# Capítulo 1

## 

A partir de agora você irá desbravar o mundo da Programação Orientada a Objetos (POO). Este é um mundo totalmente novo e com diversos desafios e nele você terá total liberdade para criar. POO é uma área de programação vasta, e com ela poderemos criar qualquer tipo de software. Qualquer mesmo!

Você gostaria de criar um programa para aumentar os lucros da padaria do seu vizinho? Quer criar um game de última geração? Quer criar um software que controle seu Drone? Quer deixar seu carro computadorizado? Gostaria de criar aplicativos mobile e se tornar milionário? Com POO você pode criar estas e outras inúmeras coisas, pois POO não tem limites.

## Aula 1 – Introdução

### - Para quem é este livro

Você conhece alguma linguagem de programação orientada a objetos? Que tal programar em linguagem POO?

Mas afinal o que é POO? POO, ou Programação Orientada a Objetos, em uma definição básica, é um padrão, ou como costumam chamar, um paradigma de programação que permite que representemos elementos de código através de formas, objetos e eventos do dia-a-dia do nosso mundo real. Simples?

Se você nunca programou ao menos cinco linhas de código em qualquer linguagem de programação isto pode parecer tão impressionante quanto complicado. Mas não se preocupe, este curso é para você. Você irá compreender tudo, durante as aulas poderá encontrar termos desconhecidos como classes de objetos mas vamos aprender tudo ao longo do caminho.

Neste livro serão abordados conceitos do básico ao avançado em orientação a objetos.

Não será necessário ter conhecimento prévio em nenhuma linguagem de programação pois você aprenderá o essencial utilizando a linguagem **Java** para seguir o curso. **Java** é uma das linguagens orientada a objetos mais utilizadas e fáceis de aprender. É bastante difundida e possui uma comunidade grande de suporte na internet, inclusive de brasileiros.

Devido a velocidade do surgimento de novos conceitos a cada dia, algumas soluções de problemas não são encontradas nos livros de Java. Mediante isso, podemos acessar fóruns como GUJ, Devmedia, Javafree e o famoso StackOverflow para sanar algumas dúvidas específicas.

### 1.2 - Paradigmas de programação

Mas quais são estes paradigmas de programação? Existem dois tipos de programação bem usuais:

* orientada a objetos
* estruturada (ou procedural)

Estes tipos são chamados de paradigmas. Ambas não passam de formas diferentes de se representar os dados e tem suas vantagens em certas aplicações. Vejamos:

* **POO :** Em resumo, a programação orientada a objetos tende a agregar propriedades e funções de acordo com suas responsabilidades e esta agregação pode ser representada por um objeto. Por exemplo: um Usuário do sistema tem nome, idade, CPF (propriedades) e ele pode cadastrar, editar e deletar dados do sistema (funções). Então a maneira como modelamos e programamos em orientação a objeto permite esta abstração para objetos do mundo real (Usuário seria o objeto, neste caso).
* **Procedural**: Programação procedural é uma forma de programação de computadores que preconiza, ou seja, aconselha que todos os programas possíveis podem ser reduzidos a apenas três estruturas: sequência, decisão e iteração.
  + Na **sequência** são implementados os passos de processamento necessários para descrever determinada funcionalidade. Um exemplo básico seria um fluxograma, onde primeiro é executado a Etapa 1 e após a sua conclusão a Etapa 2 é executada, e assim por diante.
  + Na **decisão** o fluxo a ser percorrido depende de uma escolha.
  + Na **iteração** é permitida a execução de instruções de forma repetida, onde ao fim de cada execução uma condição é reavaliada e enquanto esta seja verdadeira, a execução de parte do programa continua.

Simples? Depois que se acostuma sim. Mas não se preocupe com esta tal linguagem procedural. No decorrer do curso, esta diferença entre os dois paradigmas se fará nítida e óbvia.

### 1.3 - Histórico

O conceito de programação orientada a objetos começou a surgir entre o início da década de 50 e fim dos anos 60 no Instituto de Tecnologia de Massachusetts (MIT) em grupos de pesquisa sobre inteligência virtual onde um “objeto” faria referência a itens LISP (uma linguagem de programação bem antiga) que continham propriedades específicas e únicas entre os mesmos.

É importante lembrar que os computadores pessoais, conhecidos como PCs, nem existiam nesta época, já que foram lançados no começo da década de 70. Nesta época o paradigma de linguagem era a linguagem procedural e os computadores em si ainda eram os gigantes mainframes (computadores de grande porte). Com isso podemos imaginar que existiam grupos de cientistas criando cartões perfurados ou fitas magnéticas para representar “coisas materiais”.

Vamos falar agora de algumas linguagens que deram origem a POO como conhecemos hoje.

*DICA: “Nos primeiros computadores, que eram máquinas enormes e muito complicadas de serem utilizadas, os cartões perfurados eram o meio de incluir dados e comandos nas máquinas. Até bem recentemente, alguns sistemas utilizavam este tipo de equipamento. No decorrer da década de 60 os cartões perfurados foram sendo substituídos gradativamente pelas fitas magnéticas”*

#### 1.3.1 - SIMULA

O conceito formal de programação a objetos foi introduzido somente na metade dos anos 60 com o SIMULA 67, uma revisão do SIMULA I, que por sua vez, é uma linguagem de programação desenvolvida para simular eventos discretos. Vamos tentar entender o que são simulações de eventos discretos.

* Modelos contínuos: são modelos onde o estado do sistema tende a mudar continuamente em função do tempo, por exemplo: crescimento populacional, circuitos eletrônicos, reações químicas, modelos econométricos
* Modelos discretos: são modelos onde o estado do sistema muda somente no instante em que ocorre um evento. Para todos os demais instantes de tempo o sistema não muda, por exemplo: situações de fila, em que clientes chegam, aguardam em fila se necessário e então recebem atendimento antes de deixar o sistema. Com isso, existem apenas dois eventos que controlam a simulação: chegadas e atendimentos.

Qualquer projeto de sistemas de eventos discretos (como os sistemas automatizados de manufatura – que na prática são filas de tarefas programadas) podem e devem ser simulados para que se consiga determinar as melhores condições de operação e detectar erros de projeto antes que o sistema seja implantado com estes erros. O SIMULA 67 influenciou a criação de muitas outras linguagens como o SmallTalk, derivados do LISP, Object Pascal e C++.

*DICA:“Antigamente a computação era usada por físicos e matemáticos, por isso hoje a maioria dos programadores não saberão o que é simulação de eventos discretos, a não ser que sejam físicos ou matemáticos.”*

#### 1.3.2 - Alan Kay

Na década de 70 a empresa XEROX PARC era uma potência em Ciência da computação. Neste tempo estava em desenvolvimento o projeto Dynabook, que era um ambiente computacional “voltado para crianças de todas as idades” com objetivo educacional. A ideia é bem parecida com um notebook/tablet que temos hoje.

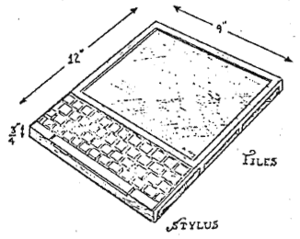


Figura 1. - Projeto Dynabook

Deste projeto surgiram:

* Interface Gráfica (com bitmaps)
* Sistemas de Janelas
* Mouse
* Redes via Modem e via Ethernet
* Placas de Som
* etc., etc.

Alan Kay, programador de SIMULA e então pesquisador e desenvolvedor do projeto Dynabook, propõe a criação de novos aplicativos para o tal computador. Ele foi então convidado a entrar no projeto e criar uma nova linguagem de programação que fosse fácil de programar e que tivesse a sintaxe muito simples. No final de 1972 a primeira versão de Smalltalk estava pronta e ainda não possuía conceito de herança (componente fundamental de POO que explicaremos nos capítulos seguintes), que foi introduzido na linguagem somente em 1976.

Alan Kay observou que o conceito de objetos tinha enorme potencial como uma ferramenta cognitiva: havia uma boa correspondência com a maneira de pensar das pessoas sobre o mundo. Ele percebeu que um substantivo isolado faz aparecer uma imagem concreta na mente das pessoas e um verbo isolado não tem esse efeito.

*DICA: “Verbos são propriedades dos substantivos: Garotos correm; cachorro corre; água corre; trem corre. Na orientação a objetos chamamos isso de polimorfismo. “*

#### 1.3.3 - SmallTalk

A orientação a objetos foi mais bem conceituada no laboratório da Xerox, em Palo Alto, sendo refinada numa sequência de protótipos da linguagem Smalltalk, criada nos anos 70. O líder desse projeto, inspirado pelo SIMULA, foi Alan Curtis Kay, considerado um dos criadores do termo “programação orientada a objetos”.

A partir dos conceitos desse sistema, como também dos seus conhecimentos em Biologia e Matemática, Alan Kay formulou sua analogia “algébrico-biológica”. Ele lançou o postulado de que o computador ideal deveria funcionar como um organismo vivo, isto é, cada célula com um objetivo particular em prol de um objetivo coletivo.

*Dica: “Naquela época já se pensava em computação bioinspirada e hoje, os estudos avançados em inteligência artificial continuam se convergindo para a máquina imitando um ser biológico para atingir objetivos de forma mais autêntica, autônoma e performática, como os seres vivos fazem”.*

O nome SmallTalk vem da premissa de que para programar nesta linguagem deveria ser tão fácil quanto conversar sobre assuntos banais (small talk em inglês).

### 1.4 - Evolução das linguagens

A partir dos anos 80 foram surgindo diversas linguagens orientadas a objeto. Derivados do C, como C++ e Objective-C que são linguagens da mesma família, amplamente usadas, porém com diferenças entre si. Por um lado C++ aparenta ser uma adaptação da linguagem C para orientação a objetos, já Objective-C é uma linguagem bastante dinâmica e atende realmente os princípios de orientação a objetos.

Outras linguagens conhecidas:

* **Java**: Aplicativos Android, Desktop, embarcados (relógios, câmeras, torradeiras, etc.)
* **C#**: Jogos, modelos 3D, Aplicativos Windows Phone
* **Python**: Aplicações científicas, páginas web, servidores de aplicações
* **Ruby**: Aplicações científicas, páginas web, servidores de aplicações
* **PHP5**: servidores de aplicações web
* **Pearl**: servidores de aplicações web
* **Visual** Basic: Aplicativos desktop para Windows (já em desuso)
* Etc., etc.

### 1.6 - Vantagens de POO

Antigamente sites não passavam de páginas estáticas que exibiam certos conteúdos pré-definidos. Hoje em dia, sistemas web se tornam cada vez mais complexos. Sistemas de software que regem empresas inteiras tendem a migrar para o ambiente web.

Devido ao aumento tecnológico e popularidade da informatização, softwares tendem a se tornarem mais complexos e críticos (softwares que em caso de falhas podem prejudicar a integridade física ou mental de uma pessoa são considerados softwares críticos. Ex.: Controle de bordo de um avião). Surgem então as vantagens de utilização da POO:

* **Abstração:** Com a abstração, podemos nos concentrar apenas em aspectos essenciais do contexto ou domínio em que o software se aplica. Em POO, por exemplo, o conceito de *Classe* é uma abstração de “coisas” que fazem parte do contexto do software. Por exemplo, em um sistema que faz controle de usuários: podemos imaginas a abstração referente à classe *Usuário*. Nela ainda existem várias entidades como *Cliente, Consultor, Administrador, Moderador.* Estas são as entidades abstraídas da entidade *Usuário.*Explicaremos nos capítulos seguintes
* **Reuso**: Como a maneira em que se projeta uma aplicação utilizando POO possui menor acoplamento de código (linguagens estruturais geralmente não possui uma modularização de componentes de código tão precisa), ou seja, podemos criar espécies de *pluggins*, que são trechos de código que podemos adicionar ou retirar do software e ele continuará funcionando como um todo. Isto é chamado de reuso ou reutilização de código. A reutilização de código é a maior vantagem de POO porque permite que programas sejam escritos mais rapidamente.
* **Manutenção**: A organização que a POO permite faz com que a manutenção do produto seja menos custosa. Um projeto orientado a objeto tenta minimizar os requisitos de manutenção corretiva e preventiva do projeto que já esta sendo utilizado pelo usuário final.
* **Legibilidade**: Devido a grande popularidade da POO foram surgindo padrões e boas práticas para escrita de código. Legibilidade de código é a facilidade de se escrever um código seguindo alguns padrões ou notações, para se ter um código mais claro e fácil de se interpretar. Além do conceito da POO em facilitar muito na compreensão do código, estes padrões e convenções criados pela comunidade irão facilitar ainda mais.

### 1.7 - Desvantagem

A execução de uma aplicação orientada a objetos é mais lenta do que aquela que vemos na programação estruturada, por exemplo, como POO é mais abstrata no sentido de representar os dados com “formas e coisas” do mundo, a execução do programa terá uma complexidade muito maior e muito mais código para ser interpretado pelo computador, diferente da execução sequencial da programação estruturada, que se comporta de forma sequencial, decisória e iterativa (repetitiva). Com computadores potentes (hoje em dia um smartphone pode ser considerado um computador potente, pois tem processamento de 2 GHz em comparação com o computador que operou na missão Apollo 11, chamado AGC, que tinha apenas 0.043MHz de poder de processamento de dados) isto não chega a ser um problema, se for o caso do desenvolvimento de uma aplicação comum (web, mobile ou desktop), mas no caso de aplicações embarcadas, ou seja, softwares que rodam nos próprios dispositivos, que são pequenos e menos potentes, como relógios, painel digital de geladeira, instrumentos musicais digitais e etc, tem se optado por linguagem procedural como C, por exemplo.

### 1.8 - Exercícios

#### Lógica de programação

Se você nunca programou saiba que é preciso exercitar o cérebro, de forma que ele pense “logicamente”. Faremos alguns exercícios de lógica de programação utilizando pseudocódigo.

Pseudocódigo é uma modalidade de representação de algoritmos um pouco mais literal e semântica, quase próxima da linguagem natural do ser humano. É utilizado para prática de algoritmos e lógica de programação e o escrevemos como uma receita de bolo, ou instruções para realizar algo. Vejamos um exemplo de algoritmo com pseudocódigo:

Questão: Escreva um algoritmo que realize a soma de 1+2 e exiba o resultado.

Algoritmo Soma**;**

VAR

digito1, digito2, soma : INTEIRO;

INÍCIO

digito1 <- 1;

digito2 <- 2;

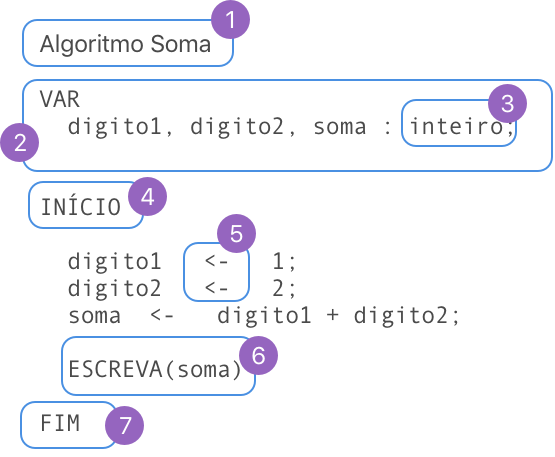
soma <- digito1 + digito2;

ESCREVA(soma);

FIM

Os ponto e vírgulas são características de algumas linguagens de programação como o Pascal, C++ e o Java, porém em outras linguagens de programação, este pode não aparecer no pseudocódigo, pois elas podem adotar outra semântica.

O pseudocódigo é bastante flexível. Basta criarmos um padrão e utilizarmos. Vamos explicar o algoritmo acima:



**Corpo do algoritmo**

Figura 1. - Explicação do algoritmo soma

1. Título do algoritmo. Geralmente mostra uma descrição breve e objetiva do que o algoritmo se trata, ou seja, o que ele fará.
2. Área para declaração de variáveis a serem utilizadas no decorrer do algoritmo.
3. Tipo que as variáveis irão aceitar. Possíveis valores: inteiro, booleano, texto, ou decimal.
4. Indica o início da execução do código.

O corpo do algoritmo está entre o INÍCIO e o FIM. É onde fica a lógica de programação.

1. “<-” atribui um valor a uma variável.
2. Exibe um dado.
3. Indica o final da execução do algoritmo.

#### Agora faça você mesmo

1. Faça um algoritmo que receba dois números e ao final mostre a soma, a subtração, a multiplicação e a divisão dos números lidos.
2. Escrever um algoritmo para determinar o consumo médio de um automóvel ao percorrer uma determinada distância. Será fornecida a distância total percorrida pelo automóvel e o total de combustível gasto.
3. Escrever um algoritmo que leia o nome de um vendedor, o seu salário fixo e o total de vendas efetuadas por ele no mês (em dinheiro). Sabendo que este vendedor ganha uma comissão de 15% do valor de seu salário para cada venda efetuada no mês, informar o seu nome, o salário fixo e salário com as comissões ao final do mês.
4. Escrever um algoritmo que leia o nome de um aluno e as notas das três provas que ele obteve no semestre. No final informar o nome do aluno e a sua média (aritmética).
5. Ler dois valores para as variáveis A e B, e efetuar as trocas dos valores de forma que a variável A passe a possuir o valor da variável B e a variável B passe a possuir o valor da variável A. Apresentar os valores trocados.
6. Ler uma temperatura em graus Celsius e apresentá-la convertida em graus Fahrenheit. A fórmula de conversão é: F=(9\*C+160) / 5, sendo F a temperatura em Fahrenheit e C a temperatura em Celsius.
7. Elaborar um algoritmo que efetue a apresentação do valor da conversão em real (R$) de um valor lido em dólar (US$). O algoritmo deverá solicitar o valor da cotação do dólar e também a quantidade de dólares que o usuário possui.
8. A Loja Mamão com Açúcar está vendendo seus produtos em 5 (cinco) prestações sem juros. Faça um algoritmo que receba um valor de uma compra efetuada pelo cliente e mostre o valor de cada um das prestações que ele pagará.
9. O custo ao consumidor de um carro novo é a soma do custo de fábrica com a percentagem do distribuidor e dos impostos (aplicados, primeiro os impostos sobre o custo de fábrica, e depois a percentagem do distribuidor sobre o resultado). Supondo que a percentagem do distribuidor seja de 28% e os impostos 45%. Escrever um algoritmo que leia o custo de fábrica de um carro e informe o custo ao consumidor do mesmo.
10. Faça um algoritmo que receba um número e diga se este número está no intervalo entre 100 e 200.
11. Escrever um algoritmo que leia o nome e as três notas obtidas por um aluno durante o semestre. Calcular a sua média (aritmética), informar o nome e sua menção: aprovado (media >= 7.0), reprovado (media <= 5.0) ou recuperação (média entre 5.1 a 6.9).
12. Ler 80 números e ao final informar quantos desses números estão no intervalo entre 10 (inclusive) e 150 (inclusive).
13. Faça um algoritmo que receba a idade de 75 pessoas e mostre uma mensagem informando "maior de idade" ou "menor de idade" para cada uma dessas pessoas. Considere a idade a partir de 18 anos como maior de idade.

### 1.9 - TDP

O TDP (ou Trabalho de Desenvolvimento Prático) é um projeto prático onde você irá criar uma aplicação orientada a objetos completa, do início ao fim.

Este projeto não é simples programação. Você irá conhecer diversas etapas da concepção de um aplicativo. Desde o amadurecimento da ideia, passando pela fase de análise, modelagem, até a fase de desenvolvimento em si. Nas primeiras aulas, enquanto nossos conhecimentos em programação orientada a objetos não estiverem maduros, você irá participar da fase analítica do projeto, logo após isso, você irá passar para fase de programação, arquitetura e por aí em diante. Serão mostradas competências suficientes para que você possa criar um programa independente.

#### Definição inicial do trabalho

##### Você foi alocado ao projeto Filmes

Você é um programador iniciante em uma grande empresa de software. Esta empresa gosta de colocar seus novos funcionários à prova, para avaliá-los na capacidade de tomada de decisão. Você será avaliado.

Na hierarquia de uma empresa comum, sempre temos um Gerente de Projetos liderando um grupo de programadores. O trabalho de um gerente de projetos consistem também em coletar informações do cliente sobre o projeto, delegar atividades ao time e criar métricas para alocação de recursos (humanos) nos projetos futuros.

Certo dia você ouviu **Nelson**, seu gerente de projetos, conversando sobre um novo projeto com o diretor da empresa. Você não ouviu muito bem, mas conseguiu entender que se tratava de algo envolvendo filmes.

##### Nelson está doente. Precisa de um substituto

Nelson agendou uma reunião para a semana seguinte para conversar com os desenvolvedores da empresa sobre o novo projeto e sobre a situação de sua saúde. Ele vai precisar de um substituto.

Que tal pesquisar um pouco mais sobre o assunto e ganhar destaque aos olhos de Nelson? Quem sabe você não o substitui no seu período de licença médica?

#### Tarefa

Pesquise na internet, sites que falam sobre filmes. Tente procurar aplicativos (móveis ou desktop) que são relacionados a filmes.