Módulo 5

Programação Orientada a Objetos

Programação Orientada a Objetos I

Java

(Rone Ilídio)

Programação Orientada a Objetos

- Registro em Pascal:

```
Type
    Cliente = Record
        Nome: String;
      Telefone: String;
      Idade: integer;
    end;
```

Programação Orientada a Objetos

- Registro em C ou C++:

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

Programação Orientada a Objetos

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

Programação Orientada a Objetos

- Quando um programa é executado ele é empilhado na memória principal do computador, a qual é dividida pelo SO em área de código e área de dados.
- Variáveis, registros e estruturas são ponteiros na área de código que apontam para posições de memória na área de dados, onde são colocados os valores.
- Classes possuem ponteiros para a área de dados (atributos) e ponteiros para a área de código (métodos).

Programação Orientada a Objetos

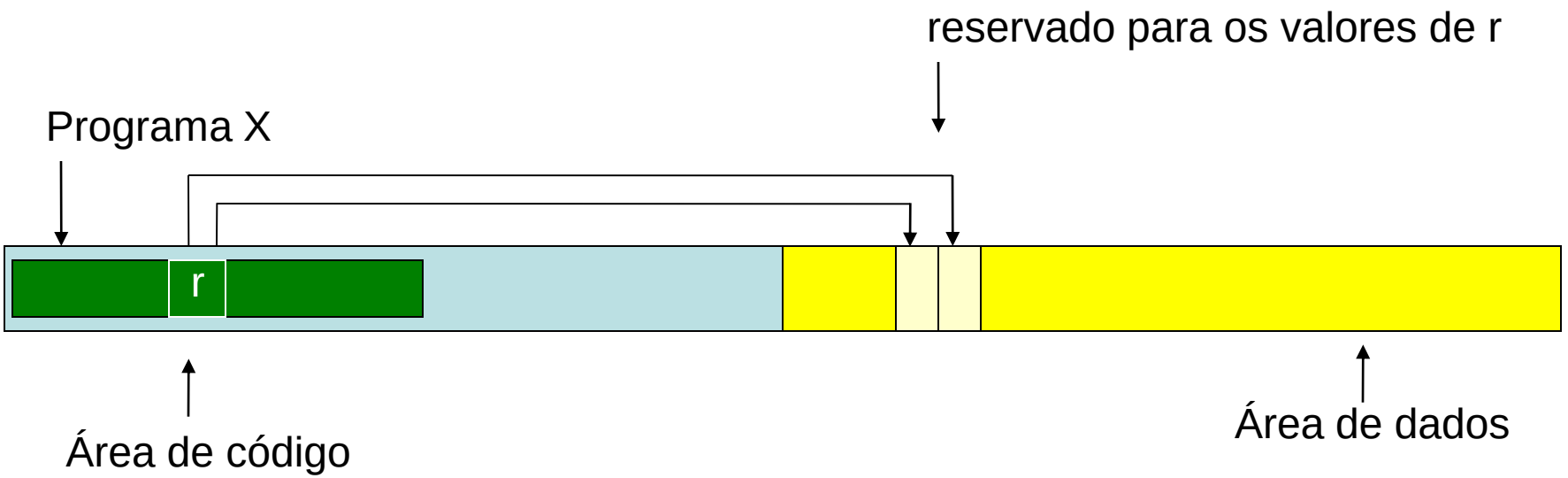
```
Program X;
type
    Registro = record
        Dado1: String;
        Dado2: Integer;
    end;

var
    r: Registro;
...
```

Programação Orientada a Objetos

```
typedef
    Struct registro{
         Char[20]  Dado1;
        Int Dado2;
    }
void main(){
    registro r;
};
```

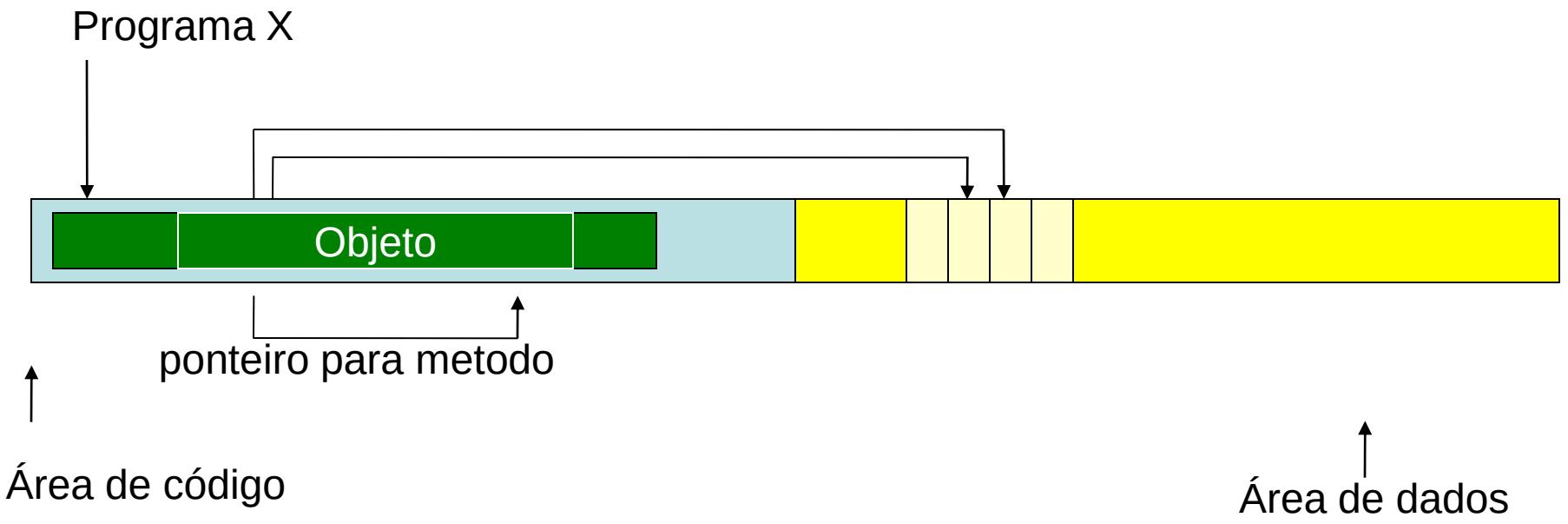
Programação Orientada a Objetos

Memória Principal -Registros



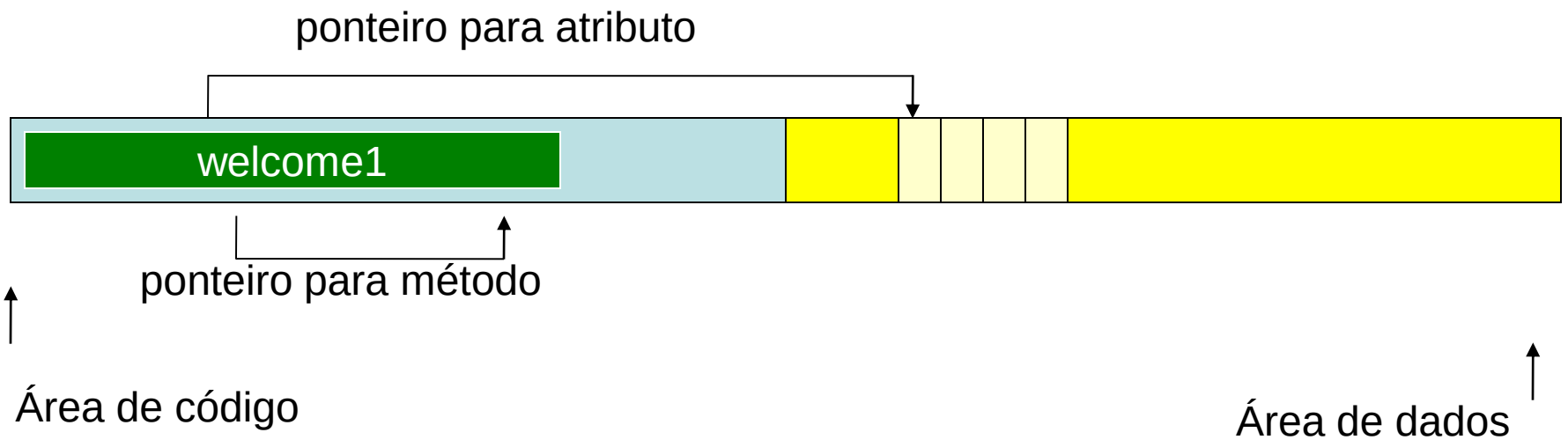
Programação Orientada a Objetos

Memória Principal - Objetos



Programação Orientada a Objetos

Memória Principal – Classes em Java



Programação Baseada em Objetos

- Até agora o objetivo foi fornecer uma base concreta de programação estruturada
- Alguns conceitos já foram abordados, como classe e método
- O exemplo a seguir utiliza os conceitos de orientação a objetos
- Obs: essas classes devem ser salvas no mesmo diretório

```
public class Pessoa{
    private String nome;private int idade;

    public String getNome(){
        return nome;
    }
    public int getIdade(){
        return idade;
    }
    public void setNome(String n){
        nome = n;
    }
    public void setIdade(int i){
        idade = i;
    }
}
```

Programação Baseada em Objetos

- Foi declarada a classe Pessoa
- Ela não é um *applet* nem um aplicativo (não possui nem o método *main*, nem os método típicos de um *applet*)
- Possui duas variáveis de instância declaradas como *private* (atributos) e cinco métodos declaradas como *public*
- Possui um método com o mesmo nome da classe – método construtor

Programação Baseada em Objetos

- A variáveis de instância são declaradas como *private* para evitar que outras classes diretamente façam acesso a seus valores
- Elas só podem ser acessadas pelos métodos da classe pessoa.
- O métodos são declarados como public pois podem se acessados em qualquer lugar

```
public class UsaPessoa{
   public static void main(String[] args){
       Pessoa p1 = new Pessoa();
       Pessoa p2 = new Pessoa();
       p1.setNome(“Zé”);
       p1.setIdade(18);

       p2.setNome(“Mané");
       p2.setIdade(20);
       System.out.println("Nome="+ p1.getNome());
       System.out.println("Idade="+ p1.getIdade());

       System.out.println("Nome="+ p2.getNome());
       System.out.println("Idade="+ p2.getIdade());
   }
}
```

Programação Baseada em Objetos

- Foi declarada a classe UsaPessoa, que é um aplicativo
- Pessoa p = new Pessoa(); → cria uma instância de Pessoa na memória, ou seja, cria na memória um objeto denominado p que é do tipo Pessoa.
- No momento da criação é executado o método contrutor (p.Pessoa()), sempre!

Programação Baseada em Objetos

- São feitas chamadas aos métodos de p
- Os métodos iniciados com *set* modificam os valores dos atributos de p
- Os métodos iniciados com *get* retorna os valores contidos nos atributos de p
- Se tentarmos acessar o valor de um atributo declarado como *private* ocorre erro na compilação.

Programação Baseada em Objetos

Importante

- Pessoa e UsaPessoa devem ser salvas no mesmo diretório
- A compilação de UsaPessoa automaticamente compila Pessoa

Métodos Construtores

- São métodos que possuem o mesmo nome da classe
- Não possuem tipo de retorno
- São executados pelo operador *new*
- Normalmente são utilizados para inicializar variáveis de instânia
- Ex: método Pessoa() da classe pessoa

Duas classes em um mesmo arquivo

- Várias classes podem ser criadas dentro do mesmo arquivo fonte, mas só uma pode ser declarada como *public*
- As demais só poderão ser acessas dentro do pacote onde se encontram
- O arquivo deve ter o nome da classe *public*
- A compilação deste arquivo gera dois arquivos .class

```
import java.awt.Graphics;
public class PosicaoSonda2 extends JApplet{
    Sonda2 s1;
    Sonda2 s2;
    public void init(){
        s1 = new Sonda2();
        String entrada;

        //Inserindo dados da primeira sonda
        entrada = JOptionPane.showInputDialog(null, "Informe o nome da sonda");
        s1.setNome(entrada);
        entrada = JOptionPane.showInputDialog(null, "Qual a longitude?");
        s1.setX(Integer.parseInt(entrada));
        entrada = JOptionPane.showInputDialog(null, "Qual a latitude?");
        s1.setY(Integer.parseInt(entrada));

    }//init

    public void paint (Graphics g){
        g.drawOval(s1.getX(), s1.getY()+10,6,6);
        g.drawString(s1.getNome(),s1.getX(), s1.getY());
    }
}
//continua no mesmo arquivo
```

```
//continuação
class Sonda2{
    private String nome; private int x;   private int y;
    public Sonda2(){
        nome = "";
        x = 0;
        y = 0;
    }
    public String getNome(){
        return nome;
    }
    public void setNome(String n){
        nome = n;
    }
    public int getX(){
        return x;
    }
    public void setX(int valor){
        x = valor;
    }
    public int getY(){
        return y;
    }
    public void setY(int valor){
        y = valor;
    }
}
```

Método toString

- Esse método é executado toda vez que chamamos um objeto somente pelo identificador, sem especificarmos um método ou atributo public

```
public class Pessoa{
        private String nome; private int idade;
        public Pessoa(){
        nome = "";
        idade = 0;
    }
    public String getNome(){
        return nome;
    }
    public int getIdade(){
        return idade;
    }
    public void setNome(String n){
        nome = n;
    }
    public void setIdade(int i){
        idade = i;
    }
    public String toString(){
        return nome + idade;
    }
}
```

```
public class UsaPessoa
{
    public static void main(String[] args)
    {
        Pessoa p = new Pessoa();
        System.out.println("Nome="+ p.getNome());
        System.out.println("Idade="+ p.getIdade());

        p.setNome("Ze");
        p.setIdade(18);

        System.out.println("Nome="+ p.getNome());
        System.out.println("Idade="+ p.getIdade());

        System.out.println("Idade="+ p);

    }
}
```

Referência this

- A referência *this* é utilizada para acessar métodos ou variáveis de instância, dentro do corpo de uma classe
- A classe Pessoa poderia se declarada da seguinte forma

```
public class Pessoa{
private String nome; private int idade;
public Pessoa(){
      nome = "";
      idade = 0;
}
public String getNome(){
      return nome;
}
public int getIdade(){
      return idade;
}
public void setNome(String nome){
      this.nome = nome;
}
public void setIdade(int idade){
      this.idade = idade;
}
public String toString(){
      return this.nome + idade;
}
}
```

Padrão de Nome

- Todas as palavras de uma classes devem ser iniciadas por maiúsculo

Ex: classe PessoaFisica, VendaPrazo

- Com exceção da primeira letra, todas as palavras de uma classes devem ser iniciadas por maiúsculo

Ex: buscaVenda, apagaRegistro

- Variáveis tudo em minúsculo

Escopo de Classe

- As variáveis de instância e métodos da classe pertencem ao escopo dessa classe
- Dentro do escopo de uma classe, os membro da classe estão acessíveis para todos os métodos dessa classe e podem ser chamados simplesmente pelo nome
- Fora do escopo da classe, o membros visíveis (ex: membros public) são chamados através do nomeDaReferenciadoObjeto.nomeDoMembro

Escopo de Classe

- Variáveis são visíveis somente dentro do escopo do método
- Uma variável local a um método sobrescreve uma variável de instância com o mesmo nome.Neste caso, para acessar a variável de instância usa-se a palavra reservada this.nomeVariável.

Escopo de Classe

```
import javax.swing.*;
public class escopo extends JApplet{
    int x=0;
    public void init (){
        int x=1;
        JOptionPane.showMessageDialog(null, "x local: " + x);
        JOptionPane.showMessageDialog(null, "x variavel de
                                    instancia: " + this.x);
        {
            int y=9;
            JOptionPane.showMessageDialog(null, "Valor de
                                                y: " + y);
        }
    }
}
```

Vetor de Objetos

- Como toda classe é tipo, pode-se criar vetores com ela
- Cada objeto do vetor deve ser inicializado separadamente
- Exemplo:

```
public  class Cliente{
    private String nome;
    private String telefone;
    public void setNome(String nome){
        this.nome = nome;
    }
    public String getNome(){
        return nome;
    }
    public void setTelefone(String telefone){
        this.telefone = telefone;
    }
    public String getTelefone(){
        return telefone;
    }
}
```

```
import javax.swing.*;
public class VetorCliente {
public static void main(String args[]){
    int t = 2;
    Cliente v[] = new Cliente[t];
    for (int i=0; i<t; i++){
            v[i] = new Cliente();
            String e = JOptionPane.showInputDialog("Nome");
            v[i].setNome(e);
            e =  JOptionPane.showInputDialog("Telefone");
            v[i].setTelefone(e);
    }
    String s = "";
    for (int i=0; i<t; i++){
            s = s + "\nNome:" + v[i].getNome()+"  Telefone:"+ v[i].getTelefone();
    }
    JOptionPane.showMessageDialog(null, s);
    }
}
```