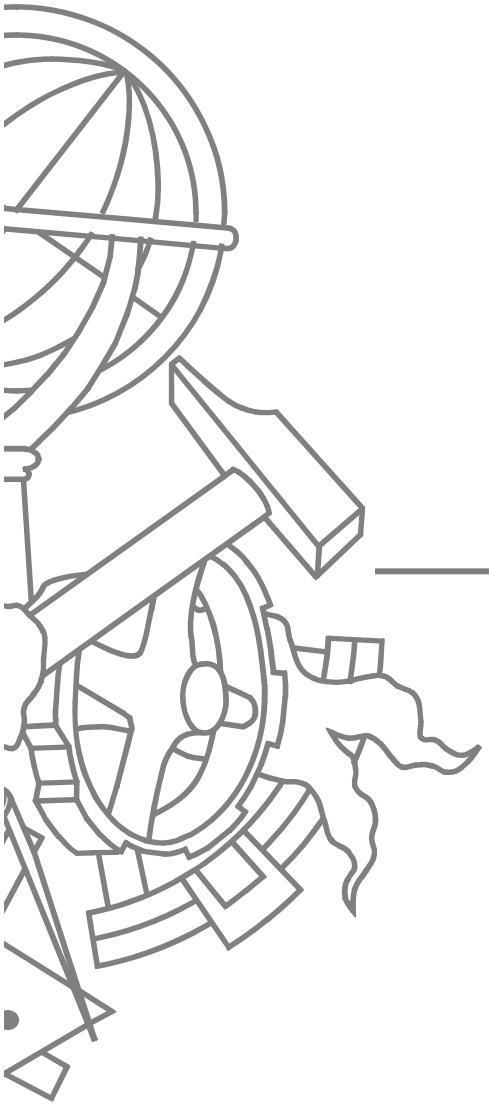
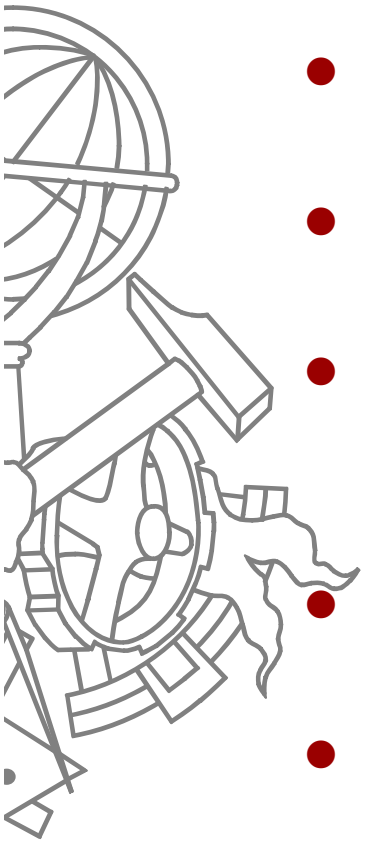


Entidades de Negócio

Parte 2.1

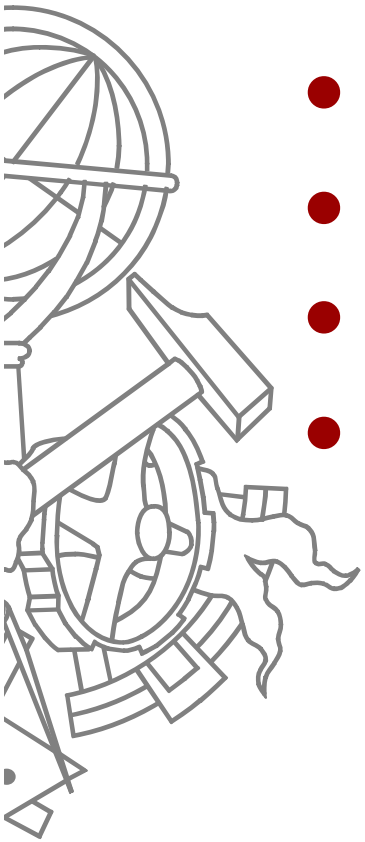
Como representar?



- Como representar uma entidade de negócio (ex., cliente)?
- Como representar colecções de entidades de negócio (ex. Carteira de clientes)?
- Como representar objectos complexos com hierarquias ou redes (ex., plano de produção)?

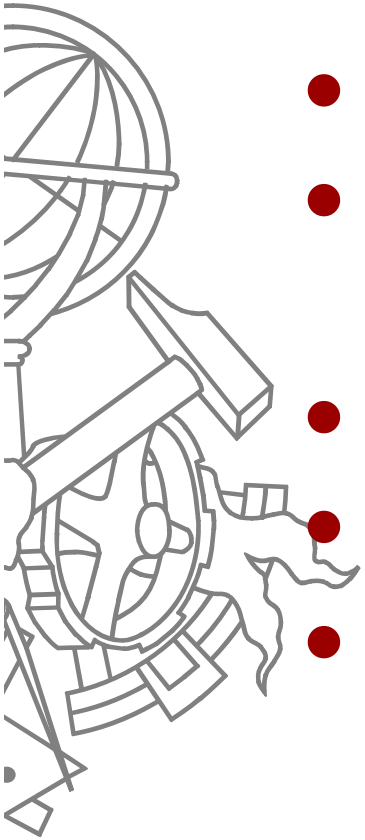
- Dados (estrutura) ou objectos (dados & comportamento)?
- Estrutura semelhante à BD ou mais adequada à representação conceptual?

Alternativas para uma entidade



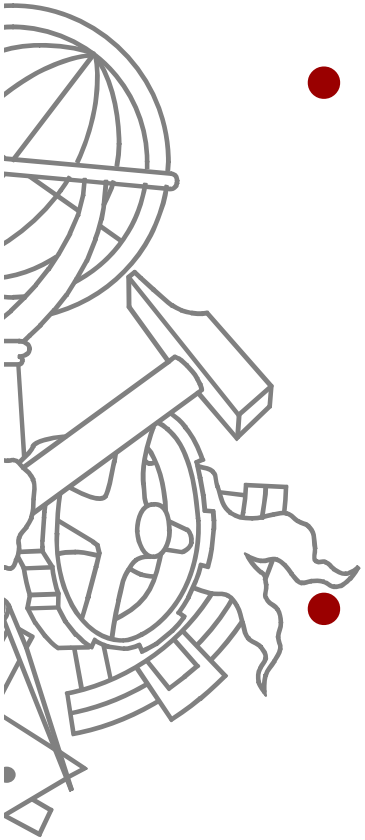
- *Custom classes*
- XML
- **`DataSet`** (.net)
- **`ResultSet`** (JDBC)

Alternativas para colecções e hierarquias



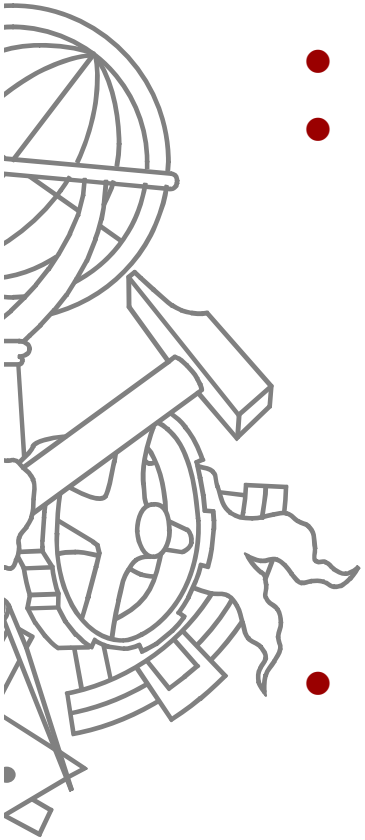
- *Custom classes*
- Suporte de cada plataforma (**`IList`** / **`List`**)
- XML
- **`DataSet`** (.net)
- **`ResultSet`** (JDBC)

Custom classes



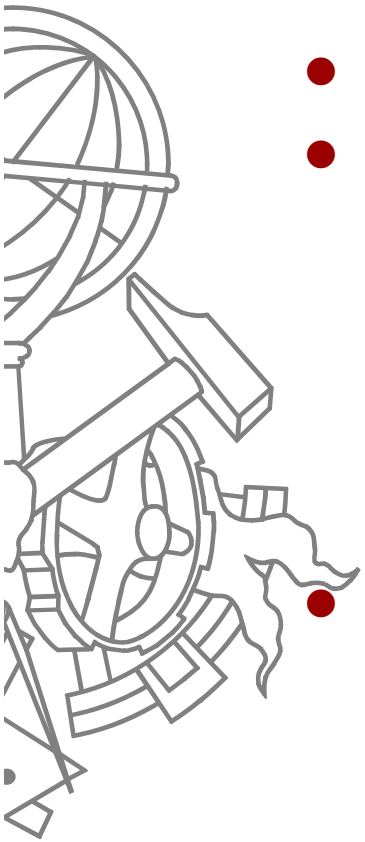
- Vantagens
  - Integração natural no programa (OO puro)
    - Tipos de dados
    - acesso explícito aos campos
  - Aproveita conhecimento actual dos programadores
- Desvantagens
  - Esforço de desenvolvimento
  - Ligação a controlos na UI
  - Mapeamento com SGBD Relacional

XML



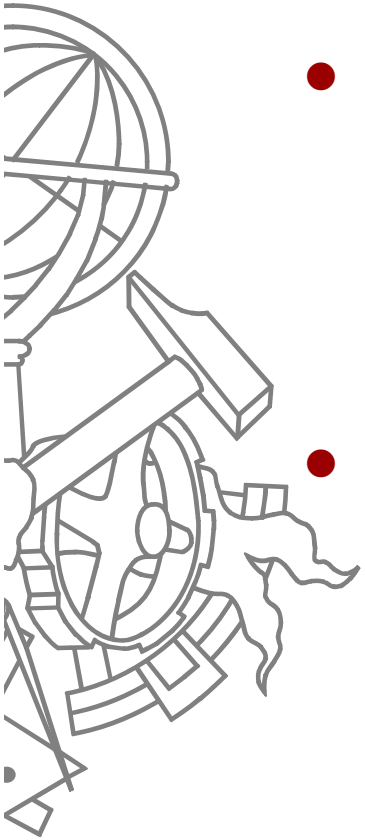
- *Strings* ou documentos
- Vantagens
  - Formato de texto facilmente exportado
  - Bom suporte em quase todas as plataformas
  - Suporta colecções e hierarquias de forma natural
  - *Standard* de facto
  - Permite a definição de esquemas externos (contratos) facilmente validáveis
  - Alguns motores de BD (ex., MS SQL Server e Oracle) já suportam a geração de XML nos comandos SELECT
- Desvantagens
  - Exige manipulação via API (ex., DOM)
    - Tudo são *strings*, exige *parse* para conversão
    - Lento
    - acesso aos campos não é explícito
  - Ligação a controlos na UI

DataSet (.net)



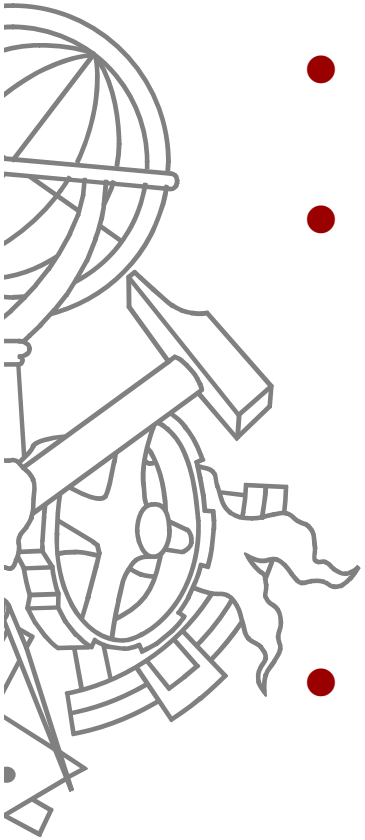
- *Typed* ou *untyped*
- Vantagens
  - Não exige desenvolvimento
  - Mimetiza modelo relacional
  - Integra muito bem com controlos UI
  - Permite representar colecções complexas e hierarquias (usando **`DataRelation`**)
- Desvantagens
  - Mimetiza modelo relacional
  - Exclusivo da plataforma .net
  - Manipulação via API
    - acesso aos campos não é explícito (*untyped*)

ResultSet (JDBC)



- Vantagens
  - Não exige desenvolvimento
  - Mimetiza modelo relacional (Tabela)
  - permite representar colecções lineares de um único tipo
- Desvantagens
  - Mimetiza modelo relacional (Tabela)
  - Exclusivo da plataforma Java
  - Só suporta uma tabela (colecções simples)
  - Manipulação via API
    - acesso aos campos não é explícito

IList



- Qualquer classe base da plataforma para colecções (ex., **`ArrayList`**)
- Vantagens
  - Não exige desenvolvimento para representar colecções
  - Existem algoritmos standard e implementações desses algoritmos para operações sobre listas (ex., ordenação)
  - Fácil para colecções lineares (ex., ArrayList)
- Desvantagens
  - Não é fortemente tipada (enquanto não existirem genéricos)
  - Complicado para representar hierarquias