

# Programação Orientada a Objetos - POOS3

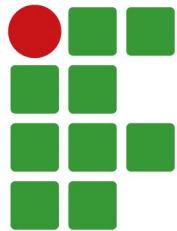
1

Tecnologia em Análise e Desenvolvimento de Sistemas

## Aula 2

Introdução à Programação Orientada a  
Objetos

2º semestre de 2018



**INSTITUTO FEDERAL**

São Paulo

Câmpus Araraquara

# O que é Programação Orientada a Objetos?

---

- É um paradigma de programação baseado no conceito de classes e objetos.
- As classes são elementos em que dados e funcionalidades podem ser agrupados, segundo sua função para um determinado sistema; essas classes são codificadas em formatos de arquivos.
- Quando uma dessas classes é utilizada como um tipo de dado para a criação de uma variável, esta é chamada de objeto.

# O que é classe?

---

- A classe é definida como uma estrutura de dados que contém **atributos** e **métodos**.
- Ao se criar uma classe, o objetivo é agrupar métodos e atributos que estejam relacionados entre si.
- Uma classe é composta de partes e estas devem representar alguma funcionalidade segundo o objetivo da classe.

# O que é classe?

---

- As partes de uma classe devem representar funcionalidades para atender o objetivo da classe.
- Exemplo, uma classe Cliente:
  - Quais os dados relacionados a um cliente? (nome, endereço, cidade, estado, etc)
  - Quais outras informações desse cliente devem ser armazenadas?
  - Quais atitudes um cliente pode tomar ou quais atitudes podem ser tomadas diante de um cliente?

# O que é classe?

---

- Concluindo, a classe é o código que declara atributos e métodos, em que cada um destes possa fazer parte da representação de um mesmo objetivo.
- O objetivo da classe é representar de forma adequada uma entidade dentro de um sistema.

# O que é objeto?

---

- Classe é somente a codificação na forma de arquivo texto, um objeto é uma instância de uma classe.
- É uma porção de memória reservada para armazenar os dados e os métodos declarados na classe.
- **Um objeto é a instância de uma classe na memória.**

# O que é objeto?

- A classe é o código-fonte escrito em um arquivo texto, enquanto o objeto é uma parte de uma aplicação durante o processo de execução.



= **Objeto**



= **Classe**

# Exemplo de uso de objeto

```
public static void main(String[] args) {  
    Scanner input;  
    input = new Scanner(System.in);  
    int lado1, lado2, lado3;  
    int contador;  
    boolean ehTriangulo;  
    String tipo;  
  
    contador = 1;  
    do {  
        System.out.println("Digite três lados do triangulo:"  
        lado1 = input.nextInt();  
        lado2 = input.nextInt();  
        lado3 = input.nextInt();  
  
        ehTriangulo = formaTriangulo(lado1, lado2, lado3);  
  
        if(ehTriangulo) {  
            tipo = tipoTriangulo(lado1, lado2, lado3);  
            System.out.println("DADOS FORMAM " + tipo);  
        }else {  
            System.out.println("DADOS NÃO FORMAM TRIANGULO");  
        }  
        contador += 1;  
    }while(contador <= 5);  
}
```

A classe Scanner possui o código de vários métodos que permitem a leitura de dados de diferentes tipos.

O objetivo da classe Scanner é recuperar dados informados pelo teclado.

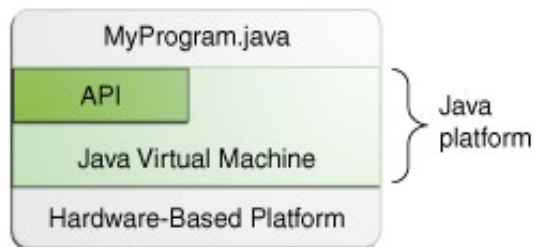
Ao **INSTÂNCIAR** um objeto da classe Scanner estamos criando uma representação na memória. Essa representação possui poder de execução, diferente da Classe que é apenas texto.

Observe o uso do objeto e não da classe.



# Documentação do Java - Javadoc

- Toda documentação das Classes que compõem a **API Java** estão disponíveis no Javadoc:
- <https://docs.oracle.com/javase/8/docs/api/index.html>



# O que é uma mensagem?

- A mensagem é definida como o ato de chamar ou requisitar a execução de um método.

```
Scanner input = new Scanner(System.in);  
lado1 = input.nextInt();  
lado2 = input.nextInt();  
lado3 = input.nextInt();
```

Esta é uma mensagem enviada ao método `nextInt()` do objeto `input` que está implementado na classe `Scanner`.

# O que é Encapsulamento?

- É a técnica utilizada para restringir o acesso a elementos de um objeto utilizando qualificadores.



Temos acesso a todas as funcionalidades de uma calculadora?

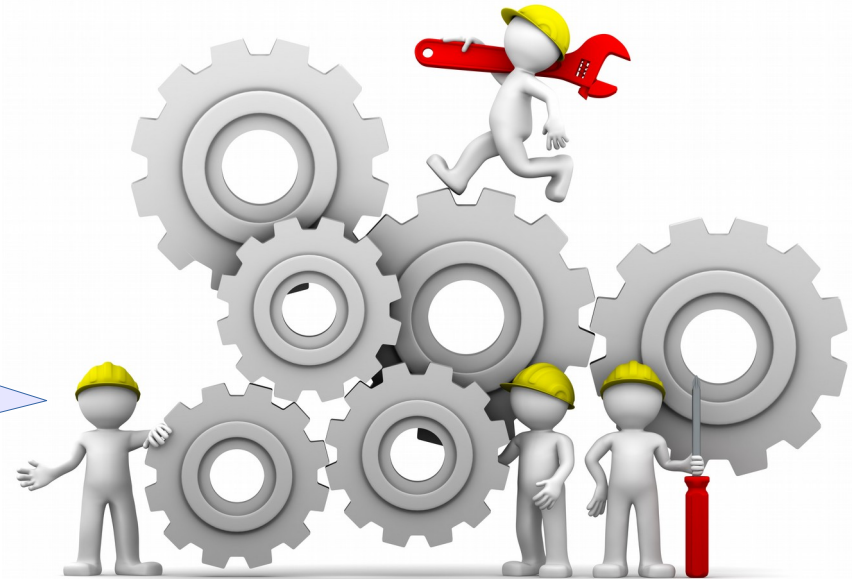
Como imaginam que é realizada uma soma entre dois números?

Qual a técnica utilizada pela calculadora para simplificar a vida do usuário?

# Encapsulamento

- Os detalhes da implementação ficam ocultos ao usuário da classe, ou seja, o usuário passa a utilizar os métodos de uma classe sem se preocupar com detalhes sobre como o método foi implementado internamente.

Para dirigir um carro o motorista não precisa saber como o motor e o câmbio funcionam, apenas deve conhecer as formas de interação com esses elementos para cumprir sua tarefa.



# Encapsulamento

---

- A ideia do encapsulamento na programação orientada a objetos é que não seja permitido acesso direto as propriedades de um objeto.
- Operasse um objetos sempre por meio dos métodos pertencentes a ele.
- A complexidade de um objeto é escondida, portanto, pela **abstração de dados** que estão “por trás” de suas operações.





# Qualificadores de Acesso

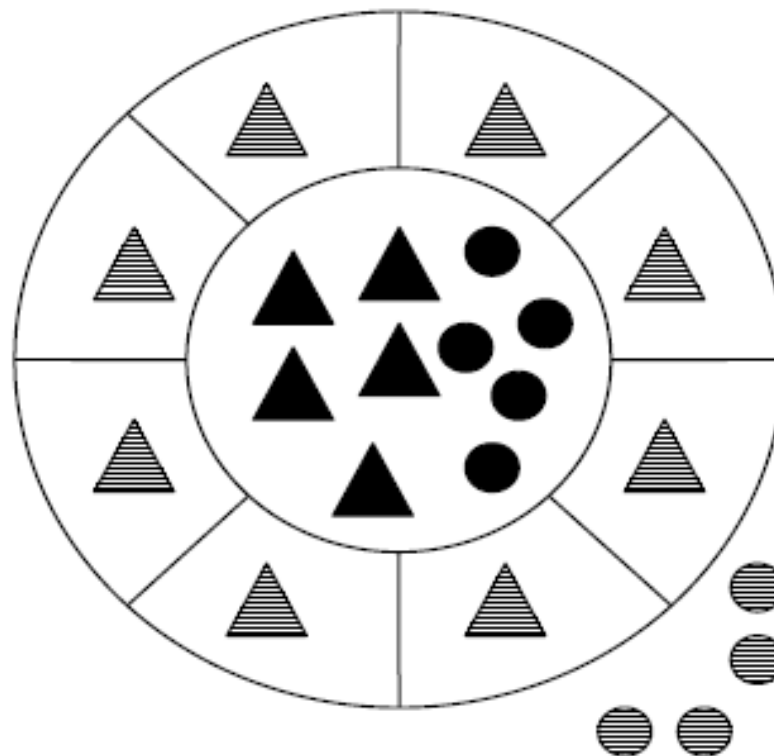
---

- **private:** o método ou atributo pode ser acessado somente dentro da própria classe;
- **public:** o método ou atributo pode ser acessado externamente por outro código;
- **protected:** o método ou atributo pode ser acessado pela própria classe ou por classes-filhas (herança);
- **package:** o método ou atributo pode ser acessado pela própria classe ou classes que participem do mesmo pacote.

# Encapsulamento

## CLASSE

-  métodos públicos
-  métodos privados
-  dados privados
-  dados públicos  
(não recomendável)



# O que são Construtores?

- Construtores são métodos especiais chamados no processo de instanciação de um objeto no sistema.
- A execução destes métodos garante a inicialização dos identificadores de forma correta.
- Um método construtor tem o mesmo nome da classe.





# Exemplo

---

- Para exemplificar a construção de uma classe, iremos construir uma classe que represente uma data.
- Sabemos que uma data é representada por três valores inteiros (dia, mês e ano) que são os atributos da classe.
- A seguir a definição da classe MinhaData com estes atributos.

# MinhaData

```
public class MinhaData {  
    private int dia;  
    private int mes;  
    private int ano;  
  
    public MinhaData(int oDia, int oMes, int oAno){  
        dia = oDia;  
        mes = oMes;  
        ano = oAno;  
    }  
  
    public String dataBrazil(){  
        String txt;  
        txt = dia + "/" + mes + "/" + ano;  
        return txt;  
    }  
  
    public String dataUS(){  
        String txt;  
        txt = ano + "-" + mes + "-" + dia;  
        return txt;  
    }  
}
```

A primeira observação é que “toda” classe é pública, caso contrário a JVM não terá acesso à classe.

Por questão de encapsulamento e segurança, todos os atributos estão bloqueados para acesso de outros códigos. Apenas os métodos da própria classe podem acessar e/ou modificar os dados.

O método construtor é encarregado de passar os dados ao objeto. É importante observar a ordem dos argumentos que são passados para o construtor. Observem que não estamos fazendo qualquer verificação nestes dados.

Implementação de dois métodos que retornam uma String com a data.

# MinhaData

```
public class MinhaData {  
    private int dia;  
    private int mes;  
    private int ano;  
  
    public MinhaData(int oDia, int oMes, int oAno){  
        dia = oDia;  
        mes = oMes;  
        ano = oAno;  
    }  
  
    public String dataBrazil(){  
        String txt;  
        txt = dia + "/" + mes + "/" + ano;  
        return txt;  
    }  
  
    public String dataUS(){  
        String txt;  
        txt = ano + "-" + mes + "-" + dia;  
        return txt;  
    }  
}
```

A variável txt é declarada em dois métodos. Seria ela um atributo da classe MinhaData?



# Main

```
public class Main {  
  
    public static void main(String[] args) {  
        MinhaData hoje;  
  
        hoje = new MinhaData(27, 07, 2018);  
  
        System.out.println("Hoje no Brasil.: " + hoje.dataBrazil());  
        System.out.println("Hoje nos EUA...: " + hoje.dataUS());  
    }  
}
```

A classe Main **hospeda** o método main(), o qual inicia nosso sistema.

Foi instanciado o objeto hoje a partir da classe MinhaData. No processo de instanciar o objeto, foi enviada uma mensagem ao método construtor da classe com os argumentos exigidos.

Após a instancia do objeto, os métodos que exibem a data foram chamados.

# MinhaData - Melhorar

- A classe MinhaData está muito simples, implemente as seguintes melhorias:
  - Não permitir que uma data inválida seja instanciada. Caso tente-se instanciar uma data inválida instancie 01/01/1900.
  - Implemente um construtor que aceite várias ordens de argumentos para instanciar uma data. Por exemplo, o mesmo construtor deve receber os seguintes dados: (1, 2, 1900) ou (1900, 2, 1) e instanciar a data 01/02/1900.
  - Implementar um método que retorne a data por extenso (Brasil).

---

# Métodos de Acesso e Modificadores

# Exemplo – Reajuste Salarial

---

- Vamos implementar um programa que leia nome, salário e valor do reajuste salarial.
- Os dados devem ser inseridos em um objeto.
- O programa deve apresentar o salário reajustado.

# Classe Funcionario

```
public class Funcionario {  
  
    private String nome;  
    private double salario;  
    private int reajuste;  
  
    public Funcionario(String argNome, double argSalario, int argReajuste) {  
        nome = argNome;  
        salario = argSalario;  
        reajuste = argReajuste;  
    }  
  
    public double salarioReajustado() {  
        salario *= 1 + (reajuste / 100.0);  
        return salario;  
    }  
}
```



Métodos de acesso, falar iremos.

Os atributos do objeto são private e por isso inacessíveis. Como recuperar os dados depois de instanciar um objeto?

O que há de errado com esse encapsulamento dessa classe?





# Métodos de acesso e modificadores

---

- `getXxx()`
  - Para recuperarmos o valor de um atributo, **que pode ser recuperado**, utilizamos os métodos `gets()`.
  - Nem todos os atributos possuem `get()` ou método `get()` público.
- `setXxx()`
  - Ao contrário do `get()` o método `set()` altera o valor do atributo.
  - Esse método é responsável pela validação e/ou formatação dos valores que serão inseridos nos atributos de nosso objeto.

Os métodos de acesso podem ser utilizados fora da classe e pelos próprios métodos da classe.

Por exemplo, seria inconsistente se ao criarmos um objeto o nome fosse minúsculo e ao alterar o nome ele fosse maiúsculo. Mas não precisamos repetir código :-D



```
public class Funcionario {
    private String nome;
    private double salario;
    private int reajuste;

    public Funcionario(String argNome, double argSalario, int argReajuste) {
        nome = argNome;
        salario = argSalario;
        reajuste = argReajuste;
    }

    public double salarioReajustado() {
        salario *= 1 + (reajuste / 100.0);
        return salario;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String argNome) {
        nome = argNome.toUpperCase();
    }

    public double getSalario() {
        return salario;
    }

    public void setSalario(double argSalario) {
        salario = argSalario;
    }

    public int getReajuste() {
        return reajuste;
    }

    public void setReajuste(int argReajuste) {
        if (argReajuste >= 0)
            reajuste = argReajuste;
        else
            reajuste = 0;
    }
}
```

```
public Funcionario(String argNome, double argSalario, int argReajuste) {  
    setNome(argNome);  
    setSalario(argSalario);  
    setReajuste(argReajuste);  
}
```

Ao utilizar os métodos set() temos uma melhor organização do sistema, e melhor **manutenibilidade**.



# Fazendo funcionar!

```
public static void main(String args[]){
    Scanner scanner;
    Funcionario funcionario;
    String nome;
    double sal;
    int reaj;
    scanner = new Scanner(System.in);

    System.out.println("Nome: ");
    nome = scanner.nextLine();
    System.out.println("Salario: ");
    sal = scanner.nextDouble();
    System.out.println("Reajuste: ");
    reaj = scanner.nextInt();
```

```
funcionario = new Funcionario(nome, sal, reaj);
```

```
System.out.println("Salario reajustado de " + nome + ": " + funcionario.salarioReajustado());
System.out.println("Salario reajustado de " + nome + ": " + funcionario.salarioReajustado());
System.out.println("Salario reajustado de " + nome + ": " + funcionario.salarioReajustado());
```

```
}
```

Aqui é instanciado um objeto Funcionário, esse tem seus atributos (características) e funcionalidades.

```
public class Funcionario {  
  
    private String nome;  
    private double salario;  
    private int reajuste;  
  
    public Funcionario(String argNome, double argSalario, int argReajuste) {  
        setNome(argNome);  
        setSalario(argSalario);  
        setReajuste(argReajuste);  
    }  
  
    public double salarioReajustado() {  
        salario *= 1 + (reajuste / 100.0);  
        return salario;  
    }  
  
    //...  
}
```

Esse método que retorna o  
salário reajustado está uma  
graça!



```
funcionario = new Funcionario(nome, sal, reaj);  
System.out.println("Salario reajustado de " + nome + ": " + funcionario.salarioReajustado());  
System.out.println("Salario reajustado de " + nome + ": " + funcionario.salarioReajustado());  
System.out.println("Salario reajustado de " + nome + ": " + funcionario.salarioReajustado());
```

```
Nome:  
Gustavo  
Salario:  
1000  
Reajuste:  
10  
Salario reajustado de Gustavo: 1100.0  
Salario reajustado de Gustavo: 1210.0  
Salario reajustado de Gustavo: 1331.0
```

Sempre que invoca-se o método para recuperar o salário reajustado o salário é reajustado novamente.

Pense, reajustar o salário é o mesmo que recuperar o salário do objeto Funcionário?



```
public class Funcionario {
```

```
    private String nome;  
    private double salario;  
    private int reajuste;
```

```
    public Funcionario(String argNome, double argSalario, int argReajuste) {  
        setNome(argNome);  
        setSalario(argSalario);  
        setReajuste(argReajuste);  
    }
```

```
    public void reajustarSalario() {  
        salario *= 1 + (reajuste / 100.0);  
    }  
  
    //...  
}
```

```
Nome:  
Gustavo  
Salario:  
1000  
Reajuste:  
10  
Salario reajustado de Gustavo: 1100.0  
Salario reajustado de Gustavo: 1100.0  
Salario reajustado de Gustavo: 1100.0
```

```
funcionario = new Funcionario(nome, sal, reaj);  
funcionario.reajustarSalario();  
System.out.println("Salario reajustado de " + nome + ": " + funcionario.getSalario());  
System.out.println("Salario reajustado de " + nome + ": " + funcionario.getSalario());  
System.out.println("Salario reajustado de " + nome + ": " + funcionario.getSalario());
```



#90707984

Reajustar o salário é uma ação que ocorre de tempo em tempos com o objeto, não uma ação corriqueira.

Além disso, o reajuste mudará (quase) sempre que o reajuste for aplicado, assim, não faz sentido que o reajuste seja um atributo do objeto funcionário.

# Material Adicional

- **Leitura Obrigatória:**

- Unidade 1 - Programação orientada a objetos / organizador Rafael Félix. - São Paulo: Pearson, 2016

- **Videoaulas**

- <https://www.youtube.com/watch?v=g2x9oyBFSCO>
- [https://www.youtube.com/watch?v=6i-\\_R5cAcEc](https://www.youtube.com/watch?v=6i-_R5cAcEc)





# Trabalhando

---

- **Exercícios Avaliativos**
  - 2, 3 e 4

