



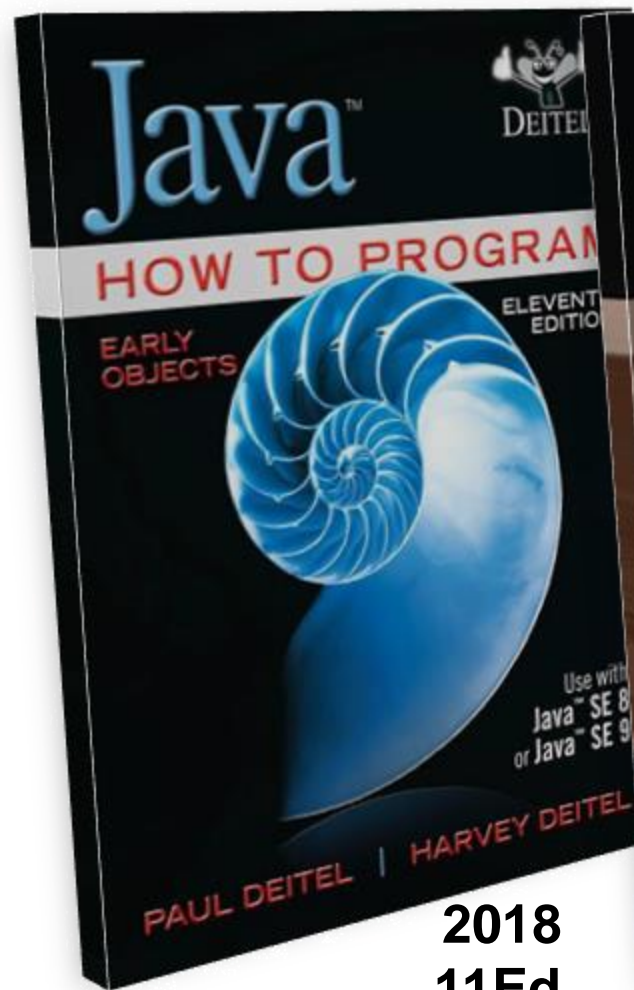
Conceitos Básicos

Linguagem JAVA

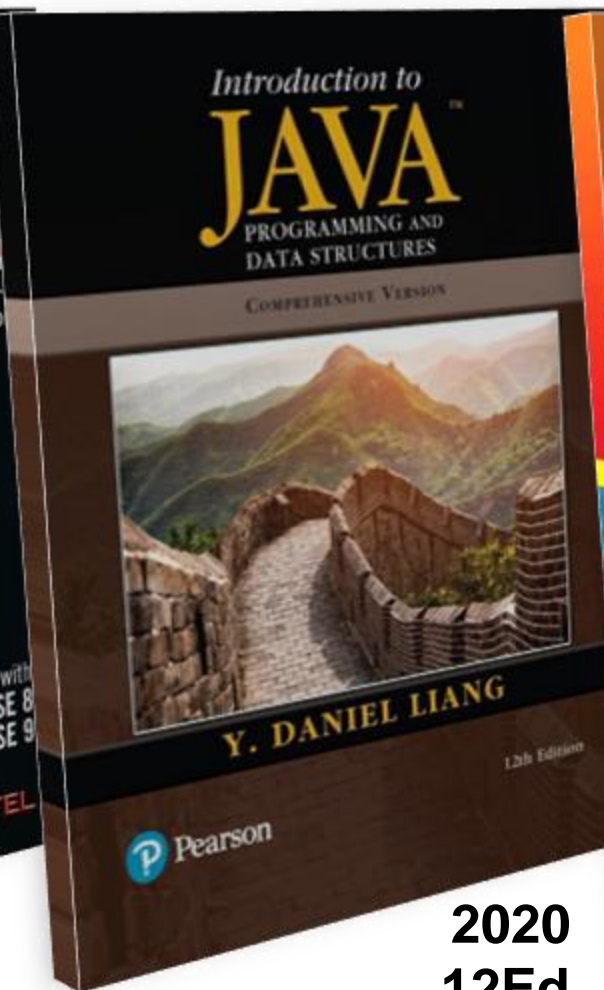


Prof. Ausberto S. Castro V.
ascv@uenf.br

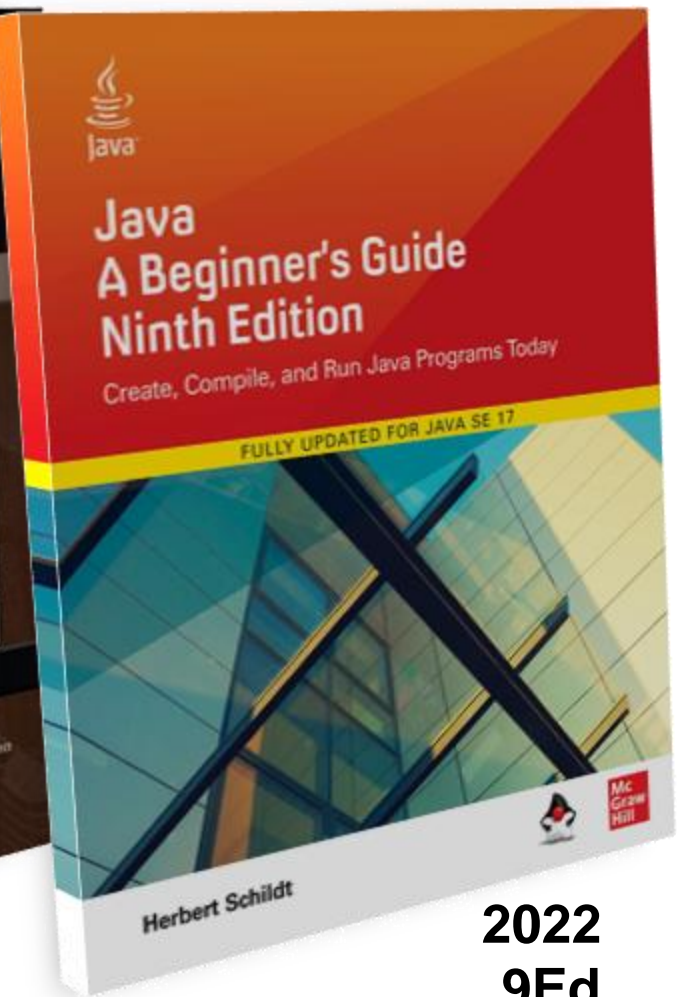
Bibliografia Básica



2018
11Ed



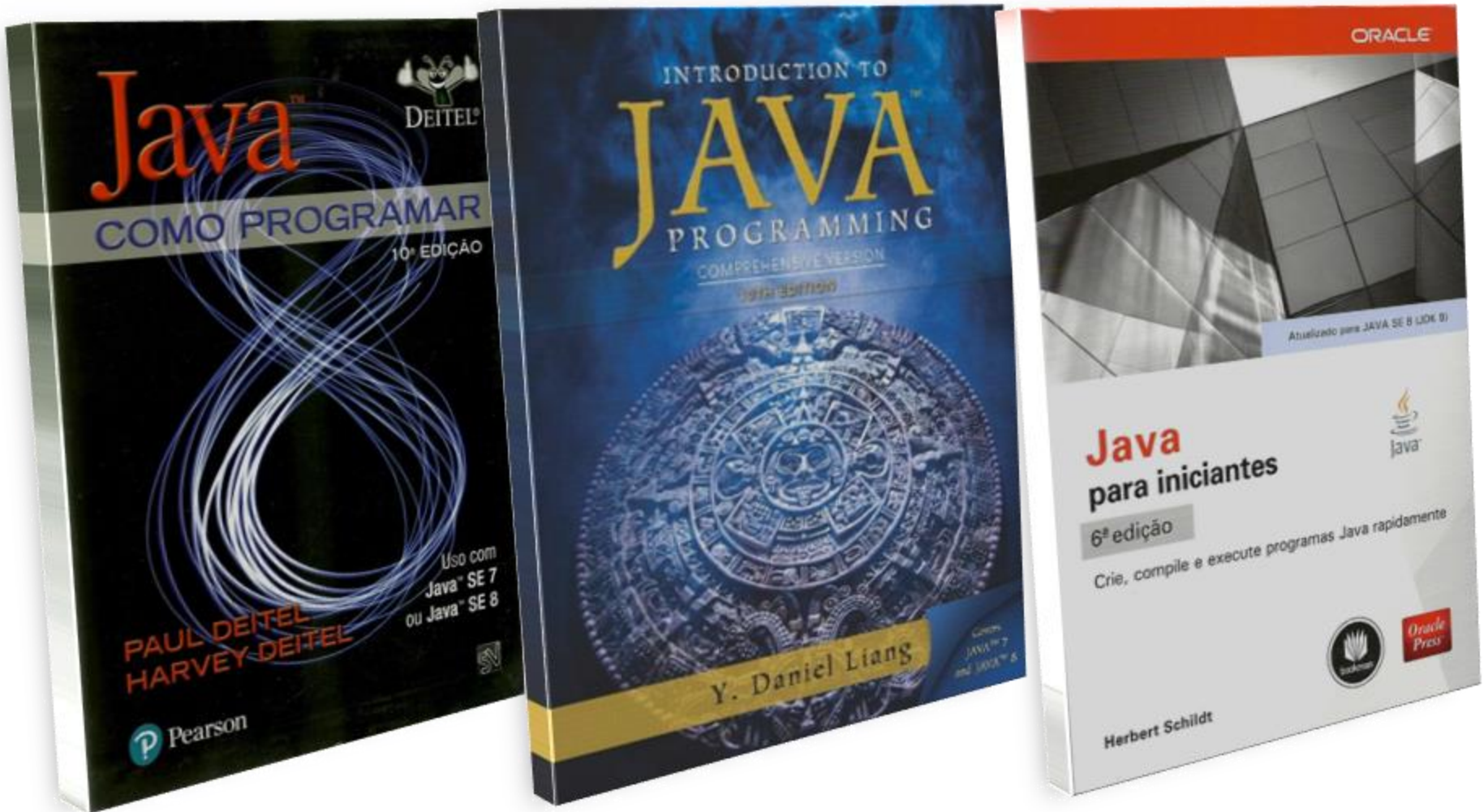
2020
12Ed



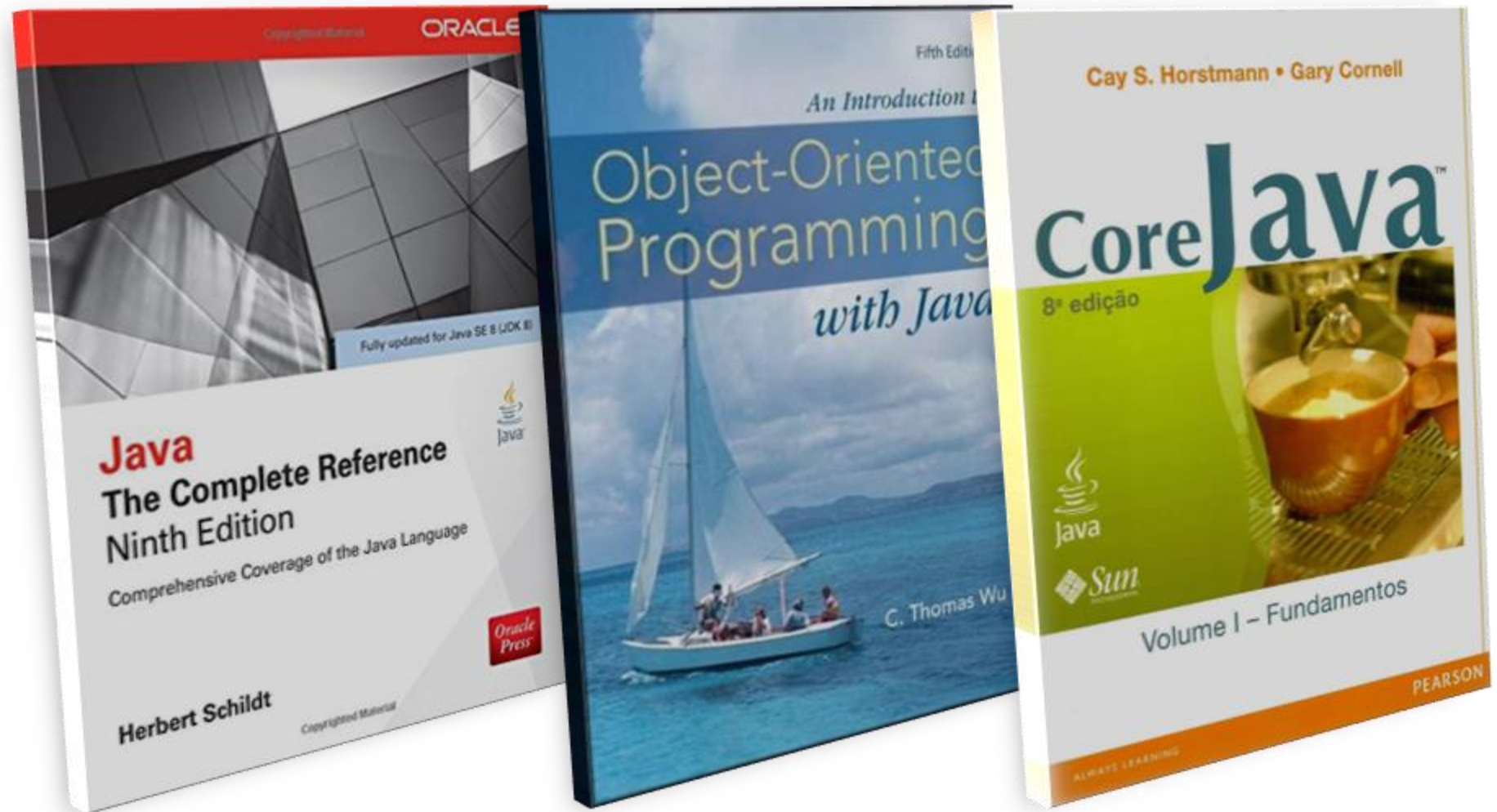
2022
9Ed

Versões anteriores em Português

Bibliografia Básica



Bibliografia complementar



Características da Linguagem JAVA

1. **Java é Simples**
2. **Java é Orientado a Objetos**
3. **Java é Distribuída**
4. **Java é Interpretada**
5. **Java é Robusta**
6. **Java é Segura**
7. **Java é de Arquitetura-Neutral**
8. **Java é Portável**
9. **Java é de boa Performance**
10. **Java é Multi-thread**
11. **Java é Dinâmica**



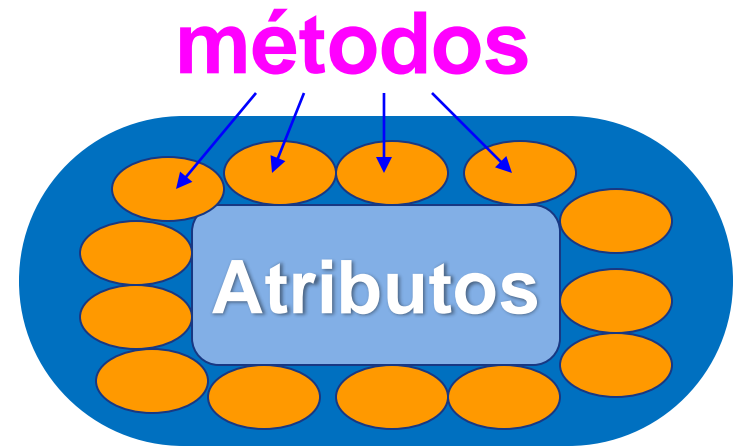
Paradigma Orientado a Objetos

❖ Paradigma OO: Baseado nos conceitos de *objeto* e *classe de objetos*

- Um *objeto* é uma variável (estrutura) junto com um conjunto de operações.

❖ Elementos

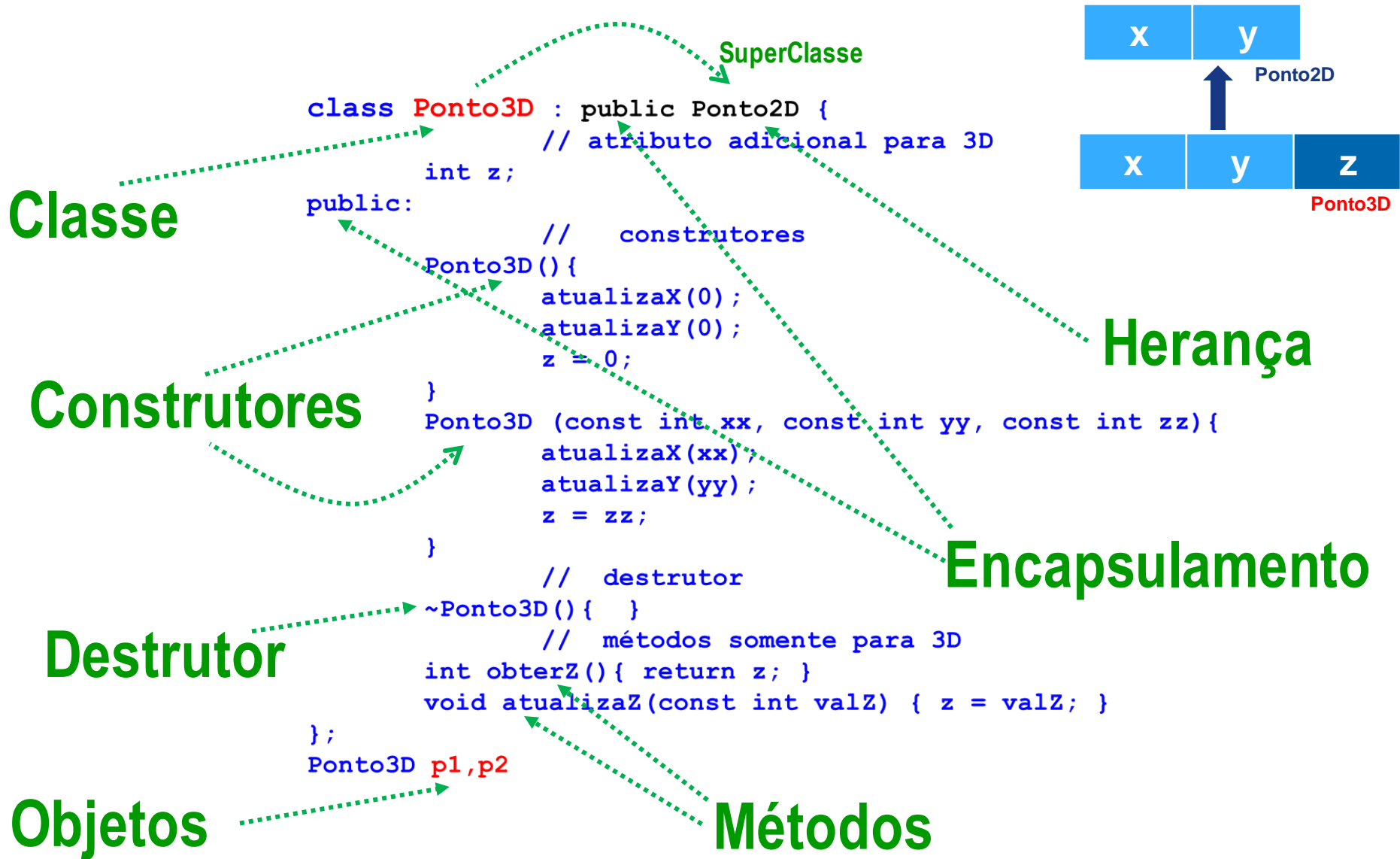
- Objetos e classes
- Métodos (mensagens, operações)
- Herança
- Polimorfismo
- Encapsulamento
- Tipos de Dados Abstratos (TDA)



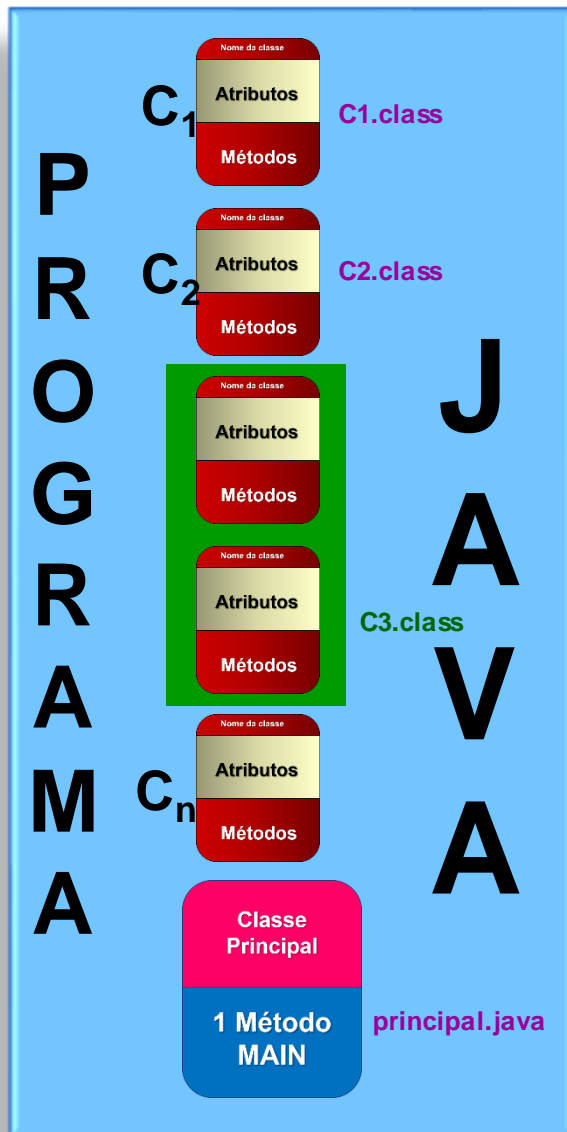
Paradigma Orientado a Objetos

```
class Ponto3D : public Ponto2D {
    // atributo adicional para 3D
    int z;
public:
    // construtores
    Ponto3D(){
        atualizaX(0);
        atualizaY(0);
        z = 0;
    }
    Ponto3D (const int xx, const int yy, const int zz){
        atualizaX(xx);
        atualizaY(yy);
        z = zz;
    }
    // destrutor
    ~Ponto3D(){ }
    // métodos somente para 3D
    int obterZ(){ return z; }
    void atualizaZ(const int valZ) { z = valZ; }
};
Ponto3D p1,p2
```

Paradigma Orientado a Objetos



Um programa em JAVA



Um programa JAVA tem:

- Uma ou várias classes (nome, atributos, métodos)
- Uma classe pode estar em único arquivo ***.class**
- Várias classes podem estar juntos em um único arquivo
- Uma classe principal com um único método `main(argumentos)`
- O arquivo da classe principal tem o nome da classe e a extensão ***.java**

```
public static void main(String args[]){  
    ...  
}
```
- A classe principal permite que uma aplicação JAVA seja executado utilizando o método `main()`
- A palavra **void** indica que o método executará uma tarefa, mas não retornará nenhuma informação ao completar a sua tarefa

VeiculoMetodo.java - Visual Studio Code

```
1  * @author Prof. Ausberto S. Castro Vera
2  */
3
4  class Veiculo {
5      int passageiros; // numero de passageiros
6      int CapCombust; // capacidade de combustivel (litros)
7      int kmpl;        // consumo de combustivel Km por litro
8
9      // metodo que exibe autonomia
10     void autonomia(){
11         System.out.println("Autonomia de " + CapCombust * kmpl + " km\n");
12     }
13 }
14
15 // Esta classe declara um objeto de tipo Veiculo.
16 class VeiculoMetodo {
17     Run | Debug
18     public static void main(String args[]) {
19         Veiculo minivan = new Veiculo();
20         Veiculo sportscar = new Veiculo();
21
22         int autonomia1, autonomia2;
23
24         // atribui valores para campos de minivan
25         minivan.passageiros = 7;
26         minivan.CapCombust = 48;
27         minivan.kmpl = 12;
28
29         // atribui valores para campos de sportscar
30         sportscar.passageiros = 2;
31         sportscar.CapCombust = 45;
32         sportscar.kmpl = 16;
33
34         // Calcula a a autonomia com tanque cheio
35         autonomia1 = minivan.CapCombust * minivan.kmpl;
36         autonomia2 = sportscar.CapCombust * sportscar.kmpl;
37
38         System.out.println("Minivan pode transportar " + minivan.passageiros + " passageiros. ");
39         minivan.autonomia();
40
41         System.out.println("Carro esportivo pode transportar " + sportscar.passageiros + " passageiros. ");
42         sportscar.autonomia();
43     }
44 }
```

classe

main(...) { ... }

Classe principal com o método main

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF Java

Primeiro Programa Java – Apache Netbeans IDE 12.5

The screenshot displays the Apache NetBeans IDE 12.5 interface. The main editor window shows a Java file named `A.java` with the following code:

```
1  /*  
2   * To change this license header, choose License Headers in Project Properties  
3   * To change this template file, choose Tools | Templates  
4   * and open the template in the editor.  
5   */  
6  
7  /**  
8   *  
9   * @author Prof. Ausberto S. Castro Vera (ascv@uenf.br)  
10  */  
11 public class A {  
12     public static void main(String args[]){  
13         System.out.println("\nOi, Brasil!\n");  
14         System.out.println("Bemvindo ao primeiro Programa JAVA! UENF 2021\n");  
15     }  
16 }  
17
```

The left sidebar shows the Project Explorer with the following structure:

- 01FirstJava
 - DoisVeiculos
 - HerancaSimples
 - MetodoConstrutor
 - MetodoParamet
 - Veiculo
 - VeiculoMetodo

The bottom Output window shows the execution results:

```
Run (01FirstJava) x DoisVeiculos (run) x  
Building 01FirstJava 1.0  
-----[ jar ]-----  
--- exec-maven-plugin:3.0.0:exec (default-cli) @ 01FirstJava ---  
  
Oi, Brasil!  
  
Bemvindo ao primeiro Programa JAVA! UENF 2021
```

An "About" dialog box is open on the right, displaying the Apache NetBeans logo and the following text:

NetBeans IDE and NetBeans Platform are based on Apache NetBeans from the Apache Software Foundation and are licensed under [Apache License Version 2.0](https://www.apache.org/licenses/LICENSE-2.0). For more information, please visit netbeans.apache.org.

Product Version: Apache NetBeans IDE 12.5
Java: 17.0.1; Java HotSpot(TM) 64-Bit Server VM 17.0.1+12-LTS-39
Runtime: Java(TM) SE Runtime Environment 17.0.1+12-LTS-39
System: Windows 10 version 10.0 running on amd64; Cp1252; pt_BR (nb)
User directory: C:\Users\ASCVI78G\AppData\Roaming\NetBeans\12.5
Cache directory: C:\Users\ASCVI78G\AppData\Local\NetBeans\Cache\12.5

The status bar at the bottom indicates the current file is `INS | Windows (CRLF)` and the time is 17:1.

Ambiente IDE: Eclipse 2021-09 (r.4.21.0)

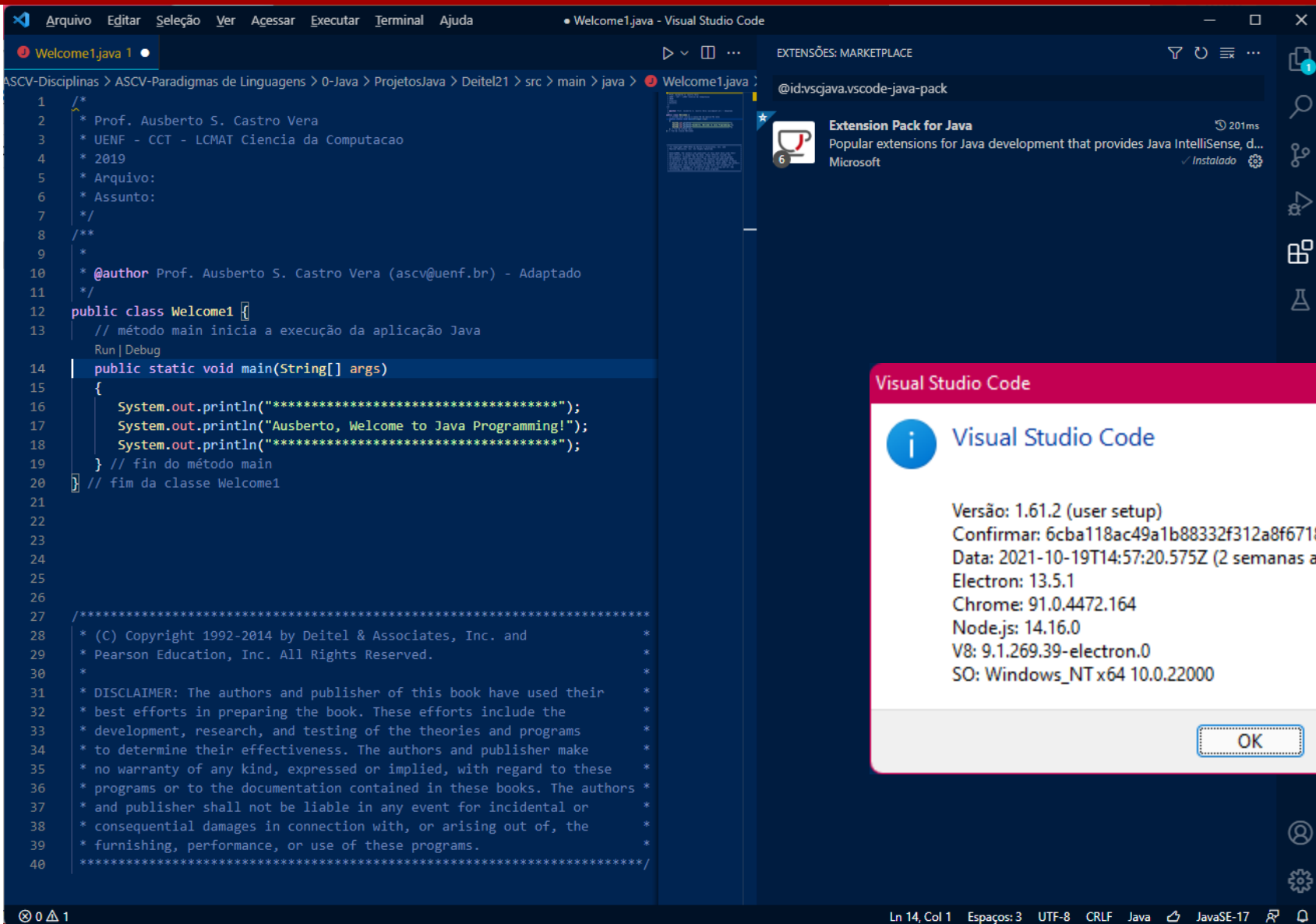
The screenshot displays the Eclipse IDE environment. The main editor window shows the source code for `ClasseAbstrata.java`, which defines an abstract class `Forma2D` and its subclasses. The Project Explorer on the left shows the project structure, including the `Forma2D` class and its methods. The Outline view on the right lists the methods of the `Forma2D` class.

The **About Eclipse IDE** dialog box is open, providing the following information:

- Eclipse IDE for Java Developers (includes Incubating components)**
- Version:** 2021-09 (4.21.0)
- Build id:** 20210910-1417
- Copyright:** (c) Copyright Eclipse contributors and others 2000, 2021. All rights reserved.
- Trademarks:** Eclipse and the Eclipse logo are trademarks of the Eclipse Foundation, Inc., <https://www.eclipse.org/>. The Eclipse logo cannot be altered without Eclipse's permission. Eclipse logos are provided for use under the Eclipse logo and trademark guidelines, <https://www.eclipse.org/logotm/>. Oracle and Java are trademarks or registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.
- Open Source Projects:** This product includes software developed by other open source projects including the Apache Software Foundation, <https://www.apache.org/>.

The dialog box also features a row of icons representing various open-source projects and a button for **Installation Details**.

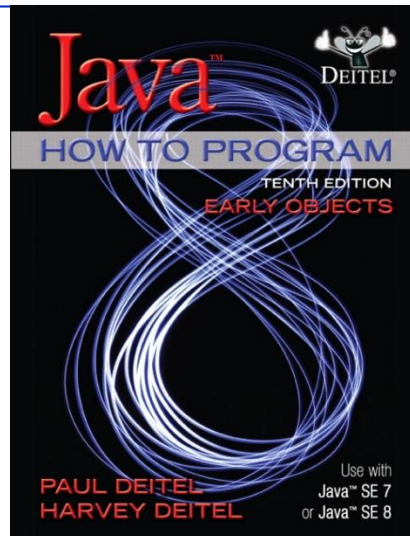
Ambiente IDE: Visual Studio Code 1.61.2



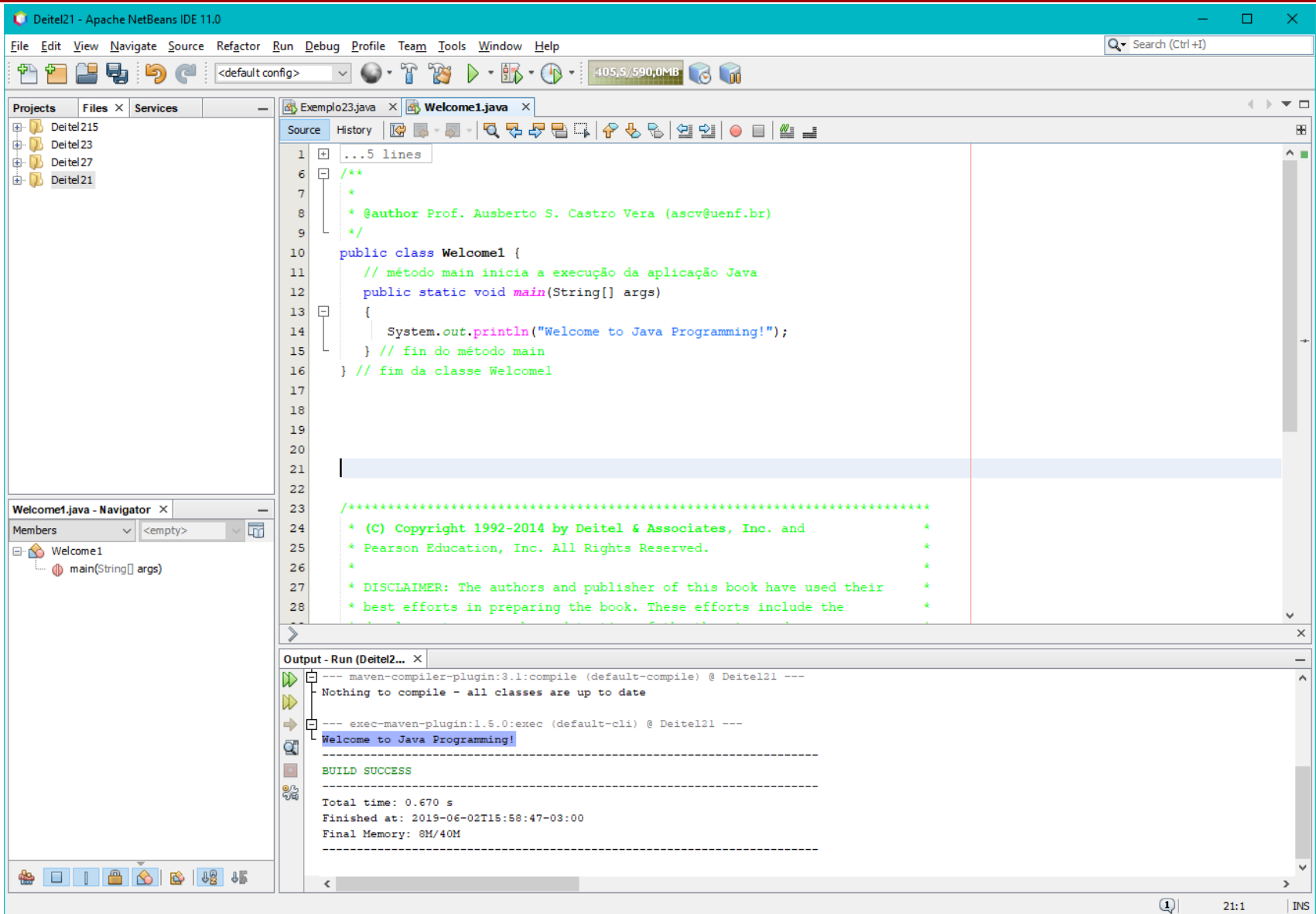
Primeiro Programa Java

```
1 // Fig. 2.1: Welcome1.java
2 // Text-printing program.
3
4 public class Welcome1
5 {
6     // main method begins execution of Java application
7     public static void main(String[] args)
8     {
9         System.out.println("Welcome to Java Programming!");
10    } // end method main
11 } // end class Welcome1
```

Welcome to Java Programming!



Primeiro Programa Java



Comentários

❖ Comentário fim-de-linha

- Começa com `//` e termina no fim da linha onde aparece
- Deve usar-se uma símbolo `//` para cada linha

```
29      sum = number1 + number2; // adiciona numeros, e armazena em sum
30
31      System.out.printf("A soma dos dois numeros = %d\n", sum); // mostra a soma
32  } // fim do método main
33  } // fim da classe Exemplo3
```

❖ Comentário Tradicional

- Pode se estender a várias linhas
`/* Este é um comentario tradicional. Pode se
estender por múltiplas linhas */`
- Começa com o símbolo `/*` e termina com `*/`.
- Todos os textos entre estes símbolos delimitadores é ignorado pelo compilador

Tipos - Variáveis

- ❖ **Valor:** algo que pode ser avaliado, armazenado, passado como parâmetro
- ❖ **Tipo:** conjunto de valores
- ❖ **Variável:** atalho para um valor $(x \leftarrow 27)$

- ❖ **Java é uma linguagem fortemente tipada**
 - Não há o conceito de uma variável “sem tipo”
 - Operações inválidas não são compiladas
 - Ajuda a impedir a ocorrência de erros
 - Melhora a confiabilidade dos programas



Identificadores

❖ Identificador

- Nome de coisas que aparecem em um programa

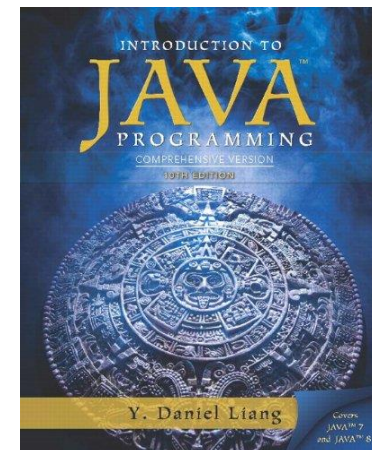
❖ Regras

- Sequencia de caracteres que podem ser :letras, digitos, _ e \$
- Começa com uma letra, underscore _ ou \$. Nunca com um digito
- Não pode ser uma palavra reservada da linguagem Java
- Não pode ser `true`, `false` ou `null`
- Pode ter qualquer comprimento

❖ Exemplos:

- \$2, `ComputeArea`, `area`, `radius`, e `print`

```
Código-Fonte  Histórico  [Icons]
1  package chapter2;
2
3  public class ComputeArea {
4      public static void main(String[] args) {
5          double radius; // Declare radius
6          double area; // Declare area
7
8          // Assign a radius
9          radius = 20; // New value is radius
10
11         // Compute area
12         area = radius * radius * 3.14159;
13
14         // Display results
15         System.out.println("The area for the circle of radius " +
16             radius + " is " + area);
17     }
18 }
```



Tipos - Variáveis

❖ **Java contém DOIS tipos de dados**

- **Tipos orientados a Objetos**

CLASSES
OBJETOS

- **Tipos Não-orientados a objetos**

booleanos
byte
char
double

float
int
long
short

Variáveis

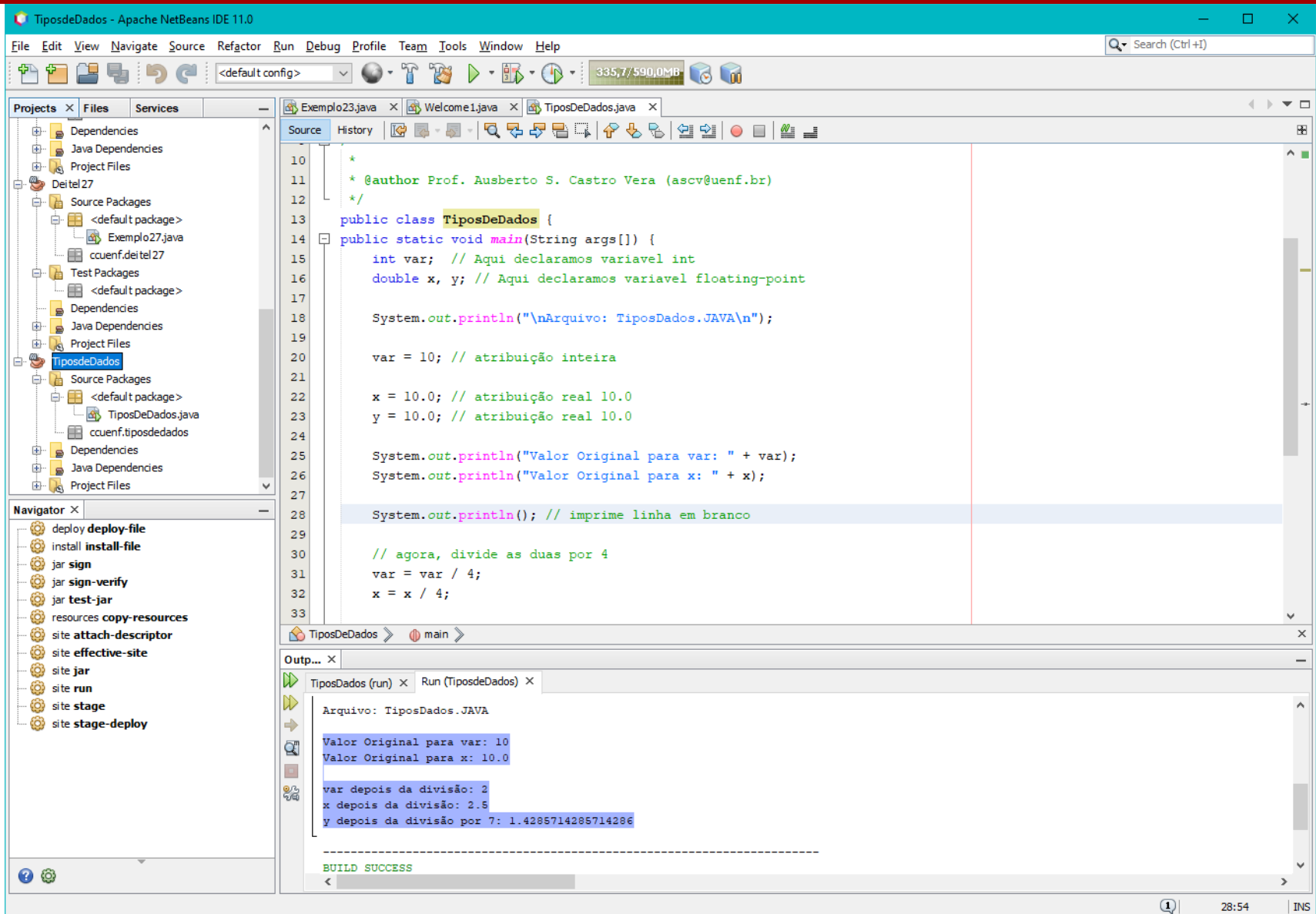
The screenshot displays the NetBeans IDE 8.2 interface. The main editor window shows the source code of a Java class named `Example2`. The code includes a package declaration, a comment, a class declaration, and a `main` method. Inside the `main` method, two integer variables, `var1` and `var2`, are declared and initialized. `var1` is assigned the value 1024, and `var2` is assigned the value of `var1` divided by 2. The program then prints the values of both variables to the console.

```
4  * 2016
5  */
6
7  /**
8   *
9   * @author Prof. Ausberto S. Castro Vera
10  */
11  class Example2 {
12      public static void main(String args[]) {
13          int var1; // declara uma variável
14          int var2; // declara outra variável
15
16          var1 = 1024; // atribui 1024 a var1
17
18          System.out.println("var1 contem " + var1);
19
20          var2 = var1 / 2;
21
22          System.out.print("var2 contem var1 / 2: ");
23          System.out.println(var2);
24      }
25  }
```

The bottom panel shows the output of the program's execution:

```
run:
var1 contem 1024
var2 contem var1 / 2: 512
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Tipos de Variáveis



Variáveis

```
// Calcular a área
```

```
radio = 1.0;
```

```
area = radio * radio * 3.14159;
```

```
System.out.println("A área é " + area +  
    " para um radio "+radius);
```

```
// Calcular uma segunda área
```

```
radio = 2.0;
```

```
area = radio * radio * 3.14159;
```

```
System.out.println("A área é " + area +  
    " para um radio "+radius);
```

Declarando variáveis

```
tipoVar NomeVar;
```

```
int x;           // Declara x como uma
                  // variável inteira;

double radio;    // Declara radio como
                  // uma variável double;

char a;          // Declara a ser uma
                  // variável caracter;
```


Operadores Numéricos

Nome	Significado	Exemplo	Resultado
+	Adição	$34 + 1$	35
-	Subtração	$34.0 - 0.1$	33.9
*	Multiplicação	$300 * 30$	9000
/	Divisão	$1.0 / 2.0$	0.5
%	Resto	$20 \% 3$	2

Operadores Relacionais

Operador Java	Símbolo Matemático	Nome	Exemplo (radio é 5)	Resultado
<	<	menor que	<code>radius < 0</code>	<code>false</code>
<=	=	menor ou igual que	<code>radius <= 0</code>	<code>false</code>
>	>	maior que	<code>radius > 0</code>	<code>true</code>
>=	≥	maior ou igual que	<code>radius >= 0</code>	<code>true</code>
==	=	igual a	<code>radius == 0</code>	<code>false</code>
!=	≠	não igual a	<code>radius != 0</code>	<code>true</code>

I/O: Fluxo de Bytes e de caracteres

- ❖ **Java enxerga um arquivo como um fluxo sequencial de caracteres ou bytes**

10111011 11010001 01010101 01001011 10111010

- ❖ **Fluxo de bytes**

- Fornecem um meio para tratamento de saída e entrada de bytes
- Usados na leitura e gravação de dados binários (arquivos)

- ❖ **Fluxo de caracteres**

- Projetados para o tratamento de entrada e saída de caracteres
- Utilizam Unicode (internacionalização)

Fluxos predefinidos

classe System

System.out

- É objeto do tipo **PrintStream**
- É o fluxo de saída básico
- Por padrão, utiliza o console (tela)

System.in

- É objeto do tipo **InputStream**
- É a entrada básica
- Por padrão utiliza o teclado

System.err

- É objeto do tipo **PrintStream**
- É o fluxo de erro básico
- Utiliza o console (tela)
- Podem ser redirecionados para outro dispositivo I/O

InputStream

- `void close()`
- `int read()`
- `void reset()`

OutputStream

- `void close()`
- `void flush()`
- `void write(int b)`
- `void write(byte buffer[])`

```
System.in.read(dados)  
System.out.print(dados)
```

Exemplo : Conta Bancaria

The screenshot shows the Apache NetBeans IDE 11.0 interface. The main editor displays the `ContaBancaria.java` file, which includes a package declaration, a class definition, and a `main` method. The `main` method creates two `Conta` objects, `contal` and `conta2`, and prints their initial balances. The `Conta` class is defined with `Nome` and `Saldo` attributes, and methods for `getNome`, `getSaldo`, `setNome`, and `deposito`.

```
10 *
11 * @author Prof. Ausberto S. Castro Vera (ascv@uenf.br)
12 */
13
14 // Fig. 3.9: ContaBancaria.java
15 // Entradas e saidas de valores com objetos Conta.
16 import java.util.Scanner;
17
18 public class ContaBancaria {
19
20     public static void main(String[] args)
21     {
22         Conta contal = new Conta("Silvia Lopes", 50.00);
23         Conta conta2 = new Conta("Paulo Souza", -7.53);
24
25         // display initial saldo of each object
26         System.out.printf("%s saldo: $%.2f\n",
27             contal.getNome(), contal.getSaldo());
28         System.out.printf("%s saldo: $%.2f\n",
29             conta2.getNome(), conta2.getSaldo());
30     }
31 }
```

The output window shows the execution of the program, displaying the initial balances of the two accounts and the results of two deposit transactions.

```
TiposDados (run) X Run (ContaBancaria) X
--- exec-maven-plugin:1.5.0:exec (default-cli) @ ContaBancaria ---
Silvia Lopes saldo: $50,00
Paulo Souza saldo: $0,00

Ingresse a quantidade de deposito para a conta 1: 20

Adicionando 20,00 para o saldo da conta 1

Silvia Lopes saldo: $70,00
Paulo Souza saldo: $0,00

Ingresse a quantidade de deposito para a conta 2: 26

Adicionando 26,00 ao saldo da conta 2

Silvia Lopes saldo: $70,00
Paulo Souza saldo: $26,00
```

CONTA

double Nome;
double Saldo;

Conta(string Nome, double saldo)
Deposito(double q)
Double getSaldo()
setNome(string n)
String getNome()

IF

*if (condição)
instrução*

```
Ch = (char) System.in.read();  
if ( ch == resposta )  
    System.out.println( "Exatamente..." );
```

```
if ( NotaAluno >= 6.0 )  
    System.out.println( "Passou" );
```

Instrução IF-ELSE

```
if (condição)
    instrução
else
    instrução
```

```
if ( NotaAluno >= 6.0 )
    System.out.println( "Aluno Aprovado" );
else
    System.out.println( "Aluno Reprovado" );
```

if-else-if

```
if (condição)
    instrução
else if (condição)
    instrução
else if (condição)
    instrução
. . . . .
else
    instrução
```

```
if (NotaAluno >= 9.0)
    System.out.println("Excelente");
else
    if (NotaAluno >= 8.0)
        System.out.println("Muito Bom");
    else
        if (NotaAluno >= 7.0)
            System.out.println("Bom");
        else
            if (NotaAluno >= 6.0)
                System.out.println("Regular");
            else
                System.out.println("Deficiente");
```

Instrução SWITCH

```
switch (expressão){  
    case constante1:  
        sequencia de instruções  
        break;  
    case constante2:  
        sequencia de instruções  
        break;  
    case constante3:  
        sequencia de instruções  
        break;  
    .....  
    default:  
        sequencia de instruções  
}
```

```
System.in.read(grado) ;  
switch (grado)  
{  
    case 10:  
    case 9:  
        ++ContaA;  
        break;  
    case 8:  
        ++ContaB;  
        break;  
    case 7:  
        ++ContaC;  
        break;  
    default:  
        mensagem() ;  
};
```

Instrução SWITCH

```
switch (expressão){  
    case constante1:  
        sequencia de instruções  
        break;  
    case constante2:  
        sequencia de instruções  
        break;  
    case constante3:  
        sequencia de instruções  
        break;  
    .....  
    default:  
        sequencia de instruções  
}
```

```
public class SwitchDemo {  
    public static void main(String[] args) {  
        int month = 8;  
        String monthString;  
        switch (month) {  
            case 1: monthString = "Janeiro";  
                    break;  
            case 2: monthString = "Fevereiro";  
                    break;  
            case 3: monthString = "Março";  
                    break;  
            case 4: monthString = "Abril";  
                    break;  
            case 5: monthString = "Maio";  
                    break;  
            case 6: monthString = "Junho";  
                    break;  
            case 7: monthString = "Julho";  
                    break;  
            case 8: monthString = "Agosto";  
                    break;  
            case 9: monthString = "Setembro";  
                    break;  
            case 10: monthString = "Outubro";  
                    break;  
            case 11: monthString = "Novembro";  
                    break;  
            case 12: monthString = "Dezembro";  
                    break;  
            default: monthString = "Mês invalido";  
                    break;  
        }  
        System.out.println(monthString);  
    }  
}
```

Laço FOR

FORMA GERAL

```
for (inicialização; condição; iteração) instrução
```

BLOCO

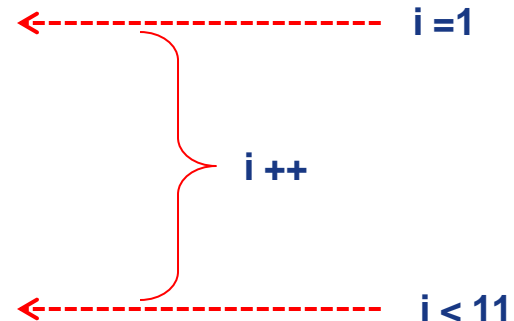
```
for (inicialização; condição; iteração)  
{  
    sequência de instruções  
}
```

Laço FOR

```
class ForDemo {  
    public static void main(String[] args){  
        for(int i=1; i<11; i++){  
            System.out.println("A conta é: " + i);  
        }  
    }  
}
```

SAÍDA:

```
A conta é : 1  
A conta é : 2  
A conta é : 3  
A conta é : 4  
A conta é : 5  
A conta é : 6  
A conta é : 7  
A conta é : 8  
A conta é : 9  
A conta é : 10
```



Laço WHILE

while (*condição*)
instrução

```
system.in.read(ch);  
while (ch <= 'z'){  
    system.out.print(ch);  
    ch++;  
};
```

```
resultado =1  
e =k;  
  
while (e > 0){  
    resultado *=2;  
    e--;  
};
```

Laço DO-WHILE

```
do {  
    instruções  
} while (condição);
```

```
system.in.read(ch);  
do {  
    system.out.print('Pressione uma tecla seguida de <ENTER>');  
    ch (char) System.in.read();    // obtém um caracter  
} while (ch != 'q');
```



Prof. Dr. Ausberto S. Castro Vera
Ciência da Computação
UENF-CCT-LCMAT
Campos, RJ

ausberto.castro@gmail.com
ascv@uenf.br

