

Programação Orientada a Objetos

Introdução - Java

Rone Ilídio
Thiago Oliveira



Plano de Ensino

- Deitel, H. M., Deitel, T. J. **Java – Como Programar**. Editora Pearson, 8ª edição, 2010.

- Anderson, J., Franceschi, H. **Java 6 - Uma abordagem ativa de aprendizado**. LTC, 2ª edição, 2010.

- Distribuição dos Pontos:

Exercícios	20 Pontos
Trabalhos	20 Pontos
Provas	60 Pontos
<hr/>	
Total	100 Pontos

Orientação a objetos

- O ser humano vê os objetos reais através de suas características e funções.

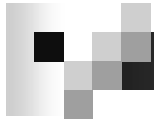
Características

- Tamanho
- Peso
- Cor
- Preço



Funções

- Liga
- Tira fotos
- Envia Mensagens
- Acessa Internet



Projetos Orientados a Objetos

- Modelagem e desenvolvimento de softwares por objetos.
- Os objetos possuem as características e funções de objetos reais.
- Mais próximo do pensamento humano.
- Facilita a modularização do software.



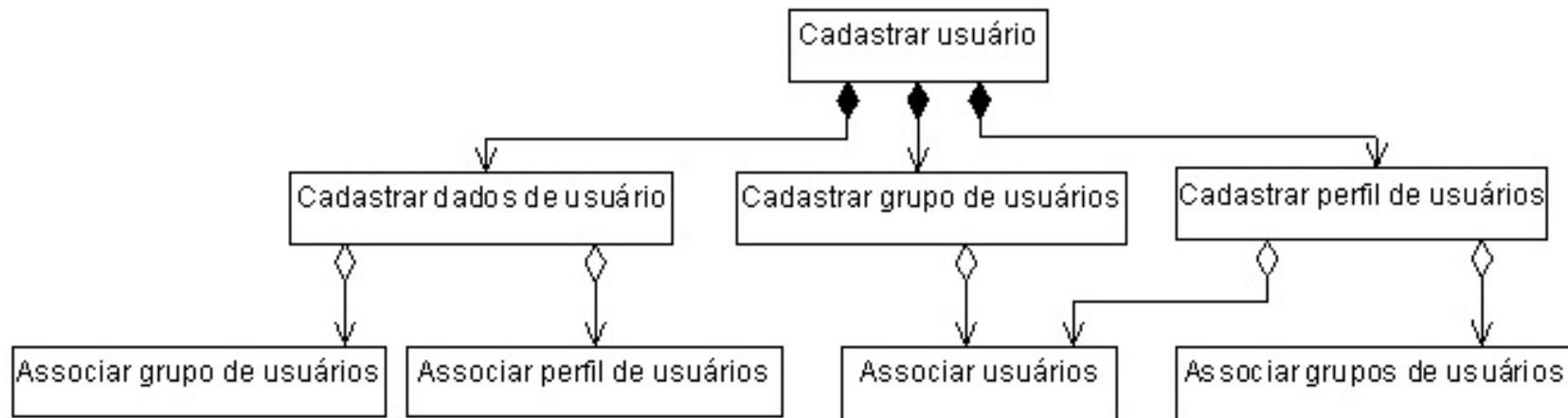
Paradigma

- *Objetivo de um projeto:* construir a ponte entre a especificação de uma aplicação e a implementação sob o modelo computacional.
 - Métodos de projetos top-down, OU
 - Projetos orientados por dados.

Paradigma

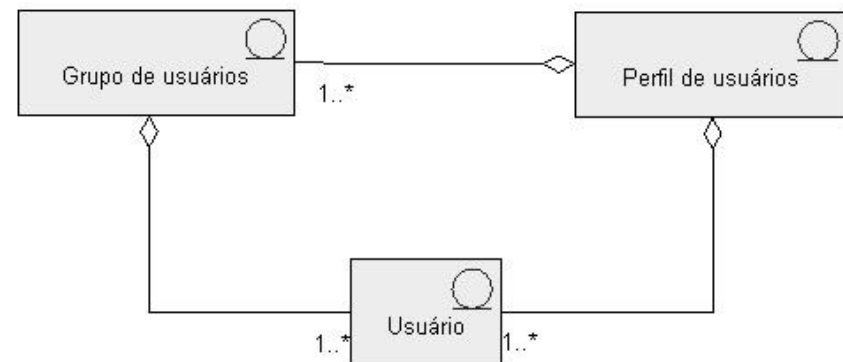
■ Métodos de projeto top-down:

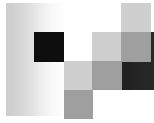
- Produz arquitetura baseada somente na função do sistema;
- Cada componente refina ou decompõe funções;
- Decomposição pára quando o nível de abstração da função refinada for diretamente implementável;



Paradigma

- Projeto orientado por dados:
 - Atenção voltada para as **Estruturas de dados** envolvidas na Especificação da Aplicação.
- **Exemplo:** serviço de cadastro de usuários em um sistema.
 - Usuário, Grupo de usuários. Perfil de usuários – Estruturas de dados.





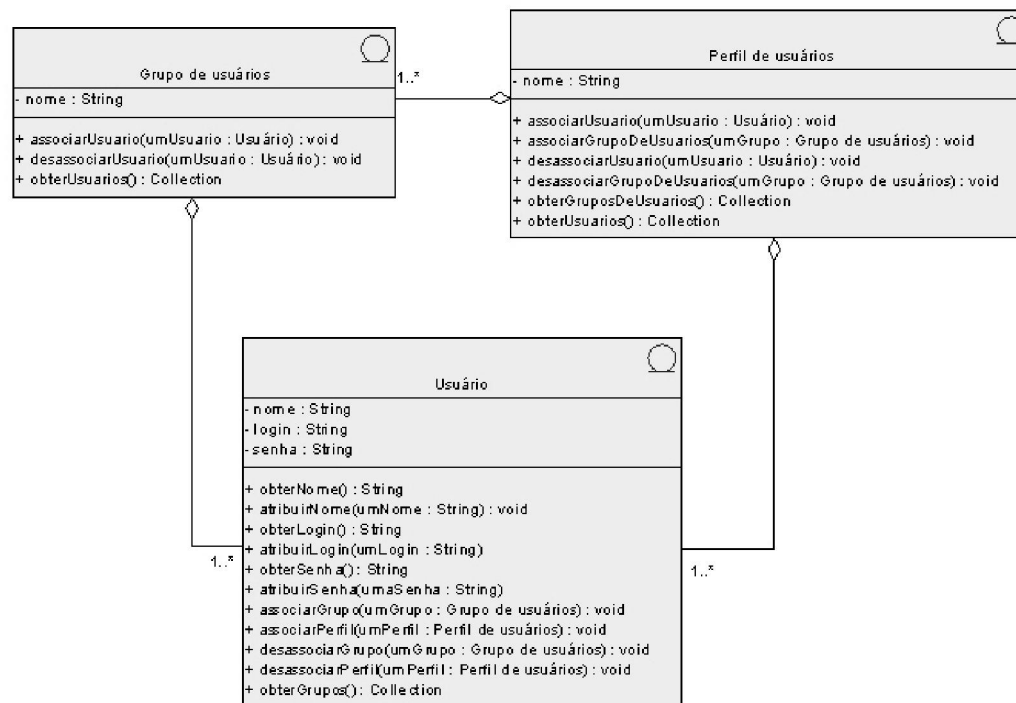
Paradigma

- Foco inteiramente voltado para Estrutura de dados também pode não ser a solução.
- Tipos abstratos de dados:
 - estrutura de dados + serviços específicos (funções);
 - provêm o equilíbrio necessário;
 - favorecem reusabilidade e extensabilidade.

Tipo Abstrato de Dados

■ Classe de estrutura de dados descrita por uma interface externa:

- Lista de serviços disponíveis;
- Propriedades destes serviços.





Sistemas OO

- As classes formam o núcleo de um programa OO.
- Os objetos provêm o comportamento, devendo ser criados apropriadamente.
- Classes suportam os conceitos de:
 - Abstração;
 - Encapsulação;
 - Proteção de dados;
 - Polimorfismo;
 - Hierarquia.



Classes

- A abstração está ligada aos seus objetivos.
 - Exemplo: abstração de uma TAD Pessoa.
- Encapsular consiste em incluir, proteger em uma cápsula, classe:
 - Proteção de dados visa garantir o acesso apenas sobre operações e atributos disponibilizados pela interface da classe;
 - Todos os atributos e operações de uma classe podem ser acessados pelas operações da mesma classe.



Linguagens OO

- Suporte à criação de objetos no computador.
 - ☐ Representação de objetos reais.
- Exemplos de linguagens:
 - ☐ C++
 - ☐ C#
 - ☐ Object Pascal
 - ☐ .Net
 - ☐ Java

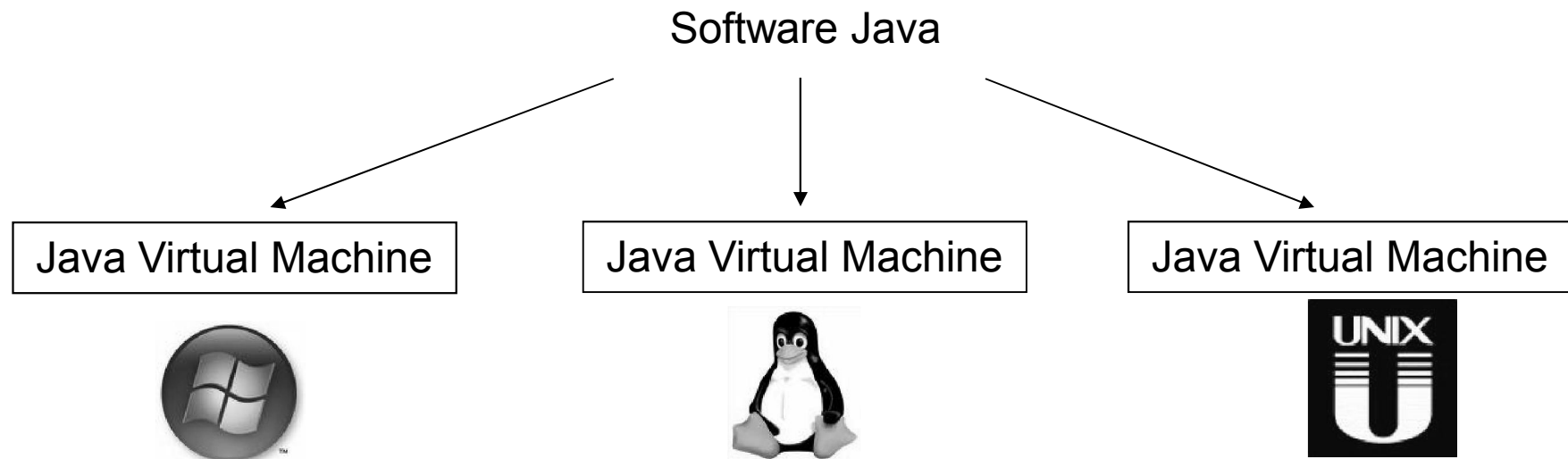


Linguagens OO

- Java é puramente orientada a objetos:
 - Não há forma de criar um programa, um procedimento ou uma função sem criar pelo menos uma classe.
- Em C++, existe essa possibilidade.

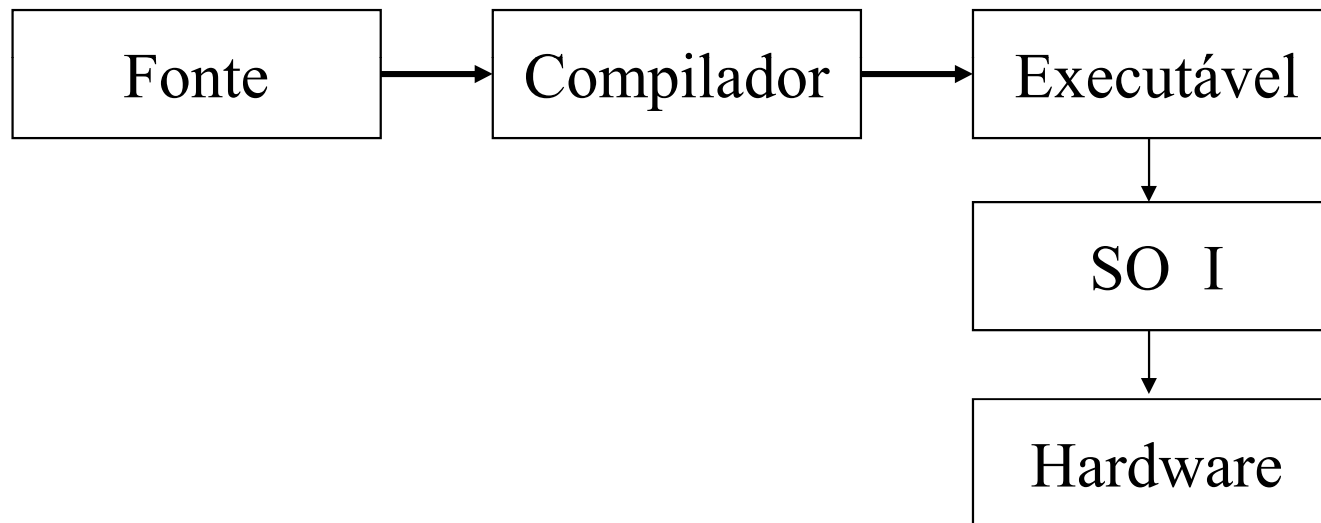
Java

- Totalmente orientada a objetos.
- Gratuita.
- Multi plataforma:



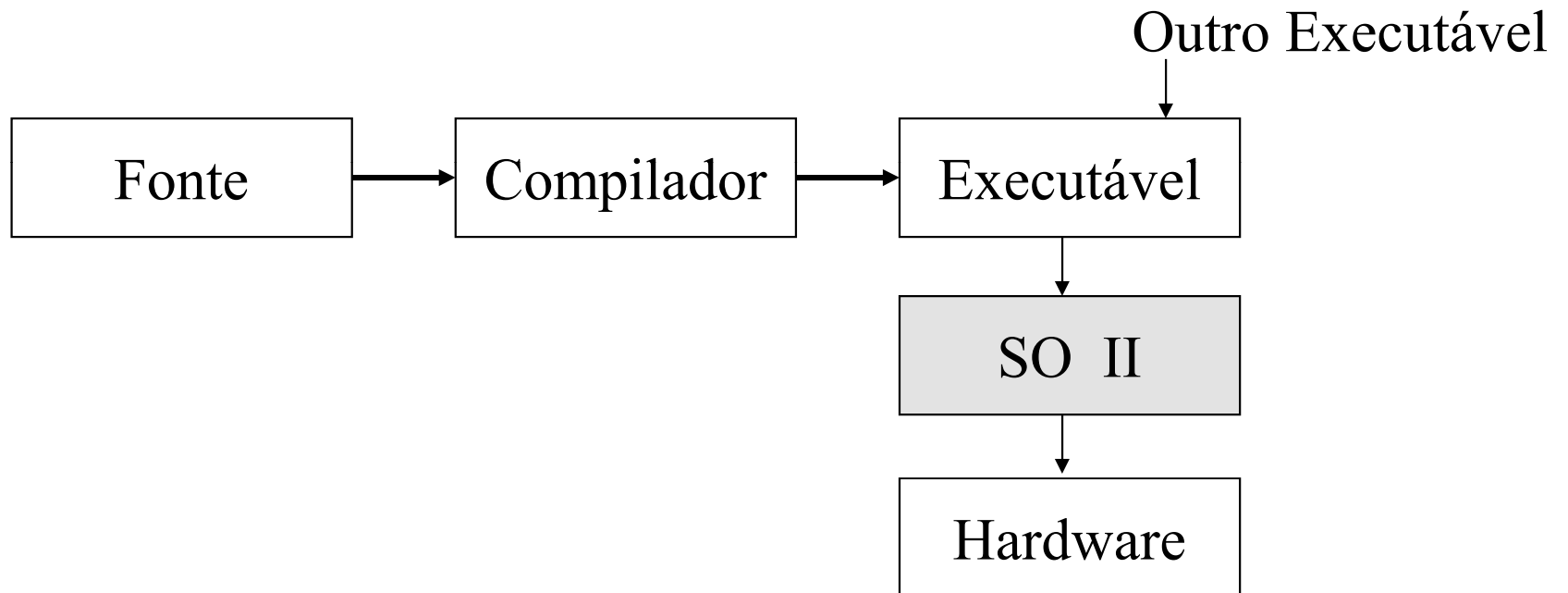
Java - Vantagens

Outras linguagens funcionam da seguinte forma:



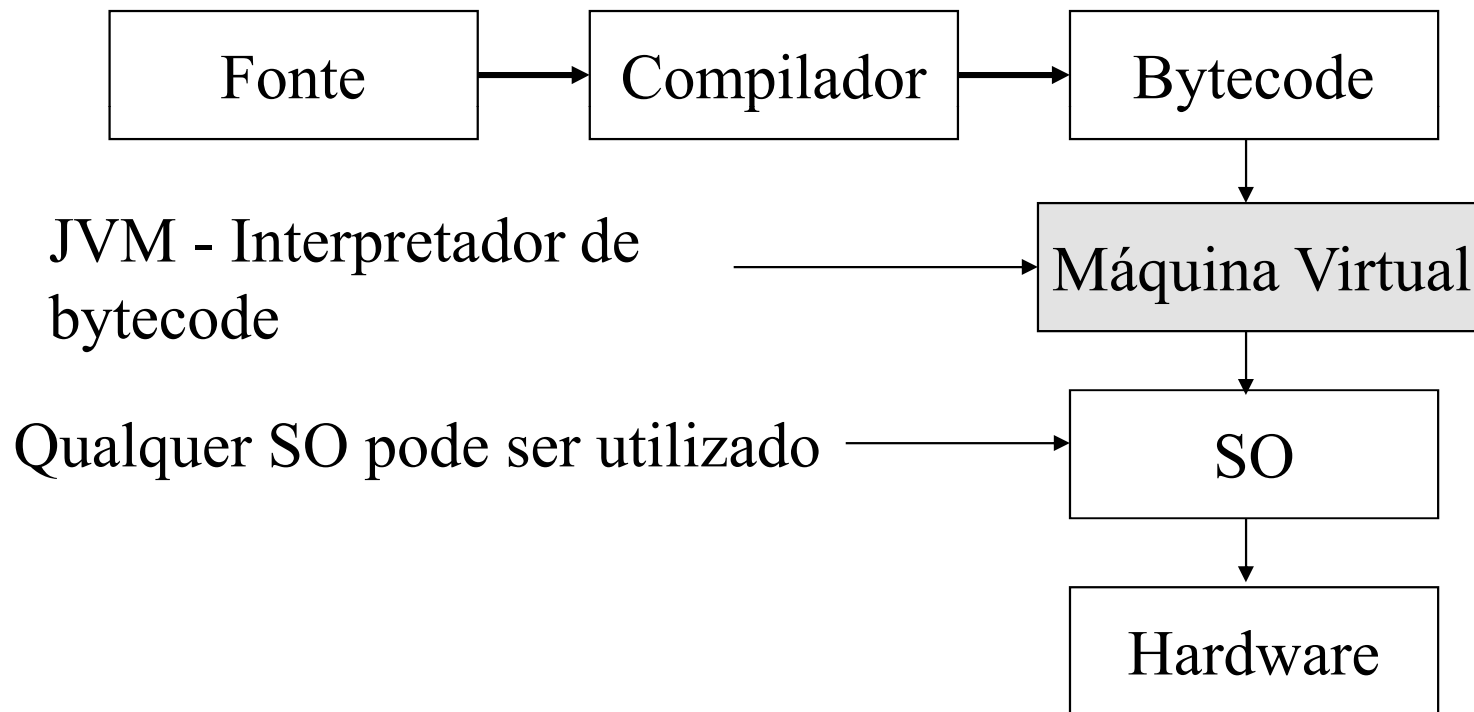
Java - Vantagens

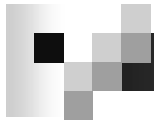
Outras linguagens funcionam da seguinte forma:



Java - Vantagens

Java é multiplataforma:



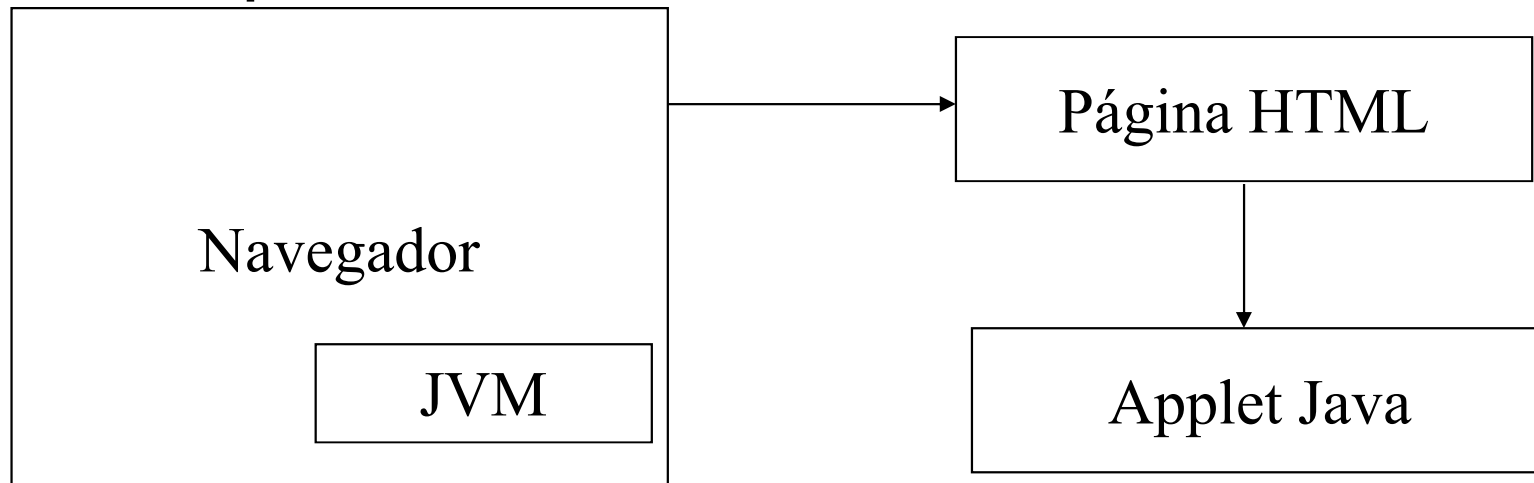


Java - Vantagens

- Derivado do C e do C++, possuindo a mesma sintaxe.
- Freeware: site da Sun (www.sun.com)
- Bibliotecas bastante desenvolvidas (API)
- Interface gráfica fácil de ser utilizada (GUI)

Java - Vantagens

- É executado por navegadores WEB (*browsers* como Firefox, Chrome e IE).
- Exemplo: site do Banco do Brasil.





Registros

- Em C ou C++:

```
typedef
```

```
    struct Cliente {  
        Char *Nome;  
        Char *Telefone;  
        int Idade;  
    }
```



Registros

```
typedef
```

```
    Struct registro{
```

```
        Char[20] Dado1;
```

```
        Int Dado2;
```

```
    }
```

```
void main(){
```

```
    registro r;
```

```
};
```



Registros x Classes

- Registros ou *structs* são uniões de dados em uma mesma estrutura.
- Classes são uniões de dados (atributos) e códigos (métodos) em uma mesma estrutura.



Classe

- Tipo abstrato de dados (TAD).
- Modelo ou protótipo de objetos reais.
- Possui a definição das características e funções desses objetos.
- Características → Atributos:
 - Definem o estado o objeto.
- Funções → Métodos:
 - Comportamento do objeto;
 - Operações sobre os dados.



Classe

- Semelhante a uma struct (C) ou record (Pascal).
 - Atributos: variáveis globais.
 - Métodos: funções que normalmente manipulam os atributos.
 - Métodos para leitura e escrita os atributos: interface de acesso.
 - Public: acesso externo
 - Private: acesso interno
- } Detalhes a frente



```
public class Pessoa{
```

```
    private String nome;  
    private int idade;
```

} ← Atributos

```
    public String getNome(){  
        return nome;  
    }
```

```
    public int getIdade(){  
        return idade;  
    }
```

```
    public void setNome(String n){  
        nome = n;  
    }
```

```
    public void setIdade(int i){  
        idade = i;  
    }
```

```
}
```

} ← Operações sobre os dados



Criação de objeto

```
public class Principal{  
    public static void main(String[] args){  
  
        Pessoa p;  
        p = new Pessoa();  
  
        //p.nome = "José";  
  
        p.setNome("José");  
        p.setIdade(18);  
  
        System.out.println("Nome="+ p.getNome());  
        System.out.println("Idade="+ p.getIdade());  
    }  
}
```



Criação de objeto

- Foi declarada a classe Principal
 - aplicativo Java;
 - utiliza classe Pessoa implementada.
- `Pessoa p = new Pessoa();`
 - **new** cria uma instância de Pessoa;
 - o objeto é denominado p.

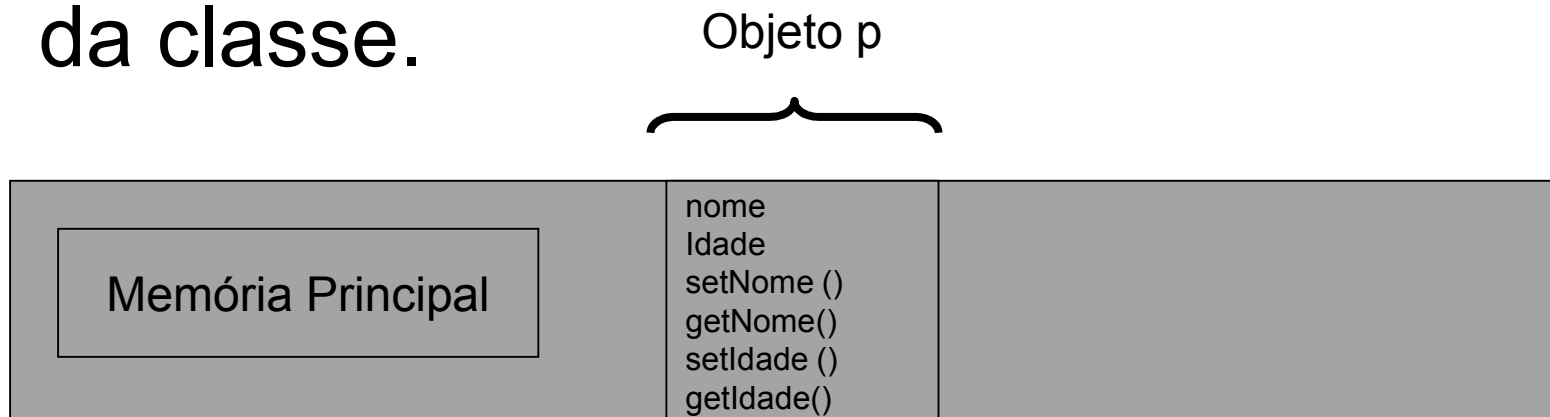


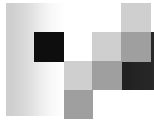
Objetos

```
public class Principal{  
    public static void main(String[] args){  
        Pessoa pessoa1, pessoa2;  
        pessoa1 = new Pessoa();  
        pessoa2 = new Pessoa();  
  
        pessoa1.setNome("José");  
        pessoa1.setIdade(18);  
        System.out.println("Nome="+ pessoa1.getNome());  
        System.out.println("Idade="+ pessoa1.getIdade());  
  
        pessoa2.setNome("João");  
        pessoa2.setIdade(35);  
        System.out.println("Nome="+ pessoa2.getNome());  
        System.out.println("Idade="+ pessoa2.getIdade());  
    }  
}
```

Objetos

- O comportamento do sistema é obtido através da interação entre objetos.
- Instâncias das classes.
- Área de memória com as características da classe.



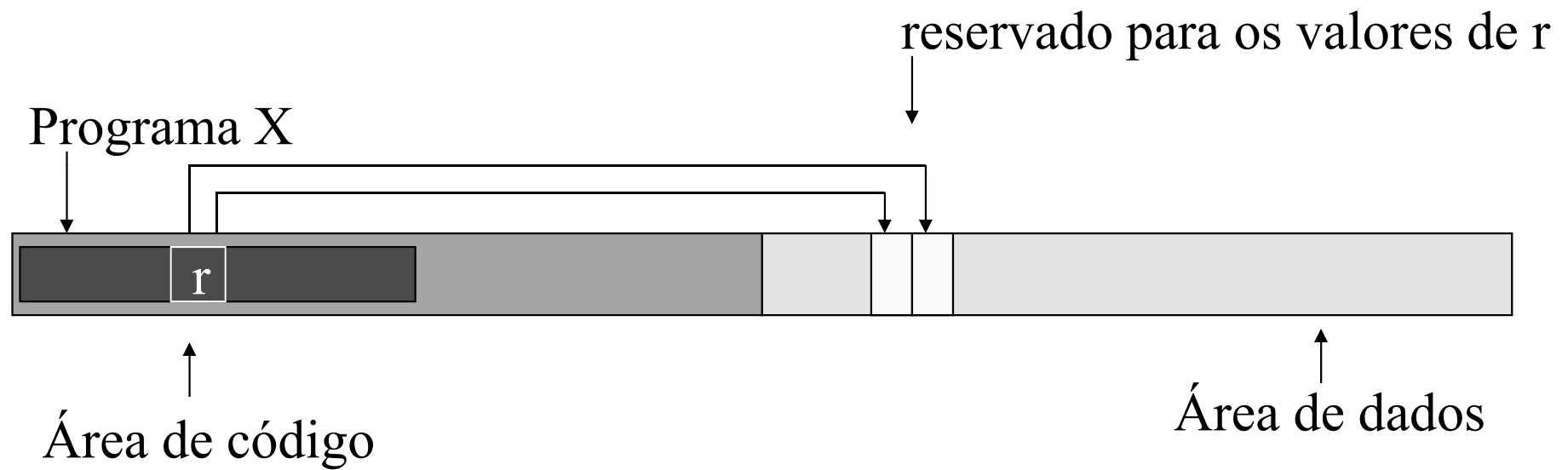


Memória

- Quando um programa é executado, é empilhado na memória principal do computador.
 - Dividida pelo SO em área de código e área de dados.
- Variáveis, registros e estruturas são ponteiros na área de código:
 - Apontam para posições de memória na área de dados, onde são colocados os valores.
- Classes possuem ponteiros para:
 - área de dados (atributos);
 - área de código (métodos).

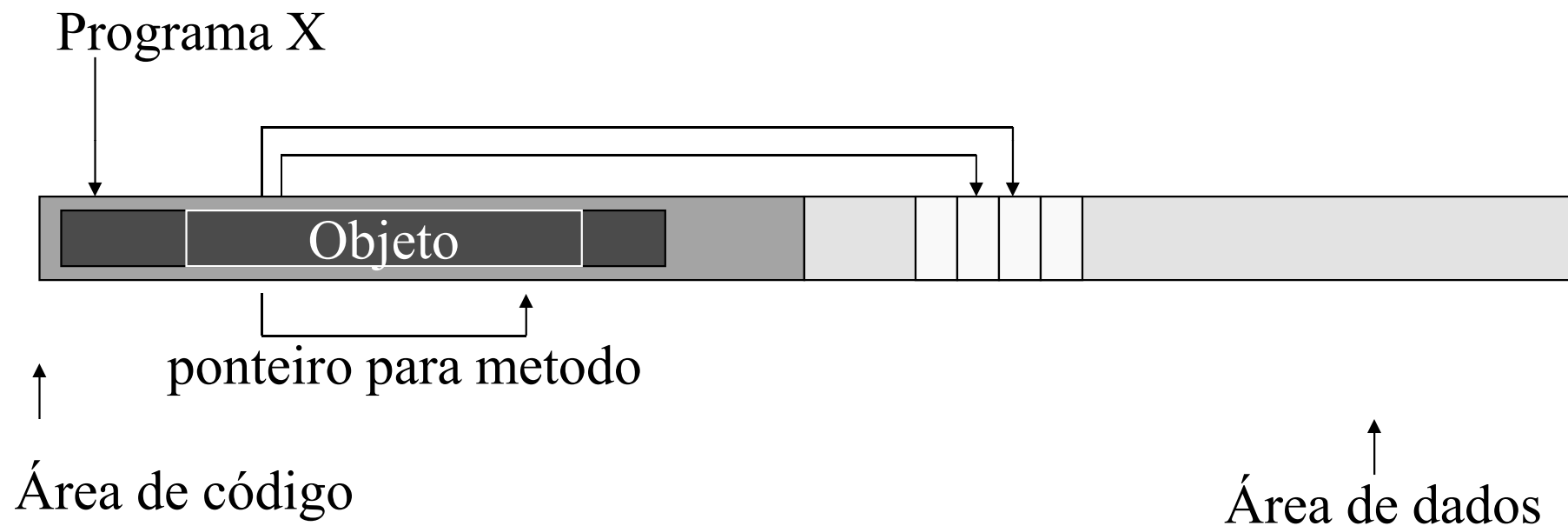
Memória

Memória Principal - Registros



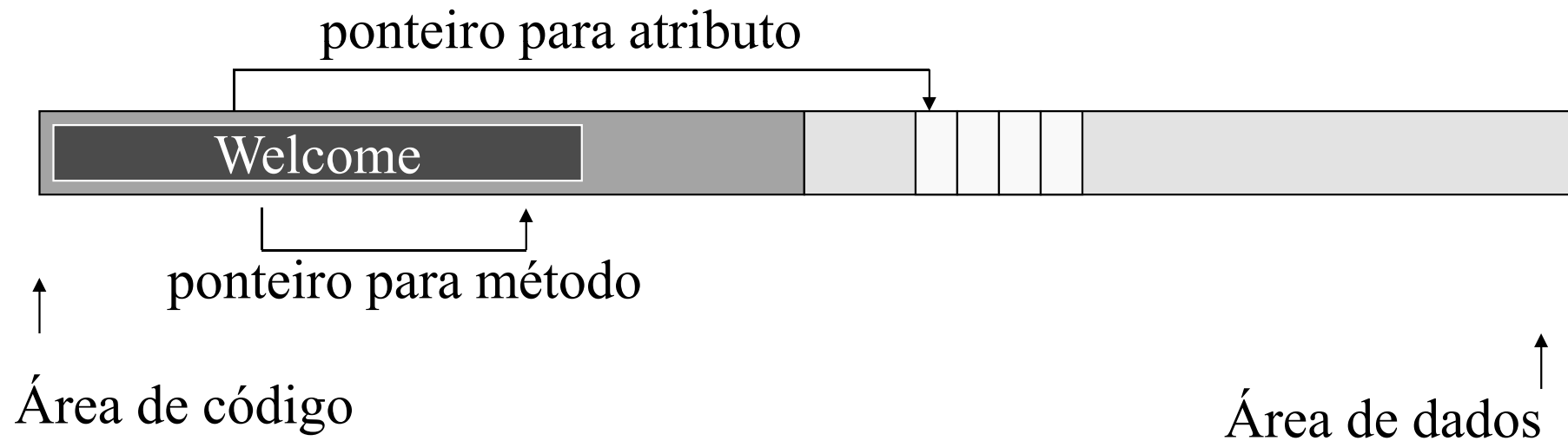
Memória

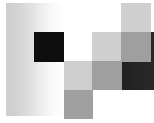
Memória Principal - Objetos



Memória

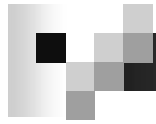
Memória Principal – Classes em Java





Aplicativos e Applets

- Aplicativos são executados localmente como um programa.
- Applets são chamados de dentro de uma página HTML e executados por um navegador WEB.



Java - Aplicativos

Primeiro programa – Hello World!

```
public class Welcome
{
    public static void main(String args[])
    {
        System.out.println("Hello World!");
    }
}
```



Java - Aplicativos

- **public class Welcome{}**

- Declaração da classe.

- **public static void main (String args []){}**

- Declaração do método principal, primeiro método a ser executado.

- **System.out.println("Hello Wolrd!");**

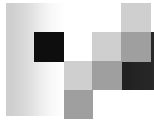
- Comando para imprimir na tela.



Java - Aplicativos

- O arquivo de ser salvo com o mesmo nome da classe mais a extensão .java, no caso Welcome.java
 - ☐ Java diferencia maiúsculas de minúsculas.

- Para executar pelo prompt do DOS vá até a pasta onde o Programa foi salvo e digite:
 - ☐ javac Welcome.java → para gerar o bytecode
Este comando gera o arquivo Welcome.class
 - ☐ java Welcome → para a JVM interpretar
Welcome.class



Java - Aplicativos

Segundo programa – Hello World! Versão 2.0

```
import javax.swing.JOptionPane;
public class Welcome2 {
    public static void main (String args []){
        JOptionPane.showMessageDialog(null,
            "Hello World!\nHello World!");
        System.exit(0);
    }
}
```



Java - Aplicativos

- **import javax.swing.JOptionPane**

- ☐ busca a classe JOptionPane, onde são implementadas caixas de diálogos, java.lang é a única API que não precisa importar.

- **public class Welcome2**

- ☐ declaração da classe Welcome2.

- **public static void main (String args [])**

- ☐ método principal, obrigatório para aplicativo ser interpretado.

- **JOptionPane.showMessageDialog(null, “Hello World!”)**

- ☐ exibe uma caixa de diálogo com um texto. O parâmetro **null** é usado para o posicionamento padrão (centro da tela).


- **System.exit(0)**

- ☐ finaliza o aplicativo, dever ser sempre utilizada em aplicativos que utilizam interface gráfica com o usuário.



Java - Aplicativos

- Terceiro programa: Adição.
- Este aplicativo trabalha com entrada de dados pelo usuário.
- Receber dois valores, converter para inteiro, somar os números e exibir o resultado.



```
import javax.swing.JOptionPane;
public class Adicao{
    public static void main(String args [ ]){
        String primeiro = JOptionPane.showInputDialog("Entre
                                                    com o primeiro número:");
        String segundo = JOptionPane.showInputDialog("Entre com o
                                                    segundo número:");

        int num1 = Integer.parseInt(primeiro);
        int num2 = Integer.parseInt(segundo);
        int soma = num1 + num2;

        JOptionPane.showMessageDialog(null, "A soma é: " + soma);
        System.exit(0);
    }
}
```



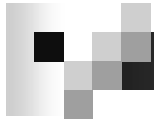
Java - Aplicativos

- **import javax.swing.JOptionPane**
 - Importa a classe JOptionPane
- **public class Adicao**
 - Declaração da classe Adicao
- **public static void main(String args [])**
 - Declaração do método principal
- **String primeiro, segundo**
 - Cria duas variáveis do tipo String
- **int num1, num2, soma**
 - Cria três variáveis do tipo inteiro



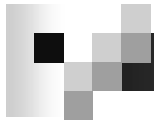
Java - Aplicativos

- **primeiro = JOptionPane.showInputDialog("Entre...")**
- **segundo =JOptionPane.showInputDialog("Entre...");**
 - Variáveis recebem dados String informados pelo usuário.
- **num1 = Integer.parseInt(primeiro);**
- **num2 = Integer.parseInt(segundo);**
 - Strings são convertidas em inteiro.
- **soma = num1 + num2;**
 - Soma dos valores inteiros.
- **JOptionPane.showMessageDialog(null, "A soma é : "
+ soma);**
 - Exibe o resultado final concatenado com uma mensagem.



Java - Applets

- É um aplicativo que interpretado por um contêiner, que na maioria das vezes é um navegador WEB.
- Chamado através de um arquivo HTML.
- Pode ser interpretado pelo aplicativo appletviewer do pacote Java.



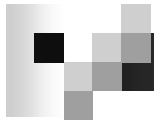
Java - Applets

- A classe JApplet possui três métodos que sempre são chamados pelo contêiner, são eles:
 - **init** - sempre chamado uma vez no início da execução do applet
 - **start** - chamado depois do init e toda vez que o usuário do navegador retornar para a página html em que o código reside
 - **paint** - chamado no início e sempre que a janela do applet for coberta por outra janela



```
import java.awt.Graphics;
import javax.swing.*;
public class MenuDesenho extends JApplet{
    int escolha;
    public void init(){
        String entrada = JOptionPane.showInputDialog(
            "Digite 1 para desenhar 10 linhas\n" +
            "Digite 2 para desenhar 10 retângulos\n" +
            "Digite 3 para desenhar 10 elipses");
        escolha = Integer.parseInt(entrada);
    }
}
```

// Continua no próximo slide...



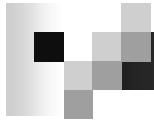
```
public void paint(Graphics g){
    super.paint(g);
    for (int i=1; i <= 10 ; i++){
        switch(escolha){
            case 1:
                g.drawLine(10, 10, 250, 10*i);
                break;
            case 2:
                g.drawRect(10, 10*i, 50, 50 + 10 * i);
                break;
            case 3:
                g.drawOval(10, 10*i, 50, 50 + 10 * i);
                break;
            default:
                g.drawString("Comando Inválido!",10, 10 * i);
                break;
        }
    }
}
```



Java - Applets

■ **WelcomeLines.java**

```
import java.awt.Graphics;
import javax.swing.JApplet;
public class WelcomeLines extends JApplet{
    public void paint(Graphics g){
        super.paint(g);
        g.drawLine(15,10,210,10);
        g.drawLine(15,30,210,30);
        g.drawString("Linhas acima e abaixo!",25,25);
    }
}
```

Java - Applets

- **import java.awt.Graphics;**
 - Pacote que permite a um applet desenhar.
- **import javax.swing.JApplet;**
 - Todo applet é filho desta classe. A classe Applet pode ser utilizada, mas não trataremos isso.
- **public class WelcomeLines extends JApplet**
 - Declaração da classe WelcomeLines, que é filha de JApplet (o conceito de herança será explicado posteriormente).
- **public void paint(Graphics g)**
 - Método chamado pelo applet após o init quando é iniciado e sempre que outra janela entrar na frente da janela do applet.



Java - Applets

- **super.paint(g);**
 - Chama o método paint da classe JApplet.
- **g.drawLine(15,10,210,10);**
- **g.drawLine(15,30,210,30);**
 - Desenharam linhas na tela. Os argumentos são X inicial e Y inicial, X final e Y final.
- **g.drawString("Linhas acima e abaixo!",25,25);**
 - Desenha strings na tela. Os argumentos são: o texto, posições X e Y iniciais.



Java - Applets

■ WelcomeLines.html

```
<HTML>
```

```
<HEAD>
```

```
<TITLE> Página Exemplo </TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
<applet code="WelcomeLines.class" width="300"  
  height="45"></applet>
```

```
</BODY>
```

```
</HTML>
```



Java - Applets

- Para interpretar um applet vá até a pasta onde o arquivo .java foi salvo.
- Compile este arquivo:
 - `javac WelcomeLines.java`
- Agora execute o arquivo WelcomeLines.html:
 - `appletviewer WelcomeLines.html`
- O appletviewer desconsidera o código html, ou seja, só interpreta applet.