uninorte

PROGRAMAÇÃO ORIENTADA A OBJETOS

Prof. Edkallenn

Sistemas de Informação - Uninorte

Programação Orientada a Objetos

- **Prof. Edkallenn Lima**
- edkallenn@yahoo.com.br (somente para dúvidas)
- **Blogs**:
  - http://professored.wordpress.com (Computador de Papel – O conteúdo da forma)
  - http://professored.tumblr.com/ (Pensamentos Incompletos)
  - http://umcientistaporquinzena.tumblr.com/ (Um cientista por quinzena)
  - http://eulinoslivros.tumblr.com/ (Eu li nos livros)
  - http://linabiblia.tumblr.com/ (Eu li na Bíblia)
- **YouTube**:
  - https://www.youtube.com/user/edkallenn
  - https://www.youtube.com/channel/UC-pD2gnahhxUDVuTAA0DHoQ
- **Redes Sociais**:
  - http://www.facebook.com/edkallenn
  - http://twitter.com/edkallenn ou @edkallenn
  - https://plus.google.com/u/0/113248995006035389558/posts
  - Instagram: http://instagram.com/edkallenn ou @edkallenn
  - Foursquare: https://pt.foursquare.com/edkallenn
  - Pinterest: https://br.pinterest.com/edkallenn/
- **Telefones**:
  - 68 98401-2103 (CLARO e Whatsapp) e 68 3212-1211.

Os exercícios devem ser enviados SEMPRE para o e-mail:
edkevan@gmail.com
ou para o e-mail:
edkallenn.lima@uninorteac.edu.br

Agenda

- Classes e Objetos
- Encapsulamento (ideia inicial)
- Atributos
- Métodos
- Troca de mensagens

Orientação a Objetos

- Na Orientação a Objetos o software é visto como uma coleção de unidades, chamadas de objetos
- Cada um dos objetos é capaz de realizar as ações relacionadas a si próprias e de solicitar ações a outros objetos
- Juntos esses objetos interagem para solucionar o problema em questão

Classe

- Uma classe, nada mais é do que a descrição de um conjunto de entidades (reais ou abstratas) do mesmo tipo e com as mesmas características e comportamentos.
- As classes definem a estrutura e o comportamento dos objetos daquele determinado tipo.
- Podemos dizer que as classes são, na verdade, modelos de objetos do mesmo tipo
- Imaginar a classe "Carro" do mundo real e trazer para o mundo virtual.
- Um carro tem rodas, pneus, lanternas, portas, motor, parachoques, etc
- O carro anda, freia, acelera, etc.

Classes

- Qual é o modelo do carro? Qual é a cor do carro? Qual é a marca dos pneus?
- As lanternas estão ligadas? O motor do carro está funcionando? **Não sabemos!**
- Na verdade, não é possível responder essas perguntas, pois ao pensar na classe "Carro", nós definimos apenas o que o carro pode ter e o que ele faz, e não o que realmente ele é.
- É uma abstração da realidade

Classes

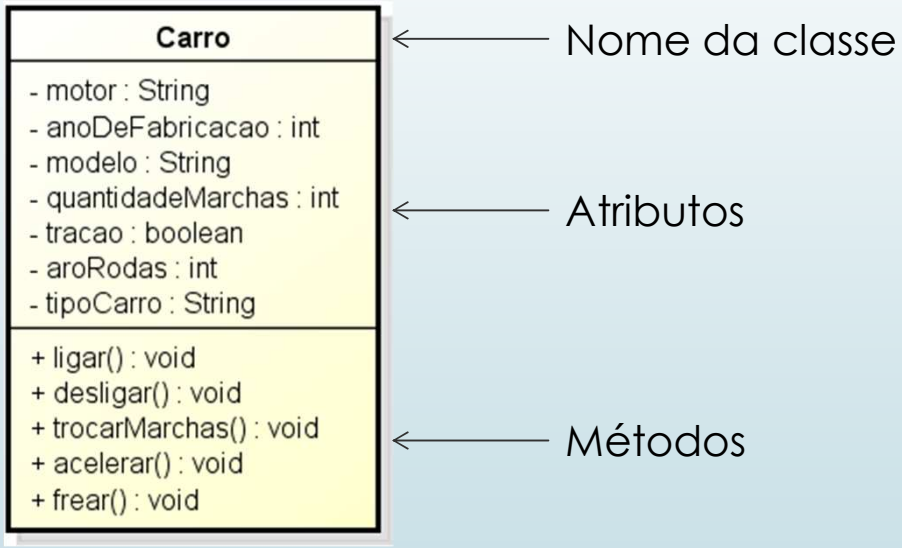
- Para ficar mais claro, pense como se não existissem carros no mundo. Você é o grande inventor dos carros! Como qualquer invenção, você precisará fazer algum desenho, escrever um rascunho, em esboço, criar algum projeto ou apenas formar uma ideia em sua cabeça de como "a coisa" deve funcionar.
- Isso são as classes! Ideias, modelos, protótipos, rascunhos... algo que ainda não existe, mas que, a partir da ideia, alguém pode começar a construir o produto final (que existe de verdade)

Classes

- Ao criar a classe "Carro", precisamos pensar em o que a classe deve ter (características/propriedades) e o que ela poderá fazer (comportamentos/ações)
- Devemos pensar apenas no essencial para que o problema que estamos tentando resolver seja solucionado. (abstração)
- Por exemplo, podemos pensar o que o carro deve ter:
  - • Fabricante
  - • Modelo
  - • Cor
  - • Tipo de combustível
  - • Ano de fabricação
  - • Valor de mercado
- E agora os comportamentos do carro (o que ele pode fazer):
  - • Ligar
  - • Desligar
  - • Mudar marcha
  - • Acelerar
  - • Frear

Orientação a Objetos

- Estrutura de uma Classe:



Orientação a Objetos

- Atributos:
  - São os elementos que definem a estrutura de uma classe.
  - O conjunto destes será responsável por representar suas características e farão parte dos objetos criados a partir da classe.
  - Os atributos também são conhecidos como variáveis de classe, e podem ser divididos em dois tipos básicos: atributos de instância e de classe.

Orientação a Objetos

- Exemplo de atributo:

Rodas

Motor

Modelo do carro?

Orientação a Objetos

- Assim como nas classes, os atributos podem ser representados a partir de substantivos.
- Além destes, podemos também usar adjetivos. Pensar em ambos pode facilitar o processo de identificação dos atributos

Implementações (sem métodos!)

Em Java

```java
public class Carro {
    //atributos
    String fabricante;
    String cor;
    String motor;
    int anoDeFabricacao;
    String modelo;
    boolean tracao;
    int aroRodas;
    String tipoCarro;
    double valorMercado;
    String status;

}
```

Em C#

```csharp
class Carro
{
    public string fabricante;
    public string cor;
    public string motor;
    public int anoDeFabricacao;
    public string modelo;
    public bool tracao;
    public int aroRodas;
    public string tipoCarro;
    public double valorMercado;
    public string status;

}
```

Orientação a Objetos

- **Métodos:**
  - Os métodos definem as **funcionalidades da classe**, ou seja, o que será possível fazer com objetos dessa classe. (o que ele faz o
  - Cada método é especificado por uma assinatura, composta por um identificador para o método (o nome do método), o tipo para o valor de retorno e sua lista de argumentos, sendo cada argumento identificado por seu tipo e nome.

Orientação a Objetos

- Exemplo de método:

Frear

"Pisar fuundo!!!"

Abastecer

OOP

- Objeto:
  - Representa uma entidade que pode ser física;
  - Um objeto é um elemento que representa, no domínio da solução, alguma entidade (abstrata ou concreta) do domínio de interesse do problema sob análise.
  - É a "materialização" da classe;
  - É o que pode ser feito através do "rascunho", da "ideia", do "modelo", do "projeto", da classe.
  - Pode se fazer centenas milhares de objetos diferentes a partir de uma mesma classe, de um mesmo modelo, de um mesmo projeto.

OOP

- **Objeto:**
  - No paradigma de orientação a objetos, tudo pode ser potencialmente representado como um objeto. Sob o ponto de vista da programação orientada a objetos, um objeto não é muito diferente de uma variável normal. Por exemplo, quando define-se uma variável do tipo **int** em uma linguagem de programação como C ou Java, essa variável tem:
    - Um espaço em memória para registrar o seu estado (valor);
    - Um conjunto de operações que podem ser aplicadas a ela, através dos operadores definidos na linguagem que podem ser aplicados a valores inteiros.

OOP

- **Objeto:**
  - Da mesma forma, quando se cria um objeto, esse objeto adquire um espaço em memória para armazenar seu estado (os valores de seu conjunto de atributos, definidos pela classe) e um conjunto de operações que podem ser aplicadas ao objeto (o conjunto de métodos definidos pela classe).
  - Um programa orientado a objetos é composto por um conjunto de objetos que interagem através de "trocas de mensagens".
  - Na prática, essa troca de mensagem traduz-se na aplicação de métodos a objetos.

POO

Objetos

Classes

- Agora já temos a classe "Carro", mas ainda não podemos ligar, acelerar ou frear um carro, e nem mesmo saber o fabricante, modelo ou a cor dele, pois ainda não existe um carro.
- Tudo que existe é uma ideia de um carro, uma especificação
- Nós podemos construir um ou vários carros a partir da classe "Carro", e isso se chama instanciação, na orientação a objetos

Classe "Carro"

Propriedades:
- Fabricante
- Modelo
- Cor
- Tipo de combustível
- Ano de fabricação
- Valor de mercado

Comportamentos:
- Ligar
- Desligar
- Mudar marcha
- Acelerar
- Frear

Objeto do tipo "Carro"

Objeto do tipo "Carro"

Implementações (em Java)

```java
public class Carro {
    //atributos
    String fabricante;
    String cor;
    String motor;
    int anoDeFabricacao;
    String modelo;
    boolean tracao;
    int aroRodas;
    String tipoCarro;
    double valorMercado;
    String status;

    //métodos
    void ligar(){
        System.out.println("O carro " + modelo + " ligou!");
        status = "ligado";
    }

    void frear(){
        System.out.println("O carro " + modelo + " está freando");
    }

    void desligar(){
        System.out.println("O carrro " + modelo + " desligou!");
        status="desligado";
    }

    void acelerar(){
        System.out.println("O carro " + modelo + " está acelerando!");
    }

    void acelerar(int velocidade){
        System.out.println("O carro " + modelo + " está acelerando!");
        for(int i=0;i<=velocidade;i+=5){
            System.out.print("Velocímetro: " + i + "\t" );
            if(i%10==0){
                System.out.println("\n");
            }
        }
    }

    void exibeStatus(){
        System.out.println("Status:.....:" + status);
    }

    void exibeInformacoes(){
        System.out.println("======= INFORMACOES DO CARRO =======");
        System.out.println("Modelo .....:" + modelo);
        System.out.println("Tipo.....:" + tipoCarro);
        System.out.println("Ano.....:" + anoDeFabricacao);
        System.out.println("Cor.....:" + cor);
        System.out.println("Fabricante.....:" + fabricante);
        System.out.println("Motor.....:" + motor);
        System.out.println("Rodas de aro.....:" + aroRodas);
        System.out.println("Valor de Mercado.:" + valorMercado);
        System.out.println("Status:.....:" + status);
        if(tracao==true){
            System.out.println("Carro traçado!");
        }else{
            System.out.println("Carro não traçado!");
        }
        System.out.println("====================================");

    }

}
```

Implementações (em C#)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace POO_2017_02
{
    class Carro
    {
        public string motor;
        public int anoDeFabricacao;
        public string modelo;
        public int quantMarchas;
        public bool tracao;
        public int aroRodas;
        public string tipoCarro;
        public string cor;

        public void ligar()
        {
            MessageBox.Show("O carro " + this.modelo + " ligou!");
        }

        public void acelerar()
        {
            MessageBox.Show("O carro " + this.modelo + " está acelerando!");
        }

        public void acelerar(int velocidade)
        {
            string mensagem = "O carro está acelerando\n";
            for (int i = 0; i <= velocidade; i += 5)
            {
                mensagem += "Velocímetro: " + i + "\t";
                if (i % 10 == 0)
                {
                    mensagem += "\n";
                }
            }
            MessageBox.Show(mensagem);
        }
```

```

        public void frear()
        {
            MessageBox.Show("O carro " + modelo + " está freando!");
        }

        public void desligar()
        {
            MessageBox.Show("O carro " + modelo + " está desligando!");
        }

        public void exibeInformacoes()
        {
            string mensagem = "";
            mensagem += "======= INFORMACOES DO CARRO =======" + "\n";
            mensagem += "Modelo ..........:" + modelo + "\n";
            mensagem += "Tipo.............:" + tipoCarro + "\n";
            mensagem += "Ano..............:" + anoDeFabricacao + "\n";
            mensagem += "Cor..............:" + cor + "\n";
            mensagem += "Quant. marchas...:" + quantMarchas + "\n";
            mensagem += "Motor............:" + motor + "\n";
            mensagem += "Rodas de aro.....:" + aroRodas + "\n";
            if (tracao == true)
            {
                mensagem += "Carro traçado!" + "\n";
            }
            else
            {
                mensagem += "Carro não traçado!" + "\n";
            }
            mensagem += "====================================" + "\n";

            MessageBox.Show(mensagem);
        }
    }

}
```

POO

- **Encapsulamento:**
  - É o princípio de projeto pelo qual cada componente de um programa deve agregar toda a informação relevante para sua manipulação como uma unidade (uma cápsula). Aliado ao conceito de ocultamento de informação, é um poderoso mecanismo da programação orientada a objetos.
- *Encapsulamento* é nada mais é do que **esconder a regra de negócio na classe certa**.
- Por exemplo, uma classe Conta deve encapsular as regras de depositar, sacar ou transferir.

Encapsulamento

- Quando alguém se consulta com um médico, por estar com um **resfriado**, seria desesperador se ao final da consulta o médico entregasse a seguinte receita:

RECEITUÁRIO (COMPLEXO)

- 400mg de ácido acetilsalicílico
- 1mg de maleato de dexclorfeniramina
- 10mg de cloridrato de fenilefrina
- 30mg de cafeína

Misturar bem e ingerir com água. Repetir em momentos de crise.

Encapsulamento

- A primeira coisa que viria em mente seria: onde achar essas substâncias? Será que é vendido tão pouco? Como misturá-las?
- Existe alguma sequência? Seria uma tarefa difícil — até complexa — de ser realizada.
- Mais simples do que isso é o que os médicos realmente fazem: passam uma cápsula onde todas estas substâncias já estão prontas. Ou seja, elas já vêm encapsuladas.
- O que interessa é o resultado final, no caso, a cura do resfriado.

Encapsulamento

- A complexidade de chegar a essas medidas e como misturá-las não interessa.
- É um processo que não precisa ser do conhecimento do paciente.
- Encapsular é esconder a complexidade. Ou deixar a complexidade onde ela é realmente necessária.

RECEITUÁRIO (ENCAPSULADO)

1 comprimido de Resfriol. Ingerir com água. Repetir em momentos de crise.

Encapsulamento: escondendo complexidades

Encapsulamento

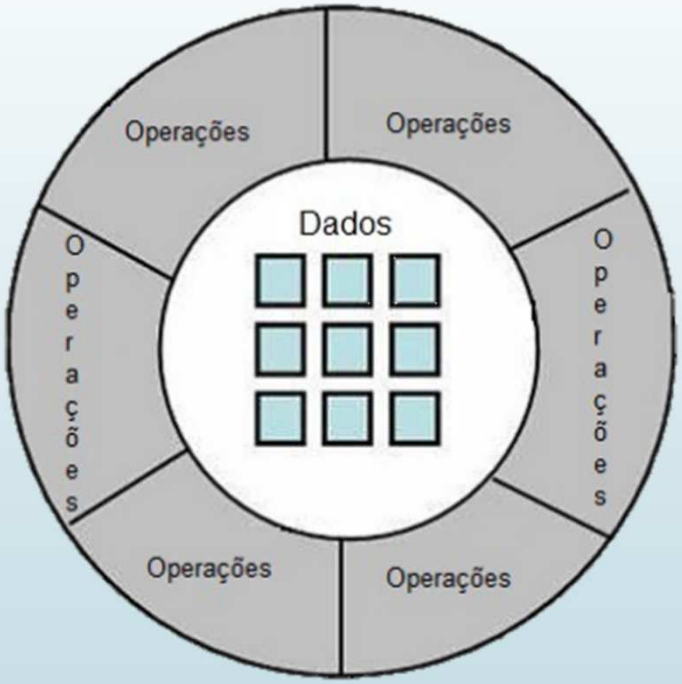
- Essa mesma ideia se aplica na Orientação a Objetos. No caso, a complexidade que desejamos esconder é a de implementação de alguma necessidade.
- Com o encapsulamento, podemos esconder a forma como algo foi feito, dando a quem precisa apenas o resultado gerado.
- Não importa para quem requisitou algum processamento/comportamento ter conhecimento da lógica realizada para gerar o resultado final.
- Apenas o resultado obtido é que é relevante.

Encapsulamento

- Uma vantagem deste princípio é que as mudanças se tornam transparentes
- Linguagens estruturadas proveem este fundamento, mas é mais difícil para atingi-lo.
- Os conceitos de classe, método, entre outros facilitam em muito a aplicação deste fundamento
- Uma outra característica do encapsulamento é também a ocultação da informação. Neste caso, ele blinda o aspecto interno do objeto em relação ao mundo exterior.
- Assim, cria-se uma casca (os métodos que aprenderemos mais adiante) ao redor das características (atributos), que tem como finalidade evitar resultados inesperados, acessos indevidos, entre outros problemas.

Encapsulamento

- Encapsulamento: ocultação da informação



Programação Orientada a Objetos

- O termo mais importante na POO é a classe
- Uma classe é um template ou modelo a partir do qual os objetos reais são feitos
- Quando você constrói um objeto a partir de uma classe, diz-se que você criou uma instância da classe.
- O encapsulamento (às vezes chamado de data hiding) é um conceito-chave para se trabalhar com objetos.
- O encapsulamento nada mais é do que combinar dados e comportamentos em um pacote e esconder a implementação dos dados do usuário do objeto.

Programação Orientada a Objetos

- Os dados (ou propriedades) dos objetos são chamados de campos de instâncias do objeto e os procedimentos que operam sobre os dados são chamados de métodos do objeto
- As classes também podem ser estendidas usando um outro princípio da OOP.
- Quando você estende uma classe, a nova classe passa a ter todas as propriedades e os métodos da classe que você estendeu.
- Você fornece somente novos métodos e campos de dados que se aplicam somente à nova classe.
- Este conceito, de estender uma classe chama-se herança.

Programação Orientada a Objetos

- Para se trabalhar (ou entender a OOP) deve-se ser capaz de identificar 3 características principais dos objetos:
  - O comportamento do objeto – o que você pode fazer com este objeto, ou quais métodos você pode aplicar a ele?
  - O estado do objeto – como o objeto reage quando você aplica estes métodos?
  - A identidade do objeto – como o objeto é distinguido de outros que possam ter o mesmo comportamento e estado.

Programação Orientada a Objetos

- O comportamento de um objeto é definido pelos métodos que se pode chamar
- Cada objeto armazena informações sobre sua aparência atual. Isso fica no estado do objeto. O estado de um objeto pode mudar com o tempo, mas não espontaneamente
- Uma mudança de estado deve ser a consequência de chamadas a métodos
- Cada objeto tem também uma identidade. Por exemplo, em um sistema de processamento de pedidos, dois pedidos são distinguidos mesmo se eles especificarem itens idênticos.

Programação Orientada a Objetos

- Essas características podem influenciar umas às outras
- Por exemplo, o estado de um objeto pode influenciar seu comportamento (se um pedido é marcado como "enviado" ou "pago", ele pode rejeitar uma chamada a método que pede para adicionar ou remover itens dele. Ou ele pode não ser enviado se estiver "vazio")

Programação Orientada a Objetos

- Em um programa procedimental tradicional começa-se o processamento de cima para baixo (com a função main)
- Na POO não há "cima" e iniciantes em POO costumam perguntar como começar.
- A resposta: encontre as classes e depois adicione os métodos
- Uma dica para identificar classes é procurar por substantivos na análise do problema
- Os métodos, por outro lado, correspondem aos verbos.

Programação Orientada a Objetos

- Por exemplo: em um sistema de processamento de pedidos, alguns desses substantivos são:
- Item
- Pedido
- Endereço de entrega
- Pagamento
- Conta
- Esses substantivos podem levar às classes Item, Pedido e assim por diante

Programação Orientada a Objetos

- Em seguida se procura por verbos:
- Os itens podem ser adicionados aos pedidos
- Os pedidos são enviados ou cancelados
- Os pagamentos são aplicados aos pedidos
- Cada verbo como adicionar, enviar, cancelar e aplicar pode se tornar um método e cada verbo identifica o objeto correspondente pela ação
- É claro que isso é somente uma "dica"
- Somente a experiência poderá dizer quais verbos ou substantivos são importantes!

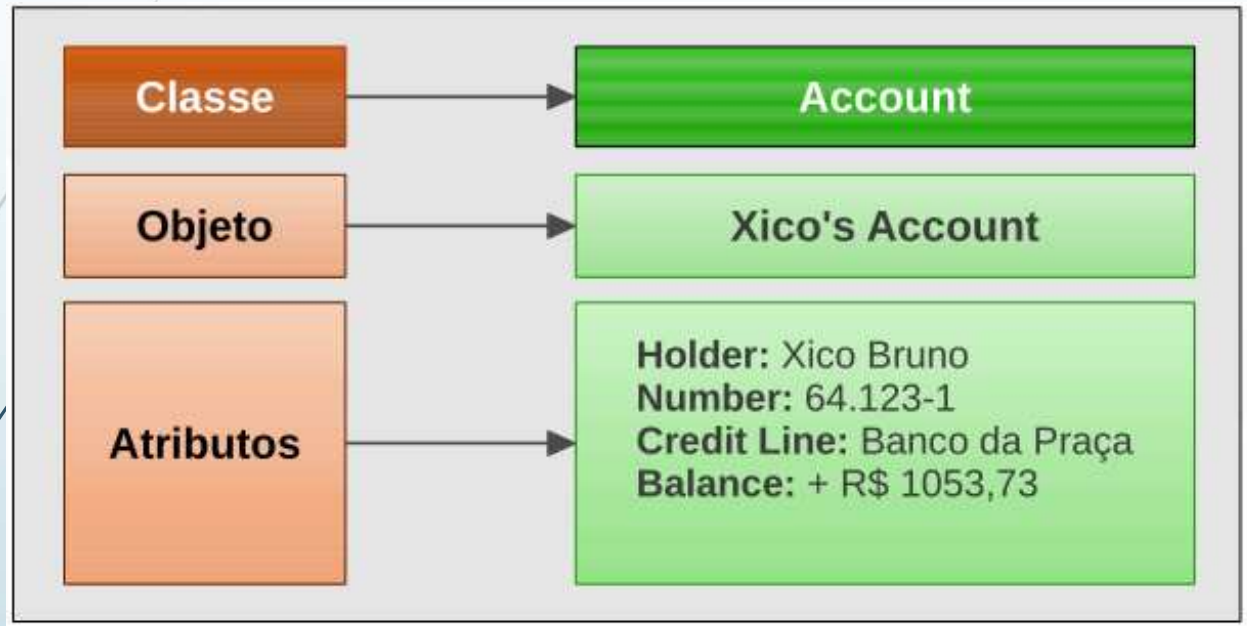
Cachorro
Atributos: - nome - peso - cor do pêlo
Métodos: - latir() - abanar()

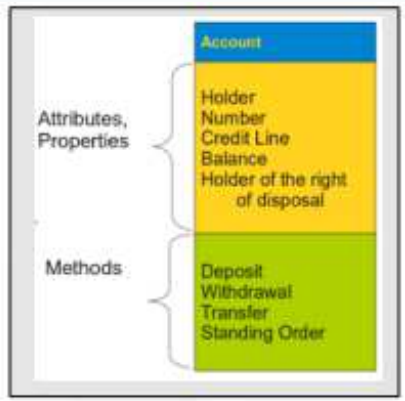
Resumo POO

- Classe:
  - Define:
    - Um tipo de “coisa”(Objeto / Instância);
    - O que esses objetos possuem como características(Atributos);
    - O tipo de ações que esses objetos são capazes de realizar(Métodos);
- Objeto:
  - Observe que uma classe “não tem vida”, ela é um conceito;
  - Já os Objetos(felinos, humanos... Da classe Mamíferos) possuem;

Resumo POO

- Por exemplo:





Resumo POO

- Métodos:
  - Serão as ações que poderão ser executadas no/com objetos da classe;
- Encapsulamento:
  - Consiste na separação de aspectos internos e externos de um objeto;
  - Exemplo: você não precisa conhecer os detalhes dos circuitos de um telefone para utilizá-lo. A carcaça do telefone encapsula esses detalhes, provendo a você uma interface mais amigável (os botões, o monofone e os sinais de tom);
  - Isso permite que o programador possa determinar a forma como os dados poderão ser "vistos" e modificados;

Resumo POO

- Herança:
  - É o mecanismo pelo qual uma classe (sub-classe) pode estender outra classe (super-classe), aproveitando seus comportamentos (métodos) e variáveis possíveis (atributos).

Resumo POO

- O polimorfismo é caracterizado quando duas ou mais classes distintas tem métodos de mesmo nome, de forma que uma função possa utilizar um objeto de qualquer uma das classes polimórficas, sem necessidade de tratar de forma diferenciada conforme a classe do objeto
- Será visto na prática!
- Uma das formas de implementar o polimorfismo é através de uma classe abstrata, cujos métodos são declarados mas não são definidos, e através de classes que herdam os métodos desta classe abstrata

Perguntas e Discussão