

# Disciplina Regular 1

## Fundamentos do

# Desenvolvimento Java

Graduação em Engenharia de Software - 2020

# Etapa 3 Aula 1

Fundamentos de Orientação a Objetos

# Competências Trabalhadas Nesta Etapa

- Escrever programas em Java com orientação a objetos:
  - Construir classes, atributos e métodos.
  - Criar instâncias e realizar chamadas de métodos.
  - Escrever construtores.
  - Implementar sobrecargas de construtores e métodos.

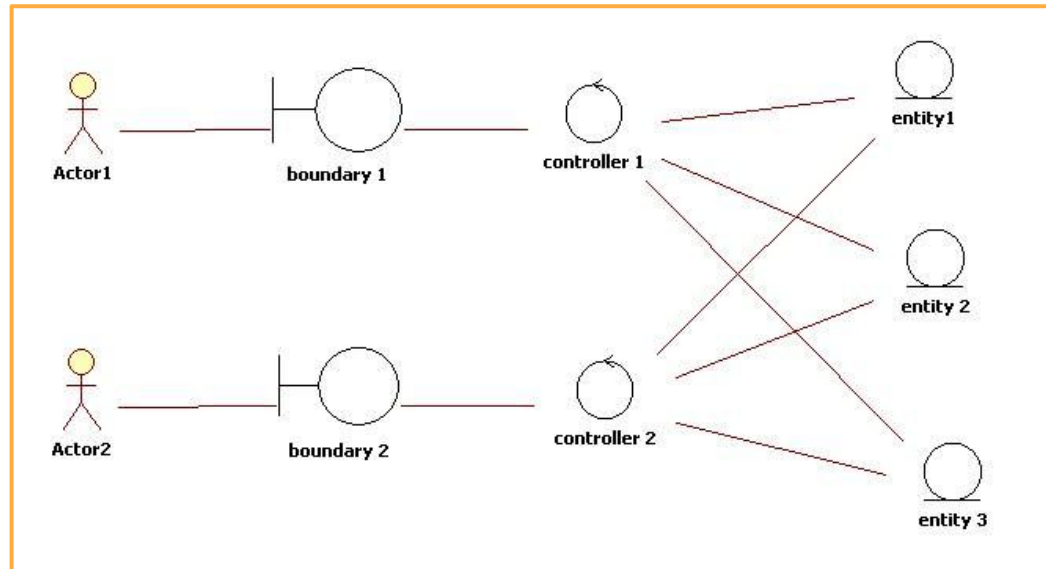
# Conceitos Básicos

# Classes

- Conceito de Classe:
  - Classe é a descrição de uma entidade existente no domínio do problema.
  - É uma “fábrica de objetos”.
  - Define a forma e a funcionalidade de objetos.
- Responsabilidades da Classe:
  - É o que a classe “sabe” e o que ela “faz”.
  - O que a classe “sabe” são as **propriedades** ou seus atributos.
  - O que a classe “faz” são os seus **métodos** ou funções.

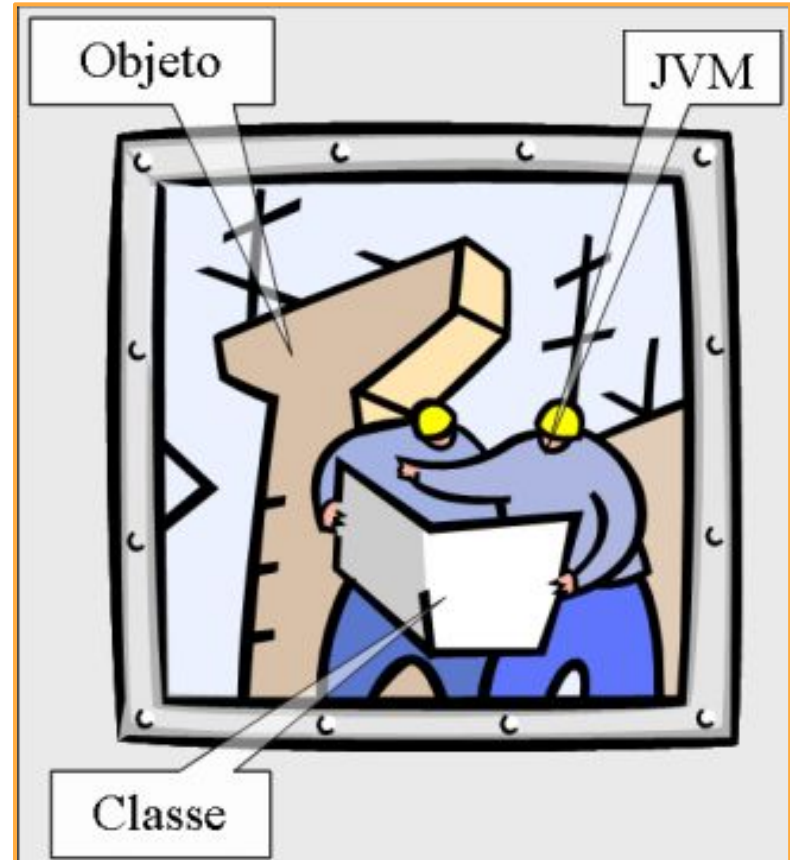
# Classes

- Classes de Entidade:
  - Cliente,
  - Pedido,
  - Item de Pedido,
  - Produto etc.
- Classes de Fronteira:
  - Botões,
  - Checkboxes,
  - Acesso a outros sistemas.
  - Webservices.
- Classes de Controle:
  - Data,
  - Conexão com Banco de Dados,
  - Gerenciador de Impressão,
  - Leitura e Gravação de Arquivos.



# Objetos

- Objeto é o conjunto de dados e métodos criado a partir da classe.
- Um objeto ocupa seu próprio lugar na memória e não interfere em outros objetos.
- A criação de um objeto também é chamada de instanciação.



# Objetos

- O operador ***new*** aloca espaço, cria um objeto da classe e retorna o seu endereço de memória (sua referência).

```
ContaCorrente cc;//Declaração da variável cc do tipo ContaCorrente  
cc = new ContaCorrente();//A referência é atribuída a cc
```

```
ContaCorrente cc = new ContaCorrente();//dois passo em uma linha
```



# Encapsulamento

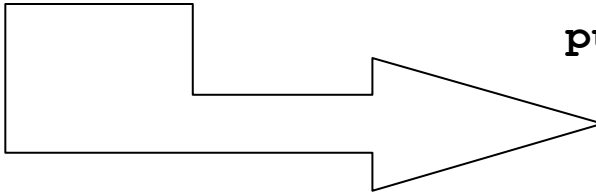
- Encapsulamento:
  - Agrupar em uma classe métodos e propriedades relacionadas a alguma coisa do mundo real.
  - Facilita a manutenção.
- Ocultamento:
  - Esconder como os objetos funcionam internamente e o que eles sabem atrás de uma *interface* (conjunto de métodos).
  - Através dos modificadores de visibilidade temos como obter o **encapsulamento** e o **ocultamento**.

# Encapsulamento

- A Classe é um “módulo”, contendo dados (propriedades) e operações (métodos).
- O objeto é referenciado como um módulo único.

```
public class Data
{
    ...
}

public class UsaData {
    public static void main(String [] args) {
        Data hoje = new Data();
        ...
        Data ontem = new Data();
        ...
    }
}
```



# Encapsulamento

```
public class Funcionario {
```

```
protected int matricula;  
protected String nome;  
protected String departamento;  
...
```

Propriedades

```
public int getMatricula() {  
    return matricula;  
}
```

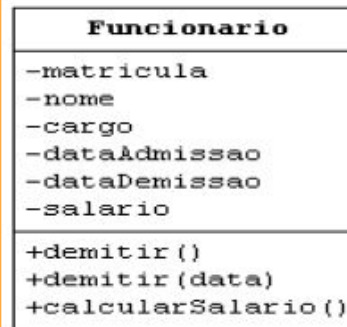
Métodos

```
public void setMatricula(int novaMatricula) {  
    matricula = novaMatricula;  
}
```

```
...
```

```
}
```

Herança

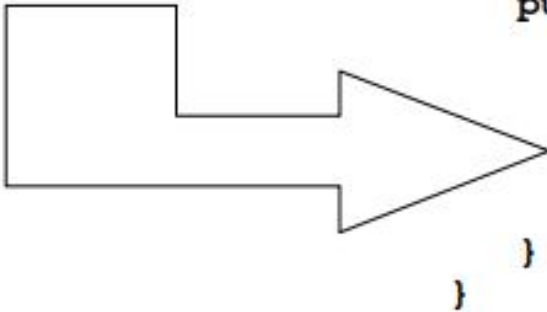


# Propriedades e Métodos

# Propriedades

- As propriedades são os dados que guardam as características e o estado dos objetos criados a partir da classe.
- É o que a classe “sabe”.

```
public class Data {  
    ...  
}
```

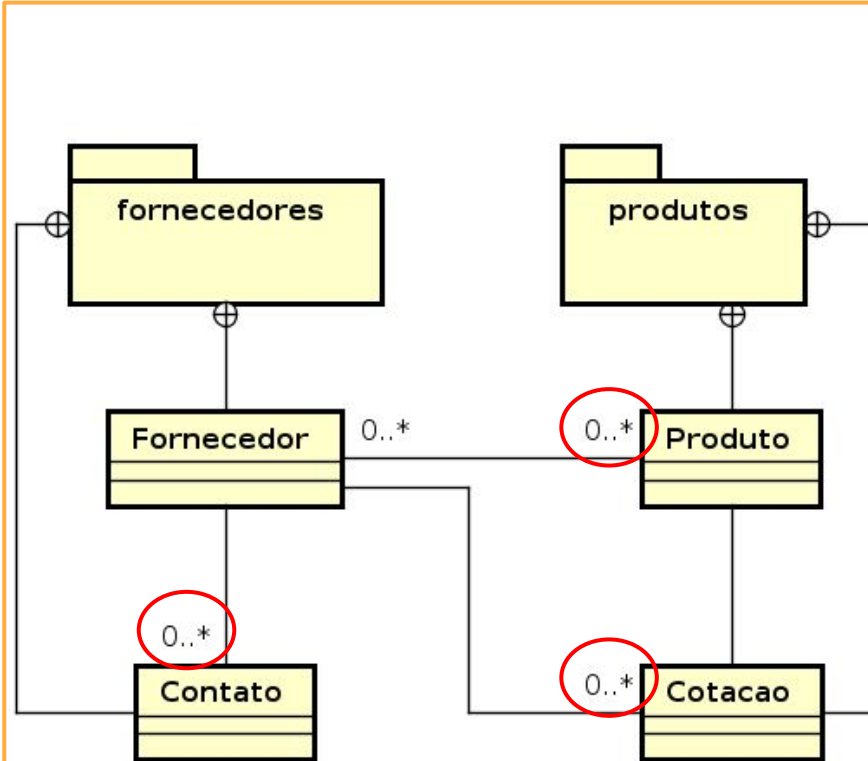


```
public class UsaData {  
    public static void main(String [] args) {  
        Data hoje = new Data();  
        ...  
        Data ontem = new Data();  
        ...  
    }  
}
```

# Propriedades

- As propriedades guardam valores para atributos de uma classe de objetos.
- Em geral, as propriedades são de tipos primitivos, porém podem ser de tipos complexos, dependendo do relacionamento entre as classes envolvidas.
- Por exemplo, Fornecedor é uma classe que se relaciona com Produto. Por esse motivo, é possível que tenha uma propriedade que represente os produtos fornecidos.

# Propriedades



powered by Astah

```
public class Fornecedor {  
  
    private Long id;  
    private String nomeFantasia;  
    private String razaoSocial;  
    private String cnpj;  
    private Endereco endereco;  
    private String tipoInscricao;  
    private String inscricao;  
    private String observacao;  
    private List<Contato> contatos;  
    private List<Produto> produtos;  
    private List<Cotacao> cotacoes;  
  
    public Long getId() {  
        return id;  
    }  
  
    public void setId(Long id) {  
        this.id = id;  
    }  
  
    public String getNomeFantasia() {
```

# Propriedades de Instância

- Propriedades de Instância são aquelas declaradas dentro da definição da classe. Todo objeto da classe possui uma propriedade própria.
- São “dinâmicas”.

```
public class Empregado {  
    private int matricula;  
    private String nome;  
    private String cargo;  
    ...  
}
```



# Propriedades de Classe

- Propriedades de classe são aquelas compartilhadas por todas as instâncias da classe.
- São estáticas.

```
public class Vendedor {  
    private int matricula;  
    private String nome;  
    private String cargo;  
    private static double comissao = 0.15;  
}
```

Todos os  
vendedores têm  
o mesmo  
percentual de  
comissão

# Métodos

- O termo Método representa as funcionalidades inerentes à classe.
- É o que a classe “faz”.

```
public class Data {  
  
    private int dia;  
    private int mês;  
    private int ano;  
  
    public void mudaDia(int novoDia) {  
        if(novoDia >= 1 && novoDia <=31) {  
            dia = novoDia;  
        }  
    }  
}
```

# Métodos

- O programador vê os métodos sob dois aspectos:
  - **Declaração**: quando o programador deseja criar um método para implementar alguma operação.
  - **Utilização**: quando o programador conhece a sintaxe de um método e deseja usá-lo nos seus programas.
    - Lembre-se das bibliotecas de classes da UA 1.

# Métodos de Instância

- Métodos de Instância são aqueles declarados dentro da definição da classe. Individualmente, todo objeto possui o seu.
- São “dinâmicos”.

```
public class Empregado {...  
    public void defineIdade(int campo) {  
        idade = campo;  
    }  
    public int retornaIdade() {  
        return idade;  
    }  
}
```

# Métodos de Classe

- Método de Classe é aquele compartilhado por todas as instâncias. Dispensa a criação do objeto.
- São estáticos.

```
public class Vendedor {  
    ...  
    public static void defineComissao(double valor) {  
        comissao = valor;  
    }  
}
```

```
Vendedor.defineComissao(0.2);
```

# Sobrecarga de Métodos

- Ocorre quando existe mais de um método com o mesmo nome em uma classe.
- Para que isso seja possível, os métodos devem ter assinaturas diferentes .
- “**Assinatura**” - formada pelo nome do método, do número e dos tipos de parâmetros.

# Sobrecarga de métodos

```
public void defineCampo(int valor){
    idade = valor;
}
public void defineCampo(String valor){
    nome = valor;
}
public int defineCampo(int valor1, String valor2){
    idade = valor1;
    nome = valor2;
}
```

```
//Exemplo de chamada de método sobrecarregado
...
emp.defineCampo(32);
emp.defineCampo("Maria");
emp.defineCampo(40, "Roberto");
...
emp.defineCampo(12f); // Erro!!!!
```

# Construtores

- São métodos especiais que as classes possuem e que **servem para executar inicializações**.
- Devem, obrigatoriamente, ter o mesmo nome da classe a qual pertencem. Não possuem tipo de retorno.
- Usam o conceito de sobrecarga de método.
- Toda classe Java tem, pelo menos, um construtor.
- Chama-se construtor default qualquer construtor sem parâmetros.



# Construtores

```
public class ContaCorrente {  
    public ContaCorrente(int novoNúmero, String novoTitular) {  
        número = novoNúmero;  
        titular = novoTitular;  
        senha = "";  
        saldo = 0.0;  
    }  
  
    public ContaCorrente(int novoNúmero, String novoTitular, double novoSaldo)  
    {  
        número = novoNúmero;  
        titular = novoTitular;  
        saldo = novoSaldo;  
        senha = "";  
    }  
}
```

# Referência this

- Refere-se ao objeto atual sobre o qual o método foi chamado.
- Obrigatória quando:
  - Os nomes dos parâmetros forem iguais aos nomes das propriedades da classe (usado para diferenciar).

```
public ContaCorrente(int número, String titular,  
                     String senha, double saldo) {  
    this.número = número;  
    this.titular = titular;  
    this.saldo = saldo;  
    this.senha = senha;  
}
```

# Referência this

- Também pode ser usado para chamar outro construtor dentro da mesma classe.
- Esta operação só é válida dentro de construtores - não é possível usar em métodos comuns.

```
public ContaCorrente(int número, String titular, String senha, double saldo) {  
    this.número = número;  
    this.titular = titular;  
    this.saldo = saldo;  
    this.senha = senha;  
}  
  
public ContaCorrente(int número, String titular) {  
    this(número, titular, "", 0.0);  
}
```