Estácio

Programação I

PROF. LUCAS CAMPOS DE M. NUNES

http://lattes.cnpq.br/2803226406709573

Brasília, agosto de 2017

Aula 3

Princípios da Orientação a Objetos

- Objetivos

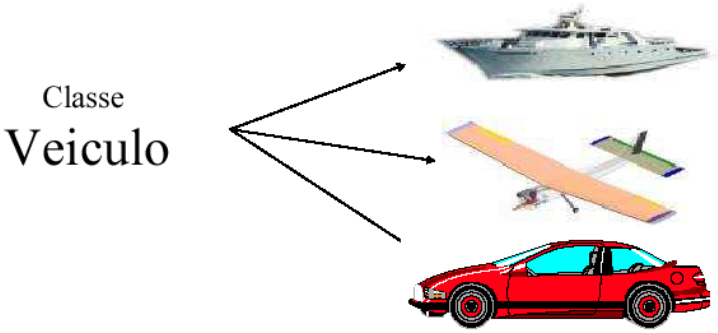
1. Identificar os conceitos básicos da Orientação à Objetos;
2. Criar e manipular classes e objetos em Java;
3. Utilizar interfaces gráficas interagindo com objetos e classes previamente definidos;
4. - Implementar classes e instanciar objetos em Java

Introdução Classes e Objetos

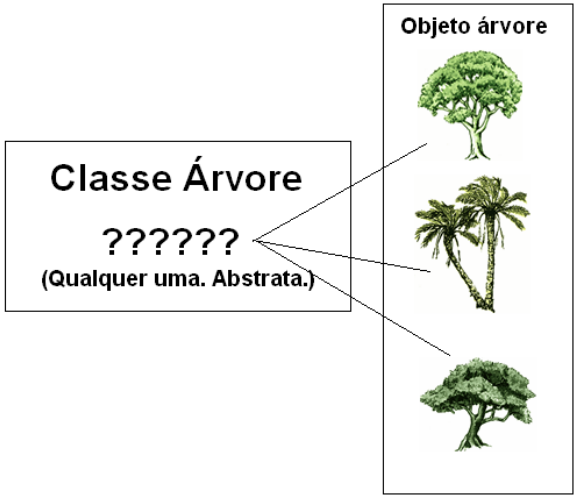
- Todo sistema orientado à objetos pode ser definido como um conjunto de objetos que trocam mensagens entre si;
- Então, quando programamos em Java, que é uma linguagem de programação orientada à objetos, estamos escrevendo ou definindo um conjunto de objetos que trocam mensagens;
- Bom, mas o que é um Objeto? O que é uma classe? O que é mensagem?

Classes

- As classes definem a estrutura e o comportamento de um tipo de objeto. Observe a figura abaixo:

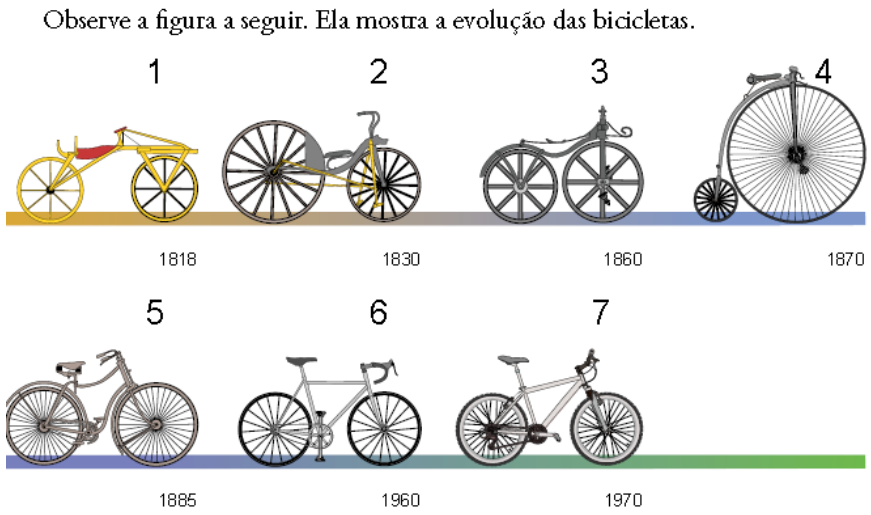


- A classe Árvore define um arcabouço de qualquer tipo de árvore. Uma árvore real, será um objeto com estrutura a estrutura definida pela classe, mas os valores de cada árvore serão únicas.



Classes

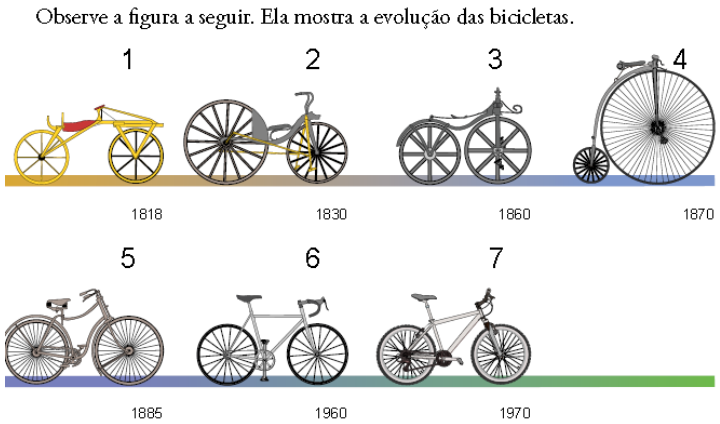
- O que elas tem em comum? Vamos tentar listar as **características:**
- rodas;
- tamanho;
- velocidade (se estiver parada, a velocidade e zero);
- marchas (ou não);
- modelo, marca, cor, material etc.



- **Comportamentos** em comum: aceleram, freiam (esses dois comportamentos afetam a propriedade velocidade), viram à direita e à esquerda (possuem também a característica direção), mudam a marcha (quando possuem esta característica);

Classes

- Para ser uma bicicleta é necessário um molde, um padrão, uma forma de fazer bicicletas;
- Isso é o conceito que damos para classes;
- Cada bicicleta criada a partir deste molde, deste padrão será chamado de objeto;



- **O objeto** também pode ser definido como instância de uma classe;

Classes

- A orientação a objetos teve sua origem no laboratório da Xerox e na linguagem ***Smalltalk***, liderada por um pesquisador chamado Alan Curtis Kay;
- Pelos seus estudos, ele pensou em como construir softwares a partir de agentes autônomos que interagiam entre si, originando, assim, a orientação a objetos com estas características:
  - Qualquer coisa é um objeto;
  - Objetos realizam tarefas através da requisição de serviços;
  - Cada objeto pertence a uma determinada classe;
  - Uma classe agrupa objetos similares;
  - Uma classe possui comportamentos associados ao objeto;
  - Classes são organizadas em hierarquias;

Abstração

- Extrair tudo que for essencial e mais nada para o escopo do sistema;
- É o processo de filtragem de detalhes sem importância do objeto real, para que apenas as características apropriadas que o descrevam e que tenham relevância para o sistema permaneçam;
- Conceito aplicado a criação de software baseado em objetos, partindo do princípio que devemos considerar a essência de cada objeto e não pensar em todos os detalhes de implementação;

Classes e Objetos em Java

- Na caixa de dialogo do novo projeto, escolhemos um aplicativo Java
- Um programa Java é uma coleção de objetos que são descritos por um conjunto de arquivos texto, onde são definidas as classes. Pelo menos uma destas classes é “public” e contém o método main(), que possui esta forma específica:

```
public static void main(String [] args) {
        // aqui o programador insere os comandos
}
```

- Como vimos nas aulas anteriores todos os programas feitos tinham esta estrutura. E o nome da classe era o nome do arquivo. Então, podemos concluir que uma classe em Java é construída dentro de um arquivo texto com o mesmo nome da classe e extensão .java.
- Lembro que a classe que contém o método main() é chamada de classe principal.

Classes e Objetos em Java

- Para se criar uma classe em Java, a sintaxe é (mais detalhes a frente):

```
class Bicicleta {
        ...
    /* aqui nesta parte são inseridos as características e
comportamentos da classe
}
```

- Para se criar um objeto da classe Bicicleta, a sintaxe é:

```
Nome_classe nome_objeto = newNome_classe;
Bicicleta minhaBike = new Bicicleta();
```

- O objeto e criado por meio do comando new;
- O exemplo anterior criou um objeto chamado “minhaBike” da classe Bicicleta;

Atributos, métodos e construtor

- Como já deve ter percebido, o conjunto de atributos (características) formam uma parte da estrutura de uma classe;
- Os atributos podem ser conhecidos também como variáveis de classe e podem ser classificados em duas formas: atributos de instância e atributos de classe;
  - **Veja esse exemplo:** Uma bicicleta da marca X possui 21 marchas e é do tipo mountain bike. Ela possui o numero de serie 12345 do lote 54321 e no momento esta parada (velocidade = 0), logo também possui aceleração igual a 0. A marcha atual dela e a numero 5.
  - Perceba que a **bicicleta foi definida** de acordo com suas características, ou como chamamos na Orientação a Objetos (OO) de **atributos**;
- Os valores que estão nos atributos podem ser modificados?
  - Sim, por **meio dos métodos**!

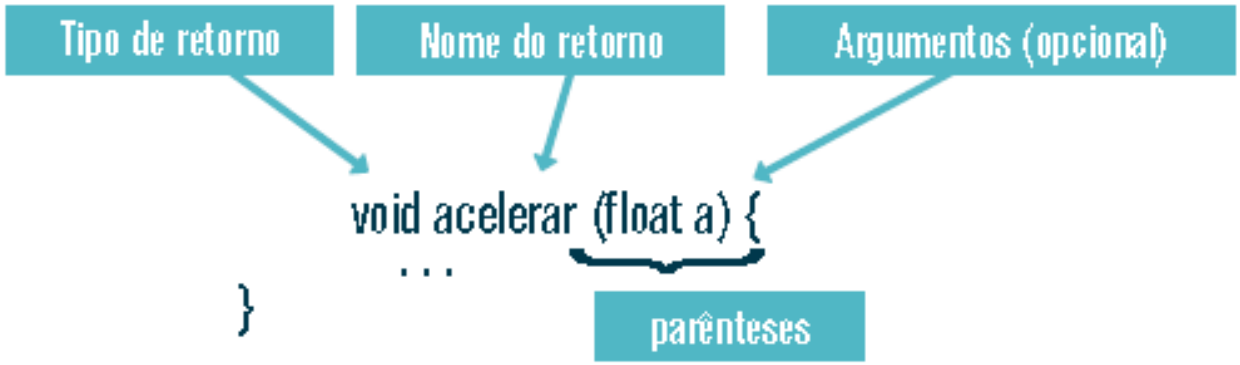
Atributos, métodos e construtor

- **Os métodos** representam o **comportamento dos objetos;**
- **Os métodos mudam os atributos (características) de um objeto**;
- No caso da bicicleta, podemos ter os seguintes métodos como exemplo: acelerar, parar, mudar a marcha, virar, etc;
- Na prática, o que aprendemos sobre funções lá na disciplina de logica de programação serão os métodos agora;

```
class Bicicleta {
        //atributos
    String numeroDeSerie;
    String lote;
    String marca;
    String tipo;
    String direcao;
    float velocidade;
    float aceleração;
    int numeroDeMarchas;
    int marchaAtual;

        //métodos
    void parar(){
        ...
    }

    void acelerar(float a){
        ...
    }

    void virar(Stringdirecao){
        ...
    }

    void mudarAMarcha(int novaMarcha){
        ...
    }
}
```

Atributos, métodos e construtor

Os únicos elementos requeridos na declaração de métodos são: o tipo de retorno do método, o nome, o par de parênteses e o corpo entre chaves { e }. Observe:



Atributos, métodos e construtor

Em geral, a definição de um método tem os seguintes componentes:

1. Modificadores: como public, private e outros. Quando não é usado, a classe e/ou seus membros são acessíveis somente por classes do mesmo pacote. Neste caso, na sua declaração não é definido nenhum tipo de modificador, sendo este identificado pelo compilador.
2. O tipo de retorno: o tipo de dado do valor retornado pelo método ou void se não há retorno.
3. O nome do método.
4. A lista de argumentos entre parênteses: os argumentos são separados por vírgula, seguido pelo tipo de dados e pelo nome do argumento. Se não houver parâmetros, usa-se os parênteses sem parâmetros, como é o caso do método parar() na classe Bicicleta.
5. O corpo do método, entre chaves.

- Assim como os nomes de classes e variáveis, os nomes dos métodos também devem obedecer a convenção usada no Java (*CamelCase*);
- Além disso, e recomendável que todo método comece por um verbo (com a primeira letra em minúscula);
- Veja alguns exemplos: correr, correrRapido, getBackground, getDados, compareTo, setX, isEmpty;

Atributos, métodos e construtor

- Quando um objeto e criado ele automaticamente recebe todos os atributos e métodos de sua classe;

```
public class Janela{
        //atributos
    int posicaoX;
    int posicaoY;
    int largura;
    int altura;
    String tipo
        // outros atributos

        //métodos
    void maximizar()      {...}
    void minimizar()      {...}
    void mover()   {...}
}
```

```
public class Janela{
        //atributos
    int posicaoX;
    int posicaoY;
    int largura;
    int altura;
    String tipo
        // outros atributos

        //constutor
    Janela(){
        posicaoX = 100;
        posicaoY = 100;
        largura = 50;
        altura = 50;
        tipo = "Formulario"
    }

        //métodos
    void maximizar()      {...}
    void minimizar()      {...}
    void mover()   {...}
}
```

observe que o nome do método é exatamente igual ao nome da classe

Atributos, métodos e construtor

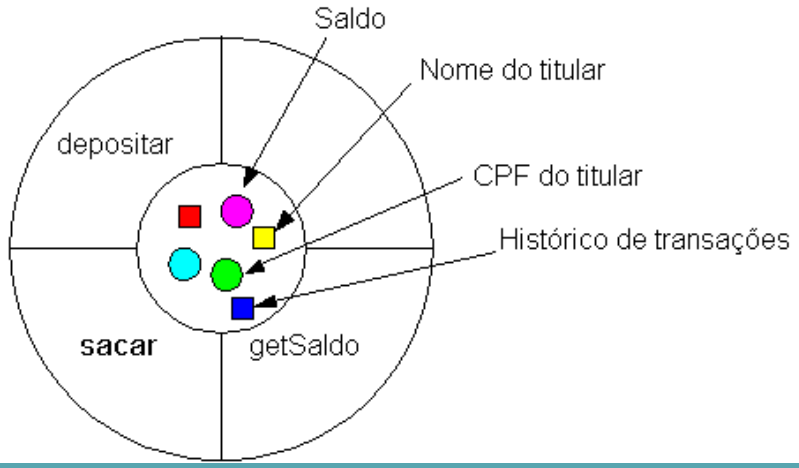
- Toda vez que uma janela e criada no Windows por exemplo, ela recebera uma posição na tela por padrão, uma altura, uma largura, tipo etc;
- **O construtor é um método especial que e executado quando um objeto e criado e serve principalmente para inserir valores nas variáveis de instancia da classe**;
- Tenha o habito de sempre criar pelo menos um construtor para suas classes;
- As declarações de construtores se parecem com declarações de métodos, exceto que eles tem exatamente o mesmo nome da classe e não possuem tipos de retorno;

Encapsulamento

- O encapsulamento é um dos quatro pilares da orientação a objetos (Abstração, Encapsulamento, Herança e Polimorfismo);
- Na orientação a objetos, o encapsulamento evita que o usuário da classe saiba detalhes de como os métodos são implementados;
- Exemplo: Quando você vai realizar uma operação (um saque, deposito) você não precisa, e nem quer saber, quais são os detalhes internos que ocorrem no banco, só quer ver seu dinheiro entrando ou saindo da conta;

Encapsulamento

- Mecanismo utilizado em orientação a objetos para obter segurança, modularidade e autonomia dos objetos;
- Implementamos através da definição de visibilidade privada dos atributos;
- Então, devemos sempre definir os atributos de uma classe como privados;
- Este mecanismo protege o acesso direto aos dados do objeto;
- Permite acesso através dos métodos públicos;



Encapsulamento

- Os modificadores de acesso servem para deixar os componentes de uma classe disponíveis ou não para outras partes dentro da classe ou mesmo para outras classes;
- Vejamos os dois principais:
  - O ***public*** faz com que o componente (atributo ou método) seja acessível para todas as outras classes;
  - O ***private*** faz com que o componente (atributo ou método) seja acessível somente dentro da propria classe.

Nível de Encapsulamento

- É a disponibilização de uma interface pública, com granularidade controlada, para manipular os estados e executar as operações de um objeto (* acesso permitido).

Modificadores	Mesma classe	Mesmo pacote	Subclasses	Universo
private	*			
<sem modif>	*	*		
protected	*	*	*	
public	*	*	*	*

Nível de Encapsulamento

- De acordo com o que já vimos sobre encapsulamento, o usuário da classe só poderá ter acesso aos componentes públicos, então **é comum e recomendável que os campos sejam *private***;
- Isso significa que eles somente podem ser acessados diretamente por sua classe;
  - Porém, é necessário acessar esses campos e seus valores;
- Isso pode ser feito indiretamente por meio de métodos públicos que obtém o valor para nós;

Resumo - Definindo uma classe em Java

- São definidas através do uso da palavra-chave class.
- Construindo uma classe:

```
[modif] class NomeDaClasse {
        // corpo da classe...
}
```

- A primeira linha é um comando que inicia a declaração da classe. Após a palavra-chave class, segue-se o nome da classe, que deve ser um identificador válido para a linguagem. O modificador modif é opcional; se presente, pode ser uma combinação de public e abstract ou final. A definição da classe propriamente dita está entre as chaves { e }, que delimitam blocos na linguagem Java.

Corpo da Classe

1. As variáveis de classe (definidas como static), ordenadas segundo sua visibilidade: iniciando pelas public, seguidos pelas protected, pelas com visibilidade padrão (sem modificador) e finalmente pelas private.
2. Os atributos (ou variáveis de instância) dos objetos dessa classe, seguindo a mesma ordenação segundo a visibilidade definida para as variáveis de classe.
3. Os construtores de objetos dessa classe.
4. Os métodos da classe, geralmente agrupados por funcionalidade.

Resumo - Criando Objetos em Java

- **Toda classe possui atributos e métodos. Em especial, toda classe tem um construtor, que é um método responsável pela instanciação do objeto.**
- **Este processo faz com que o objeto seja criado com as características desejadas.**
- **O construtor é o responsável por alocar a memória do objeto.**
- **A criação de um objeto se dá através da aplicação do operador new.**

```
ClasseNome objeto = new ClasseNome();
```

Resumo - Atributos e Métodos

- Atributos
  - Eles representam as características de um objeto. Devem ser privados, para manter o encapsulamento. A definição de atributos de uma classe Java reflete de forma quase direta a informação que estaria contida na representação da classe em um diagrama UML.
- Métodos
  - Os métodos representam as funcionalidades que os objetos podem desempenhar. São essencialmente procedimentos que podem manipular atributos de objetos para os quais o método foi definido. Além dos atributos de objetos, métodos podem definir e manipular variáveis locais; também podem receber parâmetros por valor através da lista de argumentos.

Resumo - Atributos

- A sintaxe utilizada para definir um atributo de um objeto é:

  [modificador] tipo nome [= default];

- Onde:
  - modificador é opcional, especificando a visibilidade diferente da padrão (public, protected ou private);
  - tipo deve ser um dos tipos primitivos da linguagem Java ou o nome de uma classe;
  - nome deve ser um identificador válido da linguagem Java;
  - valor default é opcional; se presente, especifica um valor inicial para a variável.

Resumo - Métodos

- A forma genérica para a definição de um método em uma classe é

```
[modificador] tipo nome(argumentos) {
        corpo do método
}
```

- Onde:
  - o modificador (opcional) é uma combinação de: public, protected ou private; abstract ou final; e static.
  - o tipo é um indicador do valor de retorno, sendo void se o método não tiver um valor de retorno;
  - o nome do método deve ser um identificador válido na linguagem Java;
  - os argumentos são representados por uma lista de parâmetros separados por vírgulas, onde para cada parâmetro é indicado primeiro o tipo e depois (separado por espaço) o nome.

Boas Práticas de Programação

- Os métodos devem ser simples, atendendo a sua funcionalidade
- O nome do método deve refletir de modo adequado a tarefa realizada
- Se a funcionalidade do método for simples, será fácil encontrar um nome adequado para o método.
- Métodos de mesmo nome podem co-existir em uma mesma classe desde que a lista de argumentos seja distinta, usando o mecanismo de sobrecarga.

Exemplo

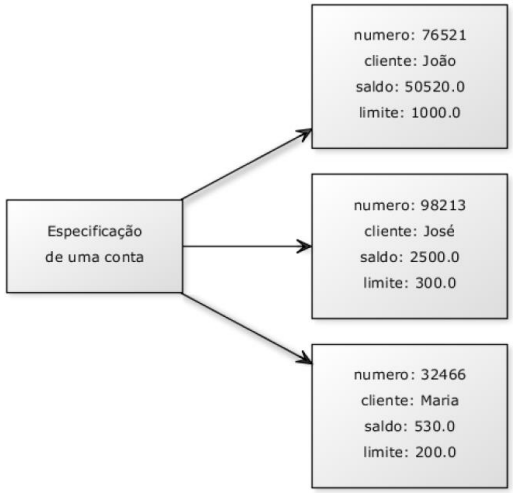
```
public class Ponto2D {
    private int x;
    private int y;
    public Ponto2D(int a, int b) {
            x = a;
            y = b;
    }
    public double distancia(Ponto2D p) {
            double distX = p.x - x;
            double distY = p.y - y;
            return Math.sqrt(distX*distX + distY*distY);
    }
}
```

Exemplo

- Neste exemplo, definimos a classe Ponto2D com dois atributos e dois métodos. Os atributos são, x e y enquanto os métodos são:
- Ponto2D – Construtor
- distancia – método que calcula a distância entre o ponto e um outro dado.

Exercício Guiado – O.O

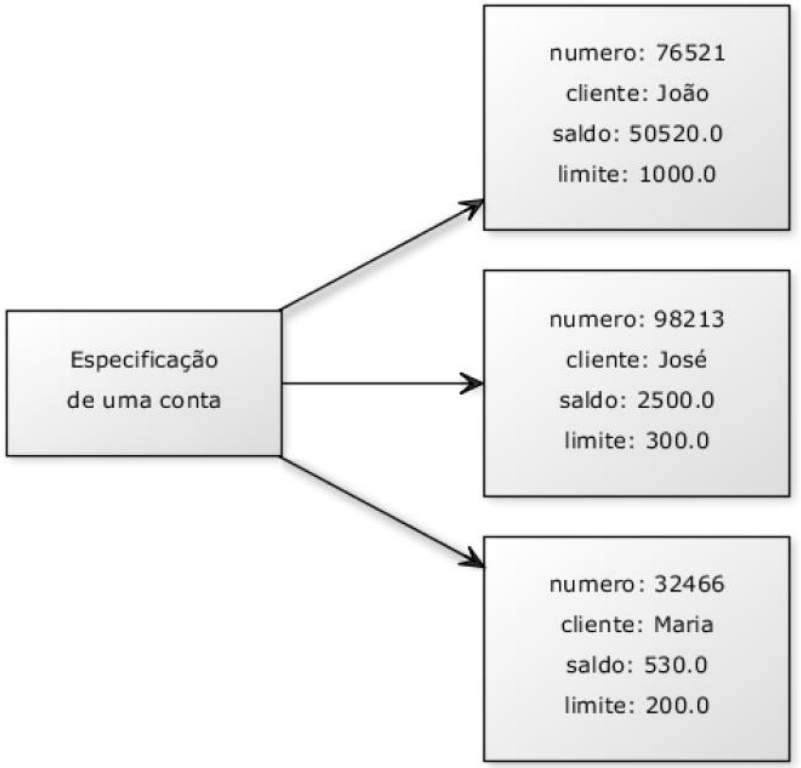
- Considere um programa para um banco, é bem fácil perceber que uma entidade extremamente importante para o nosso sistema é a conta;
- A ideia aqui é generalizarmos alguma informação, juntamente com funcionalidades que toda conta deve ter;
- O que toda **conta tem (propriedades/atributos)** e é importante?
  - número da conta;
  - nome do dono da conta;
  - saldo;
  - limite;



- **OBS: Exercício adaptado da apostila de orientação a objetos da Caelum;**

Exercício Guiado – O.O

- O que toda **conta faz (comportamentos/métodos)** e é importante para nós? Isto é, o que gostaríamos de "pedir à conta"?
  - sacar uma quantidade x;
  - depositar uma quantidade x;
  - imprime o nome do dono da conta;
  - devolve o saldo atual;
  - transfere uma quantidade x para uma outra conta y;
  - devolve o tipo de conta;



Exercício Guiado – O.O

- O que toda **conta faz (comportamentos/métodos)** e é importante para nós? Isto é, o que gostaríamos de “pedir à conta"?
  - sacar uma quantidade x;
  - depositar uma quantidade x;
  - imprime o nome do dono da conta;
  - Mostra o saldo atual;
  - transfere uma quantidade x para uma outra conta y;
  - Mostra o tipo de conta;

Exercício Guiado – O.O

- Primeiro passo é criar um novo projeto no NetBeans;
- Depois criar uma classe chamada Conta;

```
public class Conta
{
    int numero;
    String dono;
    double saldo;
    double limite;
}
```

- Na método principal crie um objeto da classe Conta;

```
public static void main(String[] args) {
        // TODO code application logic here
        Conta minhaConta = new Conta();
    }
```

Exercício Guiado – O.O

- Atribua alguns valores às propriedades e mostre no console;

```
class Programa
{
        public static void main(String[] args)
        {
                Conta minhaConta;
                minhaConta = new Conta();
                minhaConta.dono = "Duke";
                minhaConta.saldo = 1000.0;
                System.out.println("Saldo atual: " + minhaConta.saldo);
        }
}
```

Exercício Guiado – O.O

- Vamos adicionar alguns métodos que identificamos no início:
  - Sacar uma quantidade x;
  - Depositar uma quantidade x;
  - Imprime o nome do dono da conta;
  - Mostra o saldo atual;
  - Transfere uma quantidade x para uma outra conta y;
  - Mostra o tipo de conta;

Exercício Guiado – O.O

- Vamos adicionar alguns métodos que identificamos no início:
  - sacar uma quantidade x;

```
void sacar(double quantidade) {
        double novoSaldo = this.saldo - quantidade;
        this.saldo = novoSaldo;
        }
```

Exercício Guiado – O.O

- Vamos adicionar alguns métodos que identificamos no início:
  - depositar uma quantidade x;

```
public void DepositarDinheiro(double pValorDeposito)
    {
        double novoSaldo = this.saldo + pValorDeposito;
        this.saldo = novoSaldo;
    }
```

Exercício Guiado – O.O

- Vamos adicionar alguns métodos que identificamos no início:
  - imprime o nome do dono da conta;

```
public String ImprimirDonoConta (int pNumConta)
    {
        String retorno = "";
        if(this.numero == pNumConta)
            retorno = "Dono da conta " + pNumConta + " é o " + this.dono + "!";
        else
            retorno = "Não existe ninguém com esta conta!";

        return retorno;
    }
```

Exercício Guiado – O.O

- Vamos adicionar alguns métodos que identificamos no início:
  - Mostra o saldo atual;

```
public void MostrarSaldo(int pNumConta)
    {
        if(this.numero == pNumConta)
            System.out.println("O saldo da conta " + this.numero + " é de R$ "
+ this.saldo);
        else
            System.out.println("Não existe ninguém com esta conta!");
    }
```

Exercício Guiado – O.O

- Vamos adicionar alguns métodos que identificamos no início:
  - Transfere uma quantidade x para uma outra conta y;
  - Mostra o tipo de conta;

Fim!