**Programação Orientada a Objetos (POO)**

* Evolução da Programação:
* Criador da Programação Orientada a Objetos: Alan Kay (1970)
  + Trabalhou na Xerox
    - Criou o Dynabook:
      * Conceito base dos Smartphones de teclado (aparelho movel com aparência de uma tela de computador na parte superior e um teclado na parte inferior)
      * Conceito base dos Notebooks atuais (uma tela de computador na parte superior e um teclado na parte inferior)
    - Criador da Linguagem de Programação Smalltalk (Primeira Linguagem de Programação Orientada a Objetos)
  + Trabalhou na Apple
  + Trabalhou numa área de divisão e criatividade da Disney
  + Trabalhou na HP
* Como era a Programação antes da POO:
  + Dados Globais: massa de dados
  + Procedimentos: tratam a massa de dados
    - acessam essa massa, processam essa massa e geram os resultados
    - os procedimentos não faziam uso de toda a massa de dados, somente das partes úteis. Isso torna o processo muito desorganizado.
* Como ficou a Programação depois da POO:
  + Trabalha com partes menores, os objetos. Sendo assim, temos:
    - Dados de Objetos: “pedaços” da massa de dados
  + Métodos: tratam esses “pedaços” (os Dados de Objetos)
    - acessam apenas esses “pedaços” (Dados de Objetos), processam eles e geram os resultados
    - Os Dados de Objetos podem se relacionar entre si, pois é possível ter troca de dados entre eles.
* Algumas linguagens baseadas em POO:
  + C++
  + Java
  + PHP
  + Python
  + Ruby
  + C#
  + Visual Basic
  + Objective-C
  + Swift
* Vantagens de se programar em POO:
  + **C.O.M.E.R.N.**ada
    - **C**onfiável: O isolamento entre as partes gera software seguro. Ao alterar uma parte, nenhuma outra é afetada;
    - **O**portuno: Ao se dividir tudo em partes, várias delas podem ser desenvolvidas em paralelo;
    - **M**anutenível: Atualizar um software é mais fácil. Uma pequena modificação vai beneficiar todas as partes que usarem o objeto;
    - **E**xtensível: O software não é estático. Ele deve crescer para permanecer útil;
    - **R**eutilizável: Podemos utilizar um objeto de um sistema que criamos em outro sistema futuro;
    - **N**atural: Mais fácil de entender. Você se preocupa mais na funcionalidade do que nos detalhes de implementação;