**Fundamentos da Orientação Objeto**

**Paradigmas, Pilares e outros elementos**

* Conceito de POO
* Pacotes e visibilidade de recursos
* Classes e Construtores
* Java Beans e UML
* Pilares do POO
* Enums e Interfaces

**Conceitos de POO**

Linguagem de baixo nível = mais próximas das máquinas, não se aproxima da humana, exemplos: Assembly e C

Alto nível = São mais próximas a interpretação humana. Exemplo Java, Javascript, Python e C++

Programação estruturada é um paradigma de programação é feita com uma sequência de procedimentos.

POO é um paradigma que representa um objeto, feita uma abstração de algo que é real.

**Classes**

Identificador (identity), características (states), comportamento (behavior) e instanciar (new).Diagrama

Descrição gerada automaticamente com confiança média

Class nada mais é do que você criar um modelo ou um gabarito para um objeto.

O objeto nada mais é do que o produto do gabarito instanciado, ou seja, para se criar um objeto deve-se seguir o processo

1. Cria-se a classe (modelo ou gabarito)
2. Coloca-se na classe os atributos (características ou qualidades do objeto)
3. Com a classe criada nos instanciamos o objeto com o new (novo)

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Convenções e Classificações**

**Modelos (model):**  estrutura de domínio

**Serviços (service):**  regras de negócios e validações

**Repositório (repository):** integração com o banco de dados

**Utilitária (útil):** recursos comuns a toda a aplicação

Diagrama

Descrição gerada automaticamente

**Pacotes (package)**

Faz parte da organização do trabalho.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Sabemos, então que nosso projeto estará organizado na seguinte estrutura:

1. Tipo de projeto (se comercial (com), governamental (gov) ou código aberto (org)
2. Depois vem o nome da empresa ou desenvolvedor do projeto
3. Tipo da Classe ( se model, repository, service etc.)
4. Nome da classe

Exemplo: **com.nomeEmpresa.model.Usuario**