

# Modelagem e Desenvolvimento Orientado a Objeto



# Modelagem e Desenvolvimento Orientado a Objeto

## Plano de aula



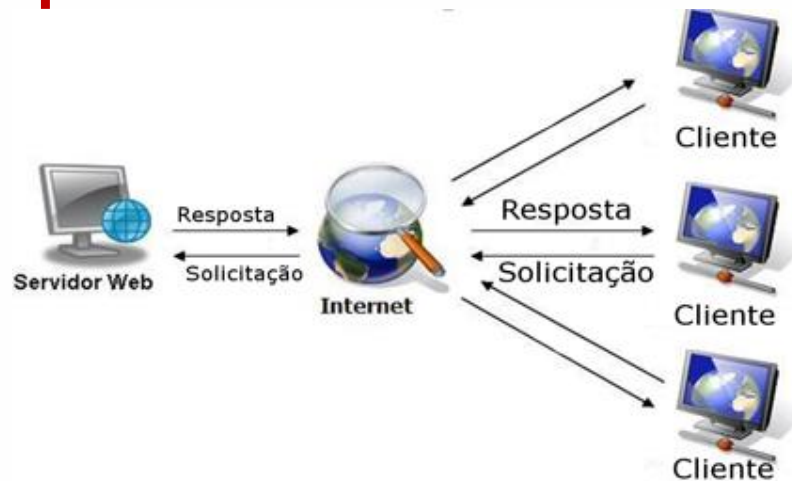
### PROGRAMAÇÃO ORIENTADA A OBJETOS

- Introdução ao Java
- Introdução à Programação Orientada a Objetos com Java
- Tipos de dados em Java
- Estruturas de Controle de Fluxo e Laço em Java
- Arrays e Coleções de Dados
- Gerando e Interceptando Exceções
- Herança e Polimorfismo
- Abstração de dados
- Encapsulamento
- Organização em Pacotes
- Conexão com Bases de Dados
- Conceitos Básicos de GUI em Java
- Lidando com Eventos de GUI em Java
- Usando Gerenciadores de Layout
- Construção de GUI em Java
- Projeto de GUI conectada a Bases de Dados
- Projeto de um Applet Java



# Modelagem e Desenvolvimento Orientado a Objeto

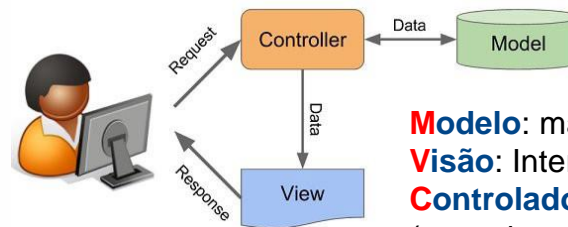
## Arquitetura Cliente / Servidor



### Arquitetura em Camadas



### Arquitetura MVC



**Modelo:** manipulação de dados.  
**Visão:** Interface gráfica do usuário.  
**Controlador:** tratamento de dados (pesquisar, listar, calcular, etc.)  
Interpretação de eventos.

CRUD (Create, Read, Update, Delete)

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS



Integrated Development Environment – IDE  
(Ambiente de Desenvolvimento Integrado)



[netbeans.org](https://netbeans.org)

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Implementação



NomeDaClasse
+ PropriedadeClasse_01 : int + PropriedadeClasse_02 : String + PropriedadClasse_03 : float
+ MetodoDaClasse_01() : void + MetodoDaClasse_02() : int + MetodDaClasse_03(parametro01 : int) : float

2 Defina o nome da classe, representada por um substantivo, com a primeira letra em maiúsculo.

3 As propriedades são definidas no segundo bloco da classe. Cada propriedade pode ter um tipo de dado específico, inclusive referenciado a outra classe. A propriedade também é representada por um substantivo.

5 Os sinais representam o encapsulamento de cada atributo / método da classe.

NomeDaClasse
+ PropriedadeClasse_01 : int + PropriedadeClasse_02 : String + PropriedadClasse_03 : float
+ MetodoDaClasse_01() : void + MetodoDaClasse_02() : int + MetodDaClasse_03(parametro01 : int) : float

4 Os métodos são definidos no terceiro bloco da classe. Assim como as funções existentes na Análise Estruturada, eles podem retornar ou não algum resultado e podem ter parâmetros recebidos. Os métodos, por serem ações possíveis da classe, são representados por um verbo.

6 Produto
+CodigoBarras : int + Descrição : String + Preço : float
+ Imprimir() : void + ObterQuantidadeEstoque() : int + CalcularSubTotal(quantidade : int) : float

```
class Produto
{
    public int CodigoBarras { get; set; }

    public string Descricao { get; set; }

    public float Preco { get; set; }

    public void Imprimir ()
    {
        // TODO - Escrever o método de impressão
    }

    public int ObtemQuantidadeEstoque()
    {
        // TODO - Escrever o método para obter a quantidade em estoque
        return 0;
    }

    public float CalcularSubTotal(int quantidade)
    {
        return quantidade * Preco;
    }
}
```

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Arquitetura Java



### IDE (NetBeans)

JDK (Java Development Kit) – Ferramentas e Compilador

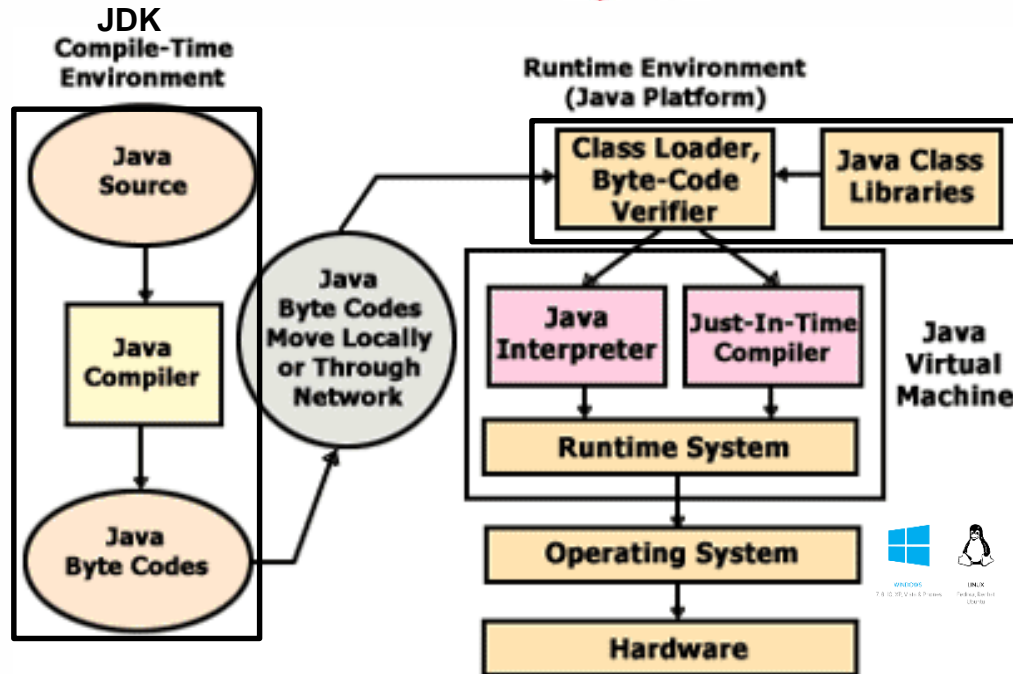
JRE (Java Runtime Machine) - Bibliotecas

JVM (Java Virtual Machine) - Execução



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Arquitetura Java



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS



### The Java™ Platform





# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Tipos de Dados - Variáveis



Ambiente de desenvolvimento

Classe: Pessoa

Nome: texto

Cor dos olhos: texto

Cor do cabelo: texto

Sexo: caracter

Altura: real

Peso: real

Tipo

←  
Classe

Programa em execução

Objeto1: pessoa

nome:Alfie

cor dos olhos: Pretos

cor do cabelo: Azul

sexo: M

altura:1,80

peso:100

Objeto2: pessoa

nome:Josefina

cor dos olhos: Amarelo

cor do cabelo: Magenta

sexo: F

altura:3,50

peso:180

←  
Instâncias

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Variáveis



- Espaço de memória de um certo tipo de dado associado a um nome para referenciar seu conteúdo.

### Nomes de Variáveis ou Constantes

- iniciar com letras:  
Seguidos de letras, números ou traço baixo  
float peso;  
peso = 81.5;

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS



### Tipos de Dados

Grupo	Tipo primitivo	Bytes	Exemplo
String	Char	2	Char sexo= 'M';
Inteiros	Byte	1	Byte idade=55;
	Short	2	Short x=3456;
	Int	4	Int y= 678934;
	Long	8	Long cod=1756453;
Reais	Float	4	Float pi=3.1415F;
	Double	8	Double valor=34.56;
Boolean	Boolean	1	Boolean casado=true;
	Tipo referência	Classes	
String	String	String nome = new String();	
Date	Date	Date dtnasc = new Date();	



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Funções de Conversão de Dados



#### Função

`Integer.parseInt( )`

`Long.parseLong( )`

`Float.parseFloat( )`

`Double.parseDouble( )`

`String.valueOf()`

`DecimalFormat()`

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Conversão de Dados - Casting



DE \ PARA	byte	short	char	int	long	float	double
byte		Implícito	char	Implícito	Implícito	Implícito	Implícito
short	byte		char	Implícito	Implícito	Implícito	Implícito
char	byte	short		Implícito	Implícito	Implícito	Implícito
int	byte	short	char		Implícito	Implícito	Implícito
long	byte	short	char	int		Implícito	Implícito
float	byte	short	char	int	long		Implícito
double	byte	short	char	int	long	float	

Casting feito **implicitamente**, pois o valor possui um tamanho menor que o tipo da variável que irá recebe-lo.

```
char a = 'a';           a
int b = 'b';            98
float c = 100;          100.0
```

Casting feito **explicitamente**, pois o valor possui um tamanho maior que o tipo da variável que irá recebe-lo.

```
int d = (int) 5.1987;    5
float e = (float) 5.0;   5.0
int f = (char) (a + 5);  102
char g = (char) 110.5;   n
```

# Modelagem e Desenvolvimento Orientado a Objeto

PROGRAMAÇÃO ORIENTADA A OBJETOS  
Processamento de Dados

Entrada

Processamento

Saída

Dados

Transformação

Informação



Operações Aritméticas  
Operações Relacionais  
Operações Lógicas



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Operadores Aritméticos



Prioridade	Operador	Operação
1	( )	Alteração Prioridade
2	Math.pow()	Potenciação
3	/	Divisão
3	*	Multiplicação
3	%	Resto
4	+	Soma
4	-	Subtração

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Operadores Relacionais



Operador	Ação
>	Maior
<	Menor
>=	Maior igual
<=	Menor igual
==	Igual
!=	diferente

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Operadores Lógicos



Operador	Ação
	OU
&&	E
!	NÃO

<condição 1>		<condição 2>	Resultado
V		V	V
V		F	V
F		F	F
<condição 1>	&&	<condição 2>	Resultado
V		V	V
V		F	F
F		F	F

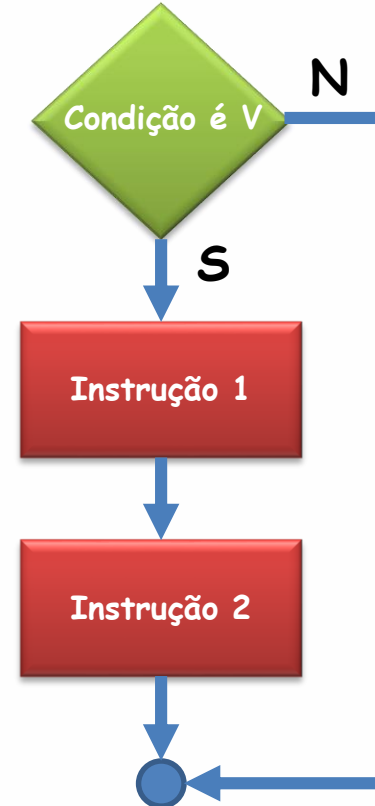
# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Estrutura de Controle Condicional



### Simple

```
if (<condição>)  
{  
    <comando>;  
    <comando>;  
}
```

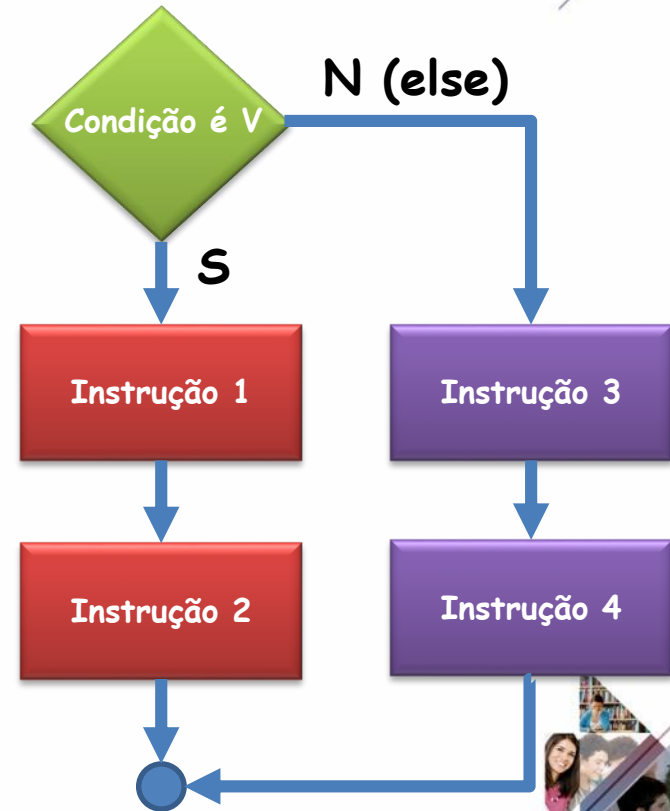


# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Estrutura de Controle Condicional

### Composta

```
if (<condição>)  
{  
    <comando>;  
    <comando>;  
}  
else  
{  
    <comando>;  
    <comando>;  
}
```



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Exercícios de Fixação



#### Exercício 1

**$x = 8$**

**$y = 4$**

**$z = 2$**

**$r = x + y / z$**

**$r = ?$**

#### Exercício 2

**$x = 8$**

**$y = 4$**

**$z = 2$**

**$r = (x + y) / z$**

**$r = ?$**

#### Exercício 3

**$x = 8$**

**$y = 4$**

**$z = 2$**

**$r = z * y - x / z$**

**$r = ?$**



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Exercícios de Fixação



#### Exercício 4

```
x = 8;  
y = 4;  
z = 2;  
if(x > y && y < z)  
    r = x + y / z;  
else  
    r = x - y * z;  
  
r = ?
```

#### Exercício 5

```
x = 8;  
y = 4;  
z = 2;  
if(x > y || y < z)  
    r = x + y / z;  
else  
    r = x - y * z;  
  
r = ?
```

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Exercícios de Fixação



### Exercício 6

```
x = 8;  
y = 4;  
z = 2;  
if(x > y || y > z && z > x)  
    r = x / y + z * y;  
else  
    r = z * x - y / z;  
  
r = ?
```

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Exercícios de Fixação

#### Exercício 7

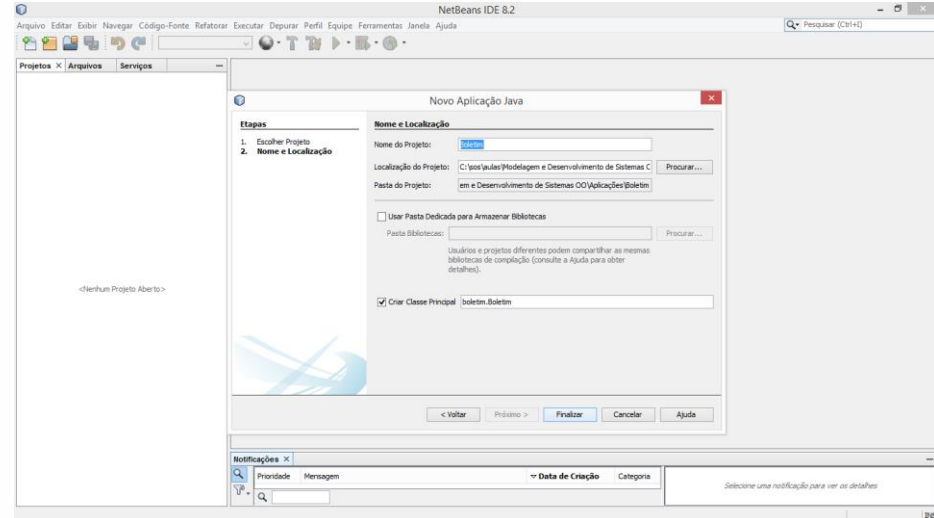
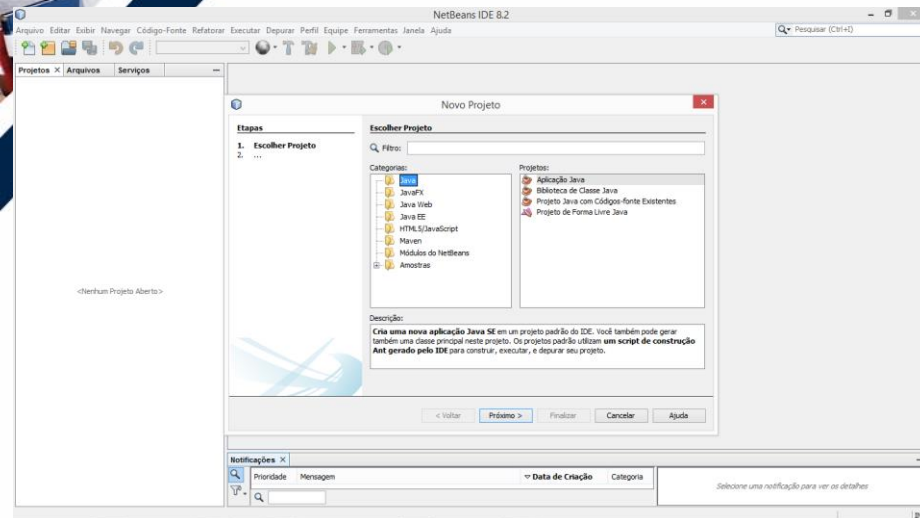


```
int a=2, b=8, c=9, x=0, y=0, z=0, r=0;
if (a < b && b != c && c < a)
    x = 2 * c - b * a;
if (a > c || b < c && a != c)
    y = c - b / a;
if (a < b || b != c && c < a)
    z = b / a - c / 3;
r = x + y + z;

r = ?
```

# Modelagem e Desenvolvimento Orientado a Objeto

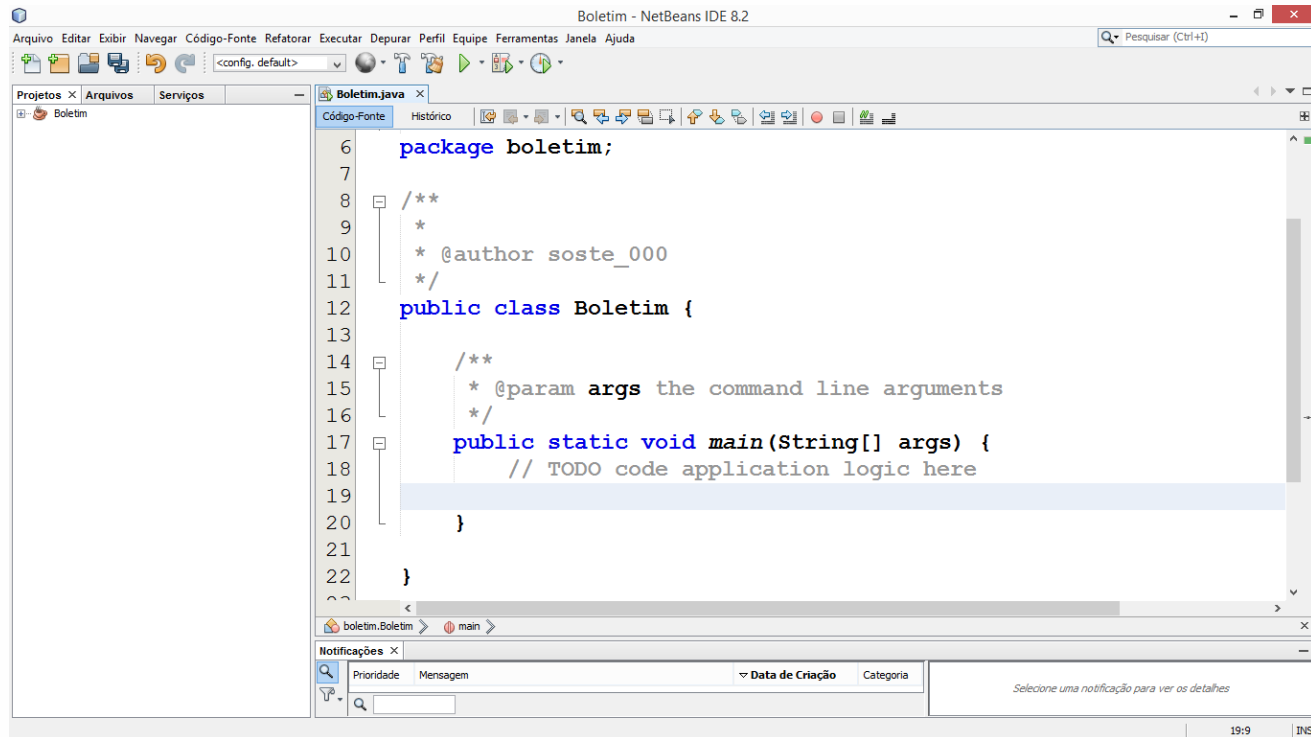
## PROGRAMAÇÃO ORIENTADA A OBJETOS Interface Gráfica - IDE



# Modelagem e Desenvolvimento Orientado a Objeto

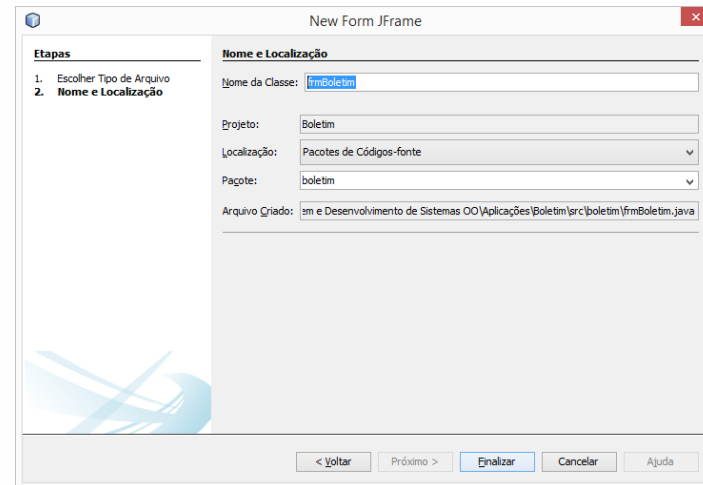
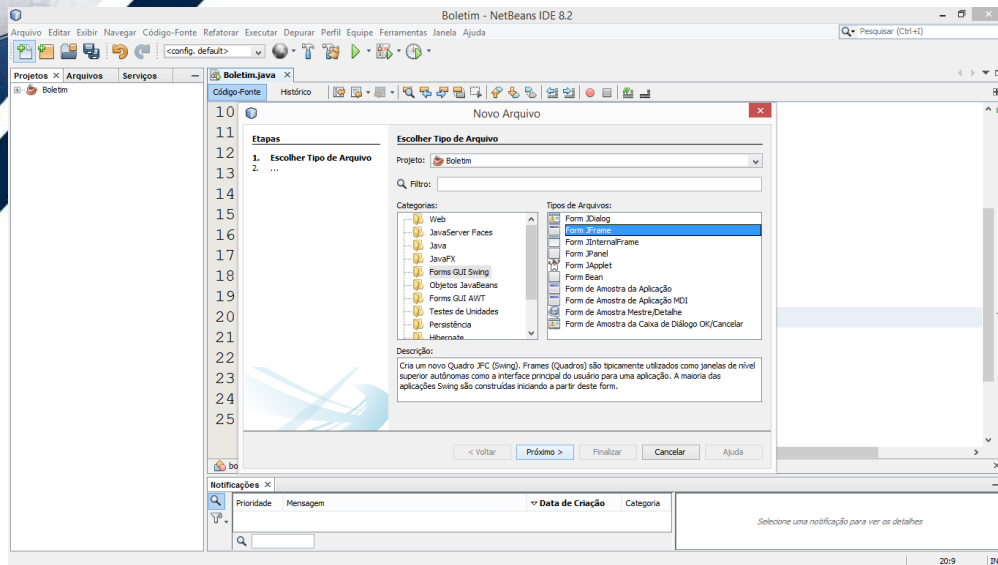
## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Aplicação Boletim - Classe



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Aplicação – Boletim (Jframe)

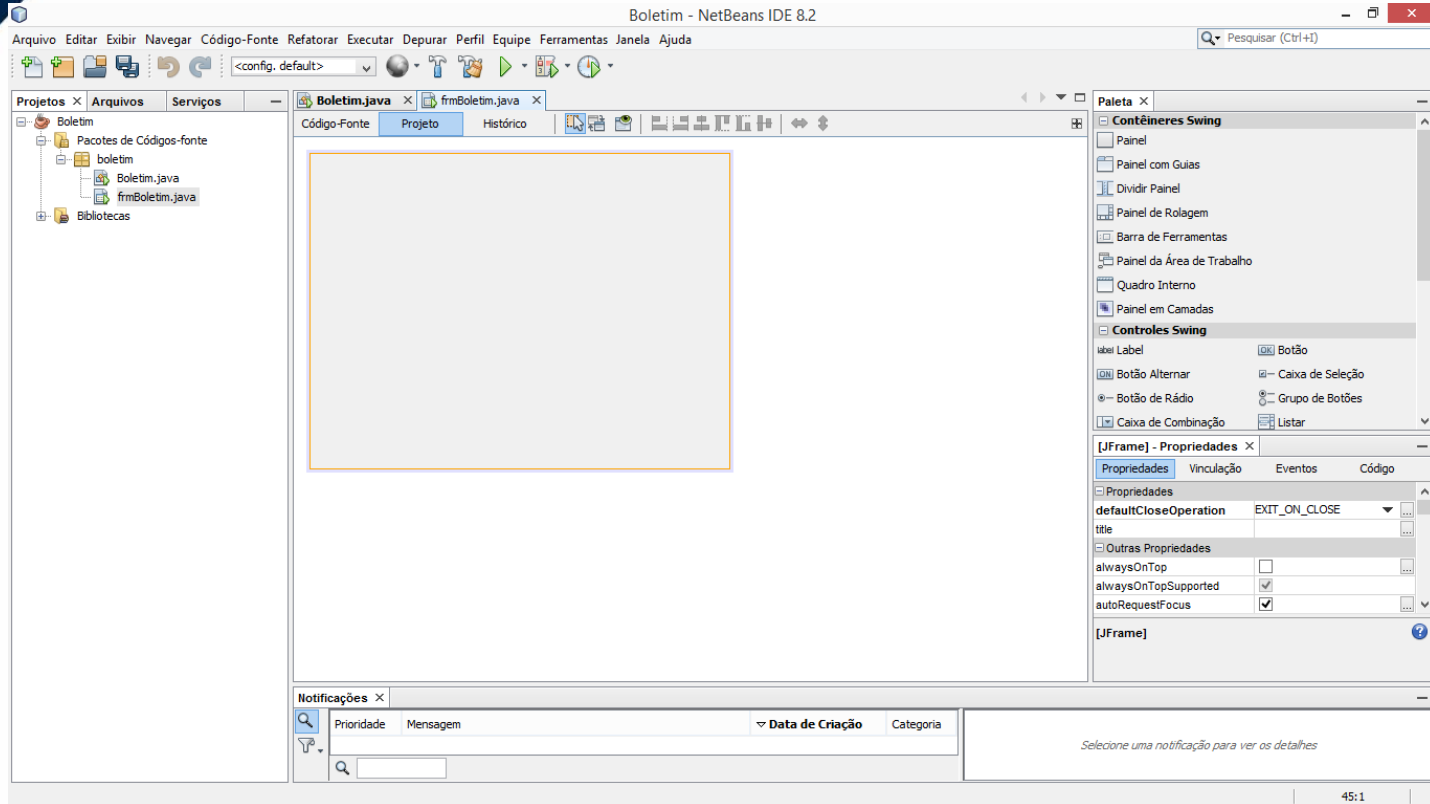




# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Aplicação – Boletim (Jframe)



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Aplicação – Boletim

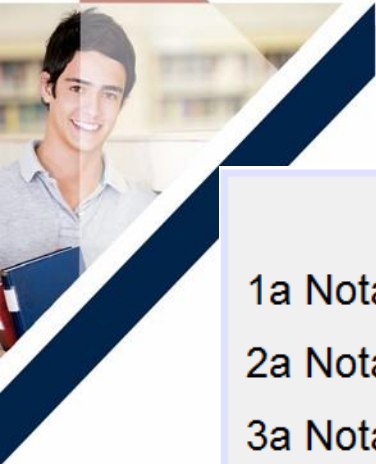


Boletim		
Dados	Descrição	Valor
nota 1	1ª nota	5.0
nota 2	2ª nota	6.0
nota 3	3ª nota	8.0
Media	Média	???
Situação	Parâmetro de aprovação: 6,0	$\geq 6$ Aprovado $\geq 3$ e $< 6$ Exame $< 3$ Reprovado
Fórmula		
$\text{media} = (n1 + n2 + n3) \div 3$		
Exemplo		
$\text{media} = (5.0 + 6.0 + 8.0) / 3$		



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Aplicação – Boletim (Jframe)



**Boletim**

1a Nota	<input type="text"/>
2a Nota	<input type="text"/>
3a Nota	<input type="text"/>
Média	<input type="text"/>
Situação	<input type="text"/>

**Boletim**

1a Nota	<input type="text" value="5.0"/>
2a Nota	<input type="text" value="6.0"/>
3a Nota	<input type="text" value="8.0"/>
Média	<input type="text" value="6,3"/>
Situação	<input type="text" value="Aprovado"/>



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Aplicação – Boletim (Jframe)



### Nomes e Propriedades

#### JLabel

- Font
- Text

#### Boletim

1a Nota

**txtnota1**

2a Nota

**txtnota2**

3a Nota

**txtnota3**

Média

**txtmedia**

Situação

**txtsituacao**

Calcular

**butCalc**

- HorizontalAlignment (Right)
- Text

#### JTextField

- Editable (desmarcar)
- HorizontalAlignment (Right)

- Editable (desmarcar)
- HorizontalAlignment (Center)

#### JButton

- Font
- Text

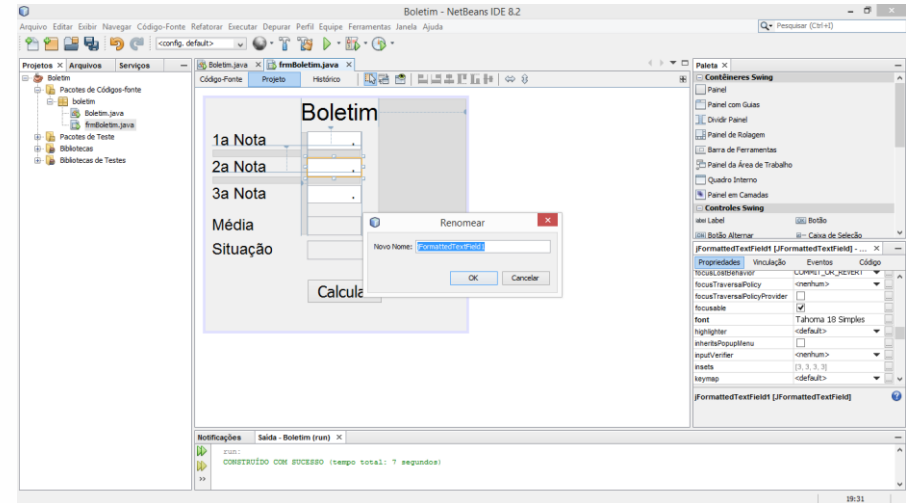
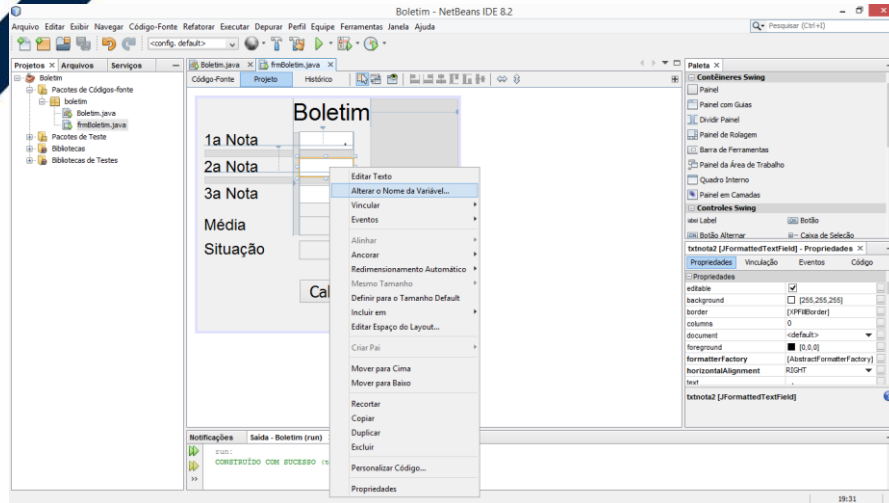
# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Aplicação – Boletim (Jframe)



### Alterar nome da variável

Clicar com botão direito



# Modelagem e Desenvolvimento Orientado a Objeto

PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicação – Boletim (Jframe)

(Diagrama de Classes)



## Boletim

- + nota1: float
- + nota2: float
- + nota3: float
- + media: float
- + situacao: String

---

- + CalcularMedia(): float
- + MostrarSituacao(): String



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicação – Boletim (Jframe)

(Definição de Atributos e Métodos)



### Boletim

+ nota1: float

+ nota2: float

+ nota3: float

+ media: float

+ situacao: String

+ CalcularMedia(): float

+ MostrarSituacao(): String

```
public class Boletim {
```

```
    public float nota1, nota2, nota3, media;
```

```
    public String: situacao;
```

} Atributos

```
    public float CalcularMedia()
```

```
    {
```

```
        return (nota1 + nota2 + nota3) / 3;
```

```
    }
```

} Método

```
    public String MostrarSituacao()
```

```
    {
```

```
        if(media >= 6)
```

```
            situacao = "Aprovado";
```

```
        if(media >= 3 && media < 6)
```

```
            situacao = "Exame";
```

```
        if(media < 3)
```

```
            situacao = "Reprovado";
```

```
        return situacao;
```

```
    }
```

} Método

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicação – Boletim (Jframe)  
(Carregar Formulário)



```
public class Boletim {  
  
    public static void main(String[] args) {  
        // TODO code application logic here  
  
        frmBoletim frm1 = new frmBoletim();  
        frm1.setVisible(true);  
  
    }  
  
}
```

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Aplicação – Boletim (Jframe)



```
import java.text.DecimalFormat;
```

A rectangular button with a blue border and a light blue gradient background. The word "Calcular" is written in a bold, black, sans-serif font in the center.

```
private void butCalcActionPerformed(java.awt.event.ActionEvent evt)
{
    Boletim b = new Boletim();
    b.nota1 = Float.parseFloat(txtnota1.getText());
    b.nota2 = Float.parseFloat(txtnot2.getText());
    b.nota3 = Float.parseFloat(txtnota3.getText());
    DecimalFormat df = new DecimalFormat("#0.0");
    b.media = b.CalcularMedia();
    txtmedia.setText(df.format(b.media));
    txtsituacao.setText(b.MostrarSituacao(b.media));
}
```

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

Aplicação – IMC



### Problema

Dados	Descrição	Valor
altura	Altura (Metros)	1.80
peso	Peso (Kg)	80.0
imc	Índice Massa Corporal	???

### Fórmula

$imc = peso \div (altura)^2$

### Exemplo

$imc = 80.0 \div (1.80)^2$   
 $imc = 24,7$  **Ideal**

**Classificação ???**

### Tabela Índice Massa Corporal

IMC	Classificação
$\leq 18.5$	Abaixo
$> 18.5$ e $< 25$	Ideal
$\geq 25$ e $< 30$	Sobrepeso
$\geq 30$ e $< 35$	Obeso
$\geq 35$ e $< 40$	Obesidade Severa
$\geq 40$	Obesidade Mórbida

## IMC

Peso

txtpeso

Altura

txtaltura

IMC

txtimc

Classificação

txtclassific

Calcular

butCalc

Potenciação: `Math.pow(base, expoente)`

Exemplo: `Math.pow(3, 2) → 9` resultado é tipo double

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Aplicação – IMC



### IMC

- + peso: float
- + altura: float
- + imc: float
- + classific: String

---

- + CalcularImc(): double
- + MostrarClassificacao(): String

# Modelagem e Desenvolvimento Orientado a Objeto



## Carregar Formulário

```
public static void main(String[] args) {
```

```
    frmImc fi = new frmImc();  
    fi.setVisible(true);
```

```
}
```

```
public class Imc {  
    public float peso, altura;  
    public double imc;  
    public String classific;  
  
    public double CalcularImc()  
    {  
        return peso / Math.pow(altura,2);  
    }  
  
    public String MostrarClassificacao()  
    {  
        if(imc <= 18.5)  
            classific = "Abaixo";  
        else  
            if(imc < 25)  
                classific = "Ideal";  
            else  
                if(imc < 30)  
                    classific = "Sobrepeso";  
                else  
                    if(imc < 35)  
                        clas = "Obeso";  
                    else  
                        if (imc < 40)  
                            classific = "Obesidade Severa";  
                        else  
                            classific = "Obesidade M3rbida";  
        return classific;  
    }  
}
```

## PROGRAMAÇÃO ORIENTADA A OBJETOS Aplicação – IMC





# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Aplicação – IMC



```
import java.text.DecimalFormat;
```

Calcular

```
private void butCalcActionPerformed(java.awt.event.ActionEvent evt)
{
    Imc i = new Imc();
    i.peso = Float.parseFloat(txtpeso.getText());
    i.altura = Float.parseFloat(txtaltura.getText());
    i.imc = i.CalcularImc();
    DecimalFormat df = new DecimalFormat("#0.0");
    txtimc.setText(df.format(i.imc));
    i.classific = i.MostrarClassificacao();
    txtclassific.setText(i.classific);
}
```

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Aplicação – Investimento

### Investimento

Valor Presente  **txtvp**

Taxa  **txttaxa**

Prazo  **txtprazo**

Valor Futuro  **txtvf**

Tipo Investimento  **txttipo**

**butCalc**

Problema		
Dados	Descrição	Valor
vp	Valor Presente	10000.00
i	Taxa Juros	5% -> 0.05
n	Prazo	12
vf	Valor Futuro	???
Fórmula		
$vf = vp \times (1 + i)^n$		
Exemplo		
$vf = 10000 \times (1 + 0.05)^{12}$ $vf = 17.958.55$		
Melhor tipo de investimento ??? Rever		

Tipos Investimentos	
Descrição	Critério
Poupança	Prazo $\geq$ 24 meses e Retorno $\geq$ 30000
Renda Fixa (CDB e LC)	Prazo $\geq$ 12 meses e Retorno $\leq$ 15000
Renda Variável (Fundos e Mercados)	Prazo $\leq$ 6 meses e Retorno $\leq$ 10000
Rever Investimento	Nenhuma das condições

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Aplicação – Investimento - Classe



### Investimento

- + vp: float
- + taxa: float
- + prazo: int
- + vf: double
- + tipo: String

---

- + CalcularVFuturo(): double
- + MostrarTipo(): String

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Aplicação – Investimento - Classe



## Carregar Formulário

```
public static void main(String[] args) {  
    frmInvestimento f = new frmInvestimento();  
    f.setVisible(true);  
}
```

```
package investimento;  
public class Investimento {  
    float vp, taxa;  
    int prazo;  
    double vf;  
    String tipo;  
  
    public double CalcularVFuturo()  
    {  
        vf = vp * Math.pow(1+taxa, prazo);  
        return vf;  
    }  
  
    public String MostrarTipo()  
    {  
        if(prazo >= 24 && vf >= 30000)  
            tipo = "Poupança";  
        else  
            if(prazo >= 12 && vf <= 15000)  
                tipo = "Renda Fixa";  
            else  
                if(prazo <= 6 && vf <= 10000)  
                    tipo = "Renda Variável";  
                else  
                    tipo = "Rever Investimento";  
        return tipo;  
    }  
}
```

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Aplicação – Investimento



```
import java.text.DecimalFormat;
```

Calcular

```
private void butCalcActionPerformed(java.awt.event.ActionEvent evt) {  
    Investimento i = new Investimento();  
    i.vp = Float.parseFloat(txtvp.getText());  
    i.taxa = Float.parseFloat(txttaxa.getText());  
    i.prazo = Integer.parseInt(txtprazo.getText());  
    i.vf = i.CalcularVFuturo();  
    i.tipo = i.MostrarTipo();  
    DecimalFormat f = new DecimalFormat("#,###,##0.00");  
    txtvf.setText(f.format(i.vf));  
    txttipo.setText(i.tipo);  
}
```

Investimento

Valor Presente 10000.00

Taxa 0.05

Prazo 12

Valor Futuro 17.958,55

Tipo Investimento Rever Investimento

Calcular

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

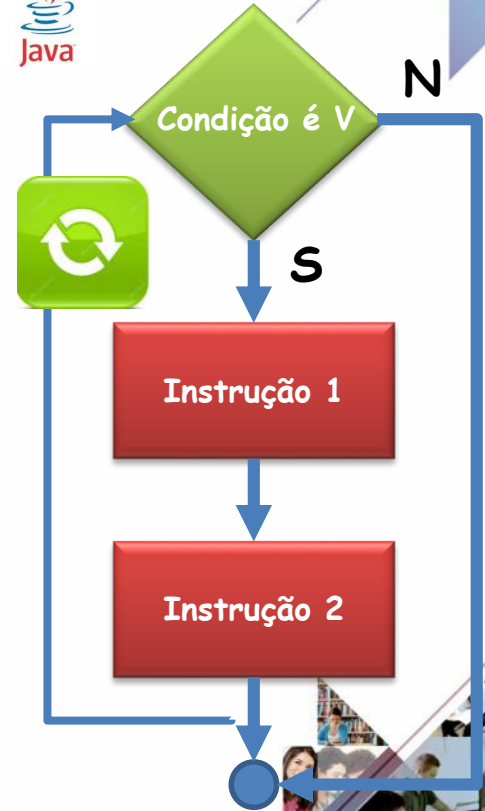
### Estruturas de Repetição

#### for



```
for (<var> = val inicial; <var> <condição> <valor>; incremento)
{
    <comando>;
    <comando>;
}
```

```
for ( j = 1; j <= 10; j = j + 1)
{
    x = x + 1;
    y = y - 1;
}
```





# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Estruturas de Repetição for



```
int x = 0, y = 15, j;  
for (j = 1; j <= 10; j = j + 1)  
{  
  
    x = x + 1;  
  
    y = y - 1;  
  
}
```

x = ?

y = ?

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Estruturas de Repetição for



```
int x = 0, y = 15, z = 0, j;  
for (j = 1; j <= 10; j = j + 1)  
{  
    x = x + 1;  
    y = y - 1;  
    if(x >= 7 && y <= 10)  
        z = z + 1;  
}
```

x = ?

y = ?

z = ?

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

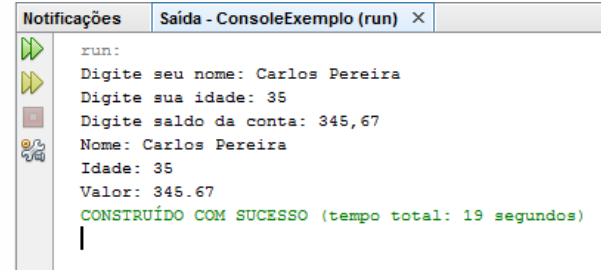
### Estruturas de Repetição

#### Modo Console



```
import java.util.Scanner;
```

```
public class ConsoleExemplo {  
    public static void main(String[] args) {  
        Scanner s = new Scanner(System.in);  
        System.out.print("Digite seu nome: ");  
        String nome = s.nextLine();  
        System.out.print("Digite sua idade: ");  
        int idade = s.nextInt();  
        System.out.print("Digite saldo da conta: ");  
        float saldo = s.nextFloat();  
        System.out.println("Nome: " + nome);  
        System.out.println("Idade: " + idade);  
        System.out.println("Valor: " + saldo);  
    }  
}
```



```
run:  
Digite seu nome: Carlos Pereira  
Digite sua idade: 35  
Digite saldo da conta: 345,67  
Nome: Carlos Pereira  
Idade: 35  
Valor: 345.67  
CONSTRUÍDO COM SUCESSO (tempo total: 19 segundos)  
|
```

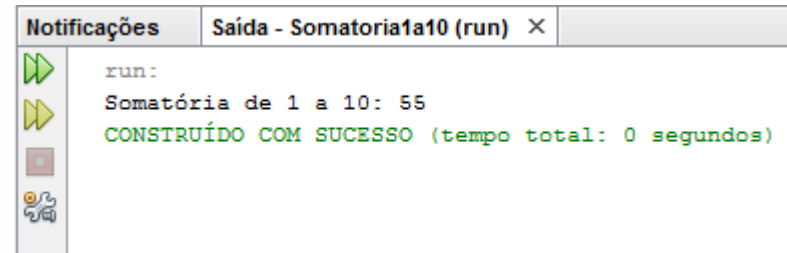
# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Estruturas de Repetição for (Modo Console)



Somatória de números entre 1 e 10.

```
public class Somatoria1a10 {  
    public static void main(String[] args) {  
        int j, ac=0;  
        for(j=1; j<=10; j=j+1)  
        {  
            ac = ac + j;  
        }  
        System.out.println("Somatória de 1 a 10: " + ac);  
    }  
}
```



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Estruturas de Repetição for (modo console)



1. Somatória de números entre 1 e 10, somente dos ímpares.
2. Somatória números 5, 10, 15, 20 , ..., 100 múltiplos de 4.

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Estruturas de Repetição





#### for (modo console)



Fatorial de um determinado número:

Exemplo: **5!**  $5 \times 4 \times 3 \times 2 \times 1 = 120$

incremento     -1   -1   -1   -1

Notificações	Saída - FatorialConsole (run) ×
	run:
	Informe número: 5
	Fatorial: 120
	CONSTRUÍDO COM SUCESSO (tempo total: 8 segundos)



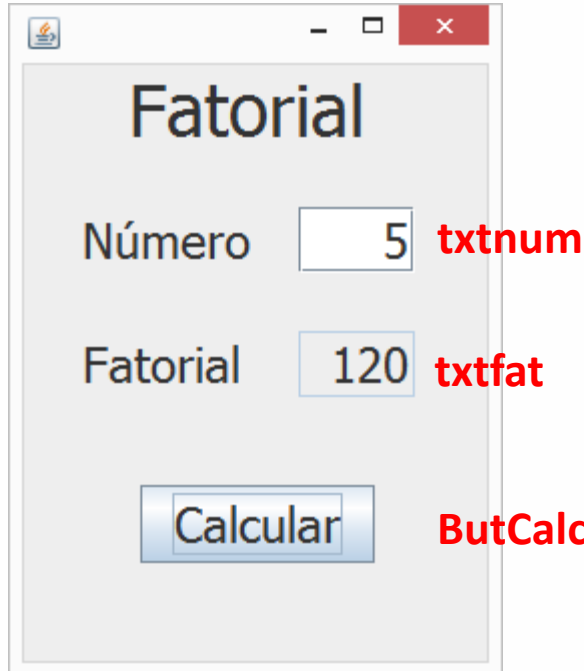
# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

Estruturas de Repetição  
for (modo gráfico - POO)



### Fatorial



The screenshot shows a Java Swing window titled "Fatorial". Inside the window, there is a label "Número" followed by a text input field containing the value "5". To the right of the input field is the label **txtnum**. Below this, there is a label "Fatorial" followed by a text input field containing the value "120". To the right of this input field is the label **txtfat**. At the bottom of the window, there is a button labeled "Calcular" with the label **ButCalc** to its right.

# Modelagem e Desenvolvimento Orientado a Objeto

PROGRAMAÇÃO ORIENTADA A OBJETOS

Estruturas de Repetição  
for (modo gráfico - POO)



## Fatorial

### Fatorial

+ num: int

+ fat: long

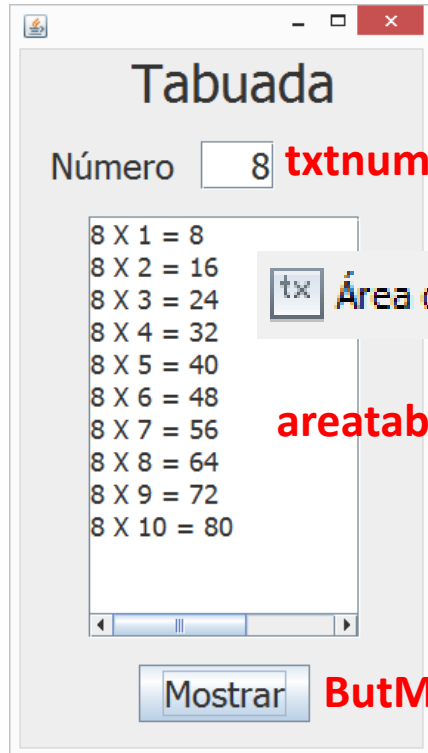
+ CalcularFatorial(): long

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Estruturas de Repetição for - Tabuada



Observação: programação não OO.



Área de Texto

Propriedade ***editable*** (desmarcar)  
Método: ***append()***

Concatenação:

*número + " X " + contador + " = " + número \* contador + "\n";*

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS Estruturas de Repetição for – Faixa Etária



Gerar 20 idades aleatórias entre 0 e 100 anos:

```
import java.util.Random;
```

```
//Inicia Aleatório  
Random rnd = new Random();  
//Gera um número aleatório (0 - 100)  
idade = rnd.nextInt(101);
```

**Observação:**  
**programação**  
**não OO.**

Idade	Faixa Etária
< 15	Criança
>= 15 e < 25	Jovem
>= 25 e < 60	Adulto
>= 60	Idoso

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

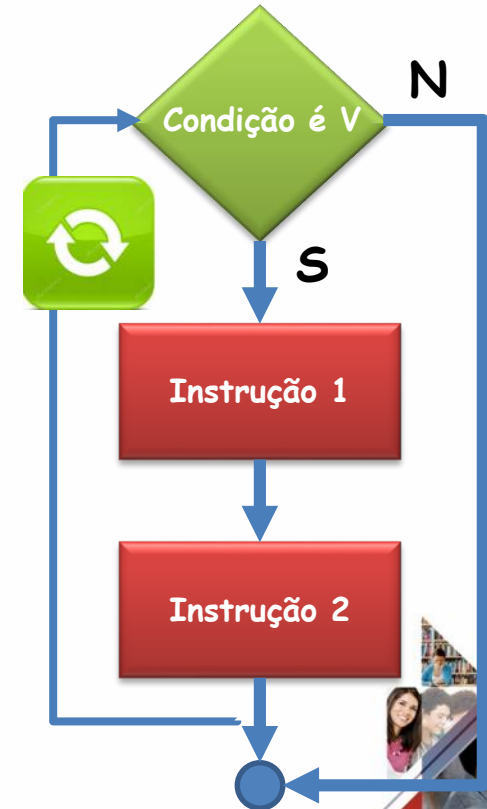
### Estruturas de Repetição

#### while



```
while(<condição V>)  
{  
    <comando>;  
    <comando>;  
}
```

```
while(x < y)  
{  
    x = x + 1;  
    y = y - 1;  
}
```



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Estruturas de Repetição

while



```
int x = 0, y = 15;
while(x < y)
{
    x = x + 1;
    y = y - 1;
    if (x == y)
        z = z + 1;
}
```

x = ?

y = ?

z = ?



# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

Estruturas de Repetição

while – Raiz Quadrada



**exata**

cc	n	ci	r
1	16	1	15
2	15	3	12
3	12	5	7
4	7	7	0

**Não exata**

cc	n	ci	r
1	19	1	18
2	18	3	15
3	15	5	10
4	10	7	3
5	3	9	-6

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

Estruturas de Repetição

while – Raiz Quadrada



Raiz Quadrada

Número  **txtnum**

Raiz Quadrada  **txtrq**

**ButCalc**

Raiz Quadrada

Número

Raiz Quadrada

Concatenação (+):

"Entre " + (cc-1) + " e " + cc

# Modelagem e Desenvolvimento Orientado a Objeto

PROGRAMAÇÃO ORIENTADA A OBJETOS

Estruturas de Repetição

while – Raiz Quadrada



## RaizQuadrada

+ num: int

+ raizq: String

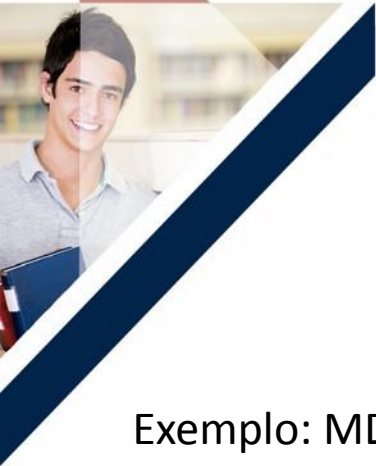
+ CalcularRQ(): String

# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Estruturas de Repetição

while – Máximo Divisor Comum (MDC)



Exemplo: MDC(27, 18)

**1**, 2, 3, 4, 5, 6, 7, 8, **9** ....

Dividendo (d1)	Divisor (d2)	Resto (r)
152	32	24
32	24	8
24	8	0

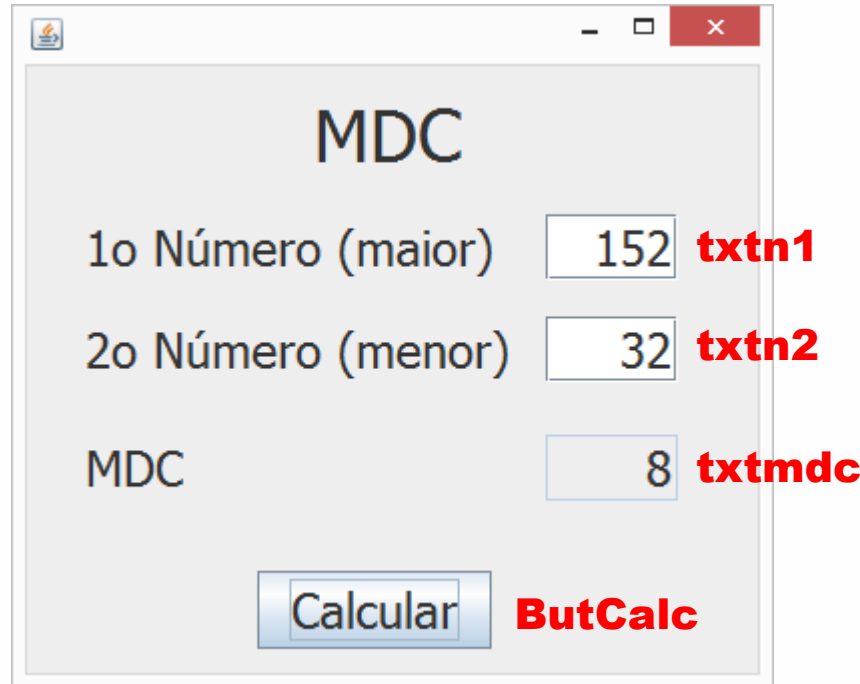


# Modelagem e Desenvolvimento Orientado a Objeto

## PROGRAMAÇÃO ORIENTADA A OBJETOS

### Estruturas de Repetição

while – Máximo Divisor Comum (MDC)



The screenshot shows a Java Swing window titled "MDC". Inside the window, there are three text input fields and a button. The first input field is labeled "1o Número (maior)" and contains the value "152". The second input field is labeled "2o Número (menor)" and contains the value "32". The third input field is labeled "MDC" and contains the value "8". Below these fields is a button labeled "Calcular". To the right of each input field, there are red labels: "txtn1" for the first field, "txtn2" for the second field, and "txtmdc" for the third field. To the right of the "Calcular" button, there is a red label "ButCalc".

# Modelagem e Desenvolvimento Orientado a Objeto

PROGRAMAÇÃO ORIENTADA A OBJETOS

Estruturas de Repetição

while – Máximo Divisor Comum (MDC)



**MDC**

+ d1: int  
+ d2: int  
+ mdc: int

+ CalcularMDC(): int

# Modelagem e Desenvolvimento Orientado a Objeto

