segue o relacionamento entre os objetos
    - Índice: por indexação de seus atributos
    - Customizado: em tempo de execução
  - Pode acontecer em dois níveis:
    - Paginação: menor unidade de leitura do disco
    - Segmentação: definição de um grupo lógico (tipo)

Questões de implementação

- Armazenamento



Armazenamento de Objetos - OID

- OIDs
  - Físicos
  - Lógicos
- Implementação e desempenho
  - Endereçamento Simples
    - Recuperação ↑ X Re-alocação ↓
  - Endereçamento Estruturado
    - acesso via paginação ↑
  - Substituto - *hash*
  - Substituto tipado - *hash*