

# *Programação Orientada a Objetos*

Aula 07

Gustavo Willam Pereira

# Composição de Objetos

- Objetos como variáveis de instância de outras classes
  - Um objeto da classe Conta precisa ter um cliente, então porque não incluir uma referência a um objeto Cliente como membro do objeto Conta
  - Essa capacidade é chamada de composição
  - Uma classe pode ter referências a objetos de outras classes como membros.

```
// classe Pessoa
```

```
public class Pessoa
{
    private
        String nome; //nome da pessoa
        String endereco; // endereço da pessoa
        String cpf; // cpf da pessoa

    public Pessoa( ) //construtor default da classe
    {
        setPessoa(" ", " ", " ");
    }
    //métodos set
    public void setPessoa(String n, String e, String c)
    {
        nome = n;
        endereco = e;
        cpf = c;
    }
    public void setNome(String n) //configurar o nome
    {
        nome = n;
    }

    public void setEndereco(String e) //configurar o endereço
    {
        endereco = e;
    }

    public void setCpf(String c) //configurar o cpf
    {
        cpf = c;
    }
}
```

```
//métodos get
    public String getNome( )
    {
        return nome;
    }

    public String getEndereco( )
    {
        return endereco;
    }

    public String getCpf( )
    {
        return cpf;
    }

} //fim da classe
```

// classe conta bancaria com cliente

public class Conta

{

private

int agencia; //agencia a qual a conta pertence

int num\_conta; // numero da conta

double saldo; // saldo atual da conta

Pessoa cliente; // dados do cliente da conta

public Conta( ) //construtor default da classe

{

setConta(0, 0, 0);

cliente = new Pessoa( );

}

//métodos set

public void setConta(int a, int nc, double s)

{

agencia = a; num\_conta = nc; saldo = s;

}

public void setCliente(String n, String e, String c)

{ cliente.setPessoa(n, e, c); }

public void setAgencia(int ag) //configurar a agencia

{ if ( ag < 0 ) agencia = 0;

else agencia = ag; }

public void setNum\_Conta(int nc) //configurar o nr. conta

{ if ( nc < 0 ) num\_conta = 0;

else num\_conta = nc; }

public void setSaldo(double s) //configurar o saldo

{ saldo = s; }

//métodos get

public int getAgencia( )

{ return agencia; }

public int getNum\_Conta( )

{ return num\_conta; }

public double getSaldo( )

{ return saldo; }

public String getNomeCliente( )

{ return cliente.getNome( ); }

public String getEnderecoCliente( )

{ return cliente.getEndereco( ); }

public String getCpfCliente( )

{ return cliente.getCpf( ); }

public void Deposito(double valor)

{ saldo = saldo + valor; }

public void Saque(double valor)

{ saldo = saldo - valor; }

} //fim da classe

```
import javax.swing.JOptionPane;
```

```
public class Movimento  
{
```

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

```
    {
```

```
        int choice = 0; //contem a escolha do usuário
```

```
        double val; //variável para conter o valor movimentado
```

```
        int agenc, ct; //variáveis para conter a agencia e conta
```

```
        Conta C1 = new Conta();
```

```
        while (choice != 8)
```

```
        {
```

```
            choice = Integer.parseInt(Instrua_Usuario());
```

```
            switch (choice)
```

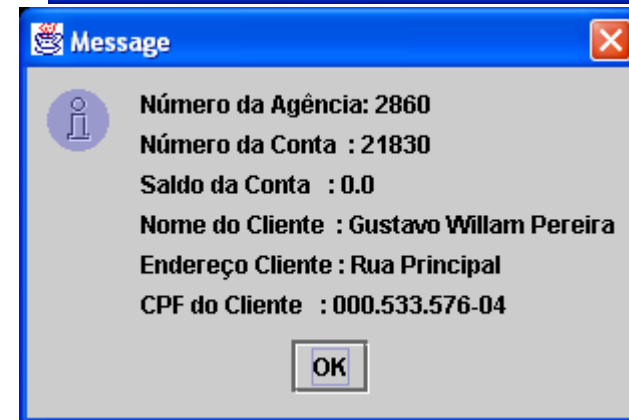
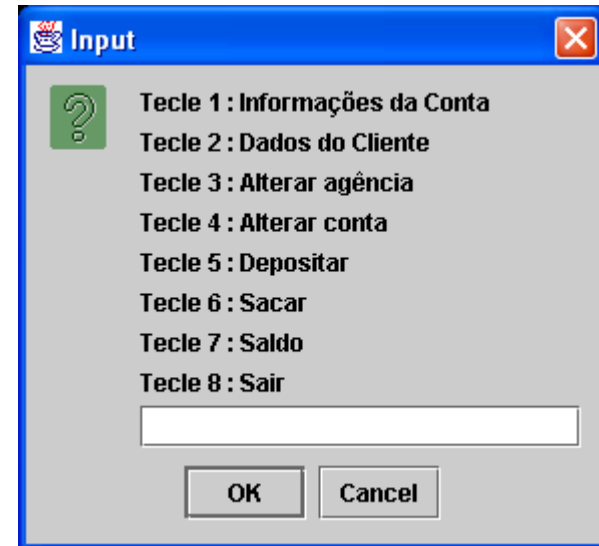
```
            {
```

```
                case 1:
```

```
                    JOptionPane.showMessageDialog(null,
```

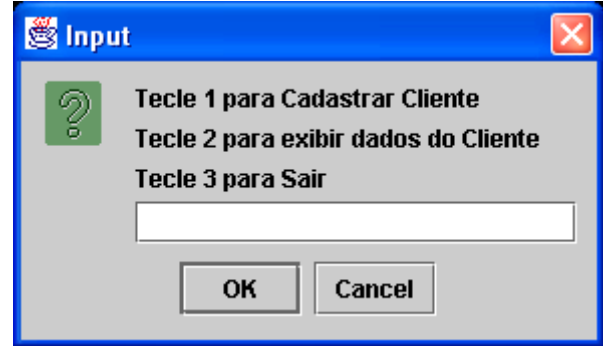
```
                        "Número da Agência: " + C1.getAgencia() +  
                        "\nNúmero da Conta : " + C1.getNum_Conta() +  
                        "\nSaldo da Conta : " + C1.getSaldo() +  
                        "\nNome do Cliente : " + C1.getNomeCliente() +  
                        "\nEndereço Cliente : " + C1.getEnderecoCliente() +  
                        "\nCPF do Cliente : " + C1.getCpfCliente());
```

```
                    break;
```



case 2:

```
Dados_cliente(C1);  
break;
```



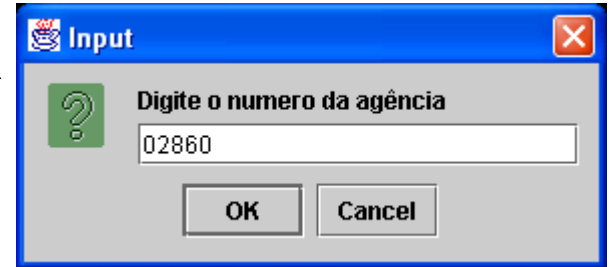
Input

Tecla 1 para Cadastrar Cliente  
Tecla 2 para exibir dados do Cliente  
Tecla 3 para Sair

OK Cancel

case 3:

```
agenc = Integer.parseInt(JOptionPane.showInputDialog(  
    "Digite o numero da agência"));  
  
C1.setAgencia(agenc);  
break;
```



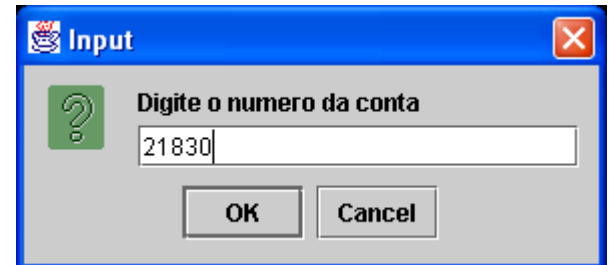
Input

Digite o numero da agência

OK Cancel

case 4:

```
ct = Integer.parseInt(JOptionPane.showInputDialog(  
    "Digite o numero da conta"));  
  
C1.setNum_Conta(ct);  
break;
```



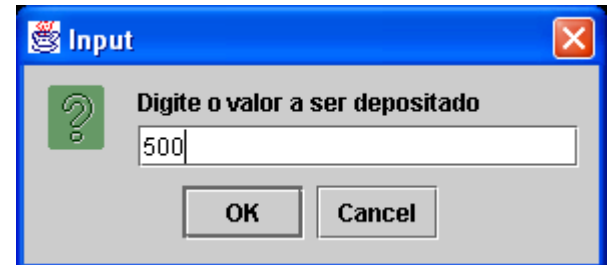
Input

Digite o numero da conta

OK Cancel

case 5:

```
val = Double.parseDouble(JOptionPane.showInputDialog(  
    "Digite o valor a ser depositado"));  
  
C1.Deposito(val);  
break;
```



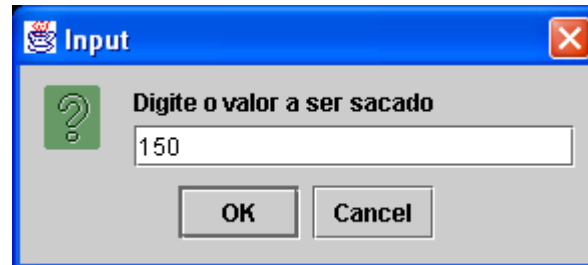
Input

Digite o valor a ser depositado

OK Cancel

case 6:

```
val = Double.parseDouble(JOptionPane.showInputDialog(  
    "Digite o valor a ser sacado"));  
  
C1.Saque(val);  
break;
```



Input

Digite o valor a ser sacado

OK Cancel

case 7:

```
val = C1.getSaldo();  
JOptionPane.showMessageDialog(null, "Saldo de :" + val);  
break;
```

case 8:

```
break;
```

default:

```
JOptionPane.showMessageDialog(null, "Opção inválida" );
```

```
}
```

```
}
```

```
System.exit(0);
```

```
}
```

public static String Instrua\_Usuario()

```
{
```

```
String input;
```

```
input = JOptionPane.showInputDialog("Tecle 1 : Informações da Conta \n" +
```

```
"Tecle 2 : Dados do Cliente \n" +
```

```
"Tecle 3 : Alterar agência \n" +
```

```
"Tecle 4 : Alterar conta \n" +
```

```
"Tecle 5 : Depositar \n" +
```

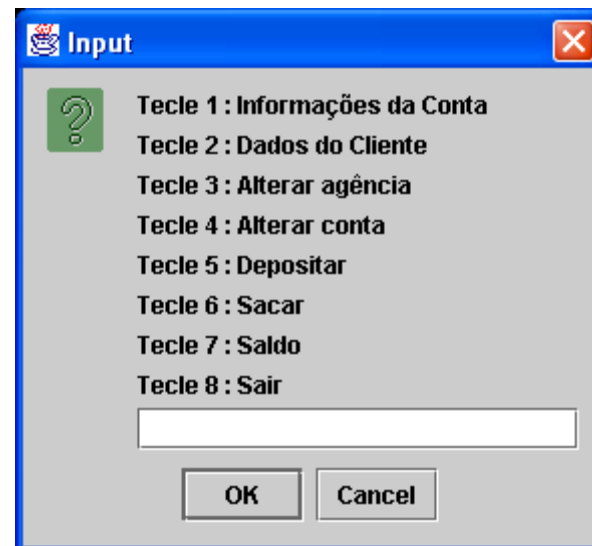
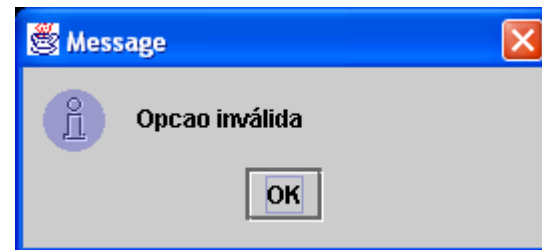
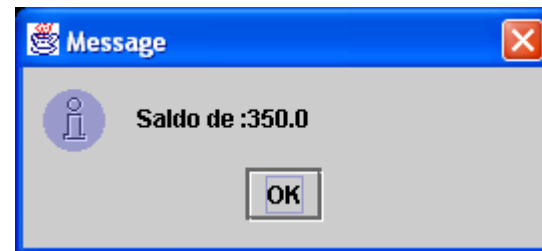
```
"Tecle 6 : Sacar \n" +
```

```
"Tecle 7 : Saldo\n" +
```

```
"Tecle 8 : Sair");
```

```
return input;
```

```
}
```



```
public static void Dados_cliente( Conta C1)
```

```
{
```

```
    String opcao;
```

```
    int op;
```

```
    // dados do cliente (nome, endereço, cpf)
```

```
    String name, adress, cp;
```

```
    opcao = JOptionPane.showInputDialog(  
        "Tecle 1 para Cadastrar Cliente" +  
        "\nTecle 2 para exibir dados do Cliente" +  
        "\nTecle 3 para Sair");
```

```
    op = Integer.parseInt(opcao);
```

```
    switch (op)
```

```
    {
```

```
        case 1:
```

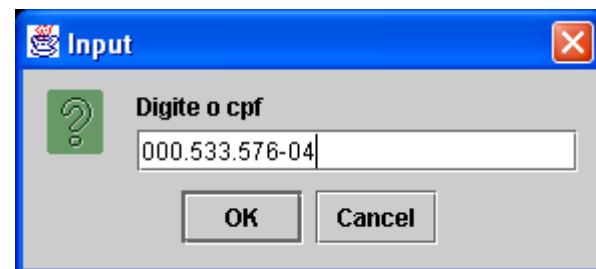
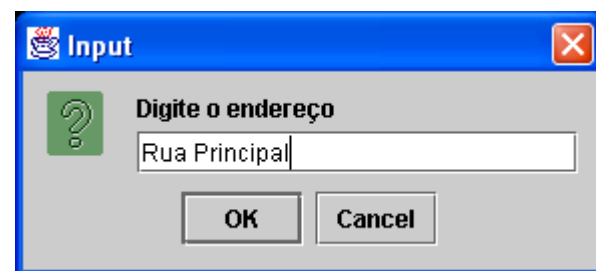
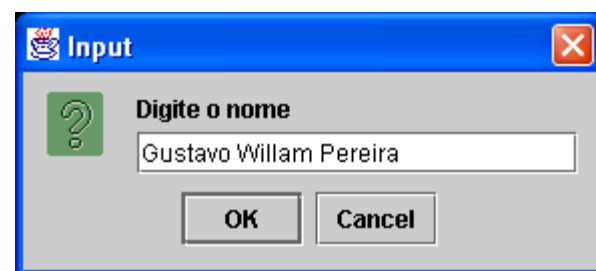
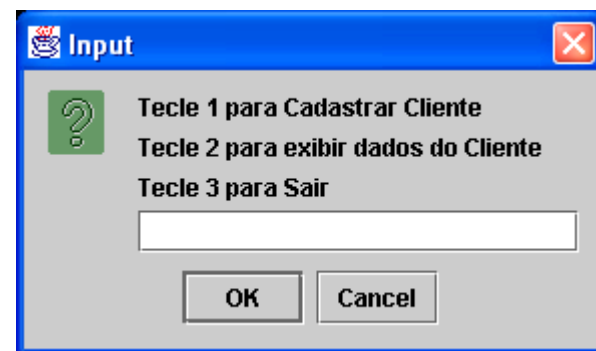
```
            name = JOptionPane.showInputDialog( "Digite o nome");
```

```
            adress = JOptionPane.showInputDialog("Digite o endereço");
```

```
            cp = JOptionPane.showInputDialog("Digite o cpf");
```

```
            C1.setCliente(name, adress, cp);
```

```
            break;
```





case 2:

```
JOptionPane.showMessageDialog(null,  
    "Nome do Cliente: " + C1.getNomeCliente() +  
    "\nEndereço      : " + C1.getEnderecoCliente() +  
    "\nCPF           : " + C1.getCpfCliente());
```

break;

case 3:

break;

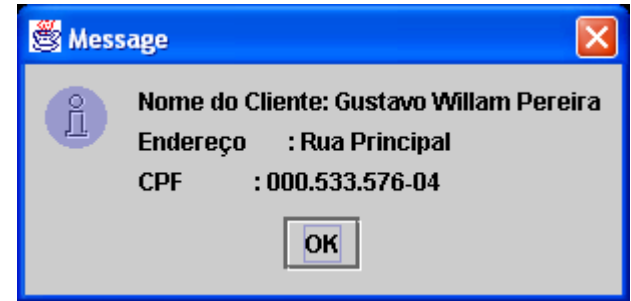
default:

```
JOptionPane.showMessageDialog(null,  
    "Opção inválida" );
```

```
} //fim do switch
```

```
} // fim do método
```

```
} // fim da classe
```



# Membros de classe static

- Cada objeto de uma classe tem sua própria cópia de todas as variáveis de instância de classe
- Em certos casos, apenas uma cópia de uma variável particular deve ser compartilhada por todos objetos de uma classe.
- Uma variável de classe static é utilizada por essa e outras razões.
- Todos os objetos compartilham a mesma parte dos dados.
- Um membro de uma classe public static pode ser acessada por uma referência a qualquer objeto dessa classe ou pode ser acessada pela nome da classe utilizando o operador ponto. (Ex: Math.random()).

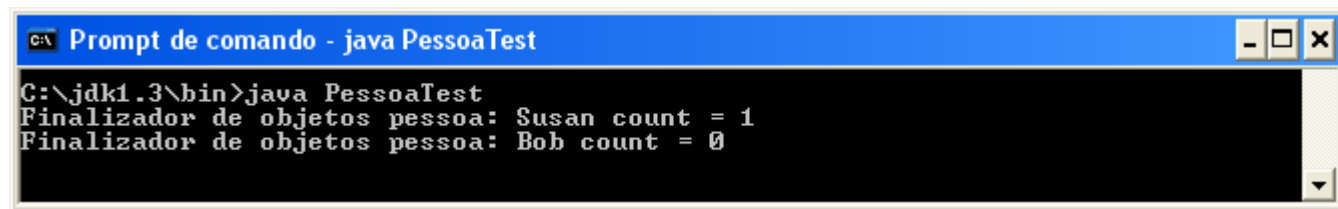
# Membros de classe static

- Um membro de uma classe private static pode ser acessada somente por métodos da classe.
- Os membros static de classe existem mesmo quando nenhum objeto dessa classe existe. Eles estão disponíveis logo que a classe é carregada na memória em tempo de execução.
- Um método declarado static não pode acessar os membros de classe não-static

# Finalizadores

- Construtores são capazes de inicializar dados em um objeto de uma classe quando a classe é criada
- Os construtores adquirem vários recursos do sistema, como a memória (quando o operador new é utilizado)
- Os recursos devem ser devolvidos ao sistema quando eles não são mais necessários para evitar desperdício.
- Java realiza coleta de lixo automática para ajudar a retornar a memória de volta para o sistema
- Cada classe em Java pode ter um método finalizador que retorna recursos para o sistema.
- O método finalizador de uma classe sempre tem o nome finalize, não recebe parâmetros e não retorna nenhum valor
- Uma classe deve ter apenas um método finalize

# Finalizadores



```
C:\jdk1.3\bin>java PessoaTest
Finalizador de objetos pessoa: Susan count = 1
Finalizador de objetos pessoa: Bob count = 0
```

// classe Pessoa

```
public class Pessoa
{
```

```
    private
```

```
        String nome; //nome da pessoa
```

```
        String endereco; // endereço da pessoa
```

```
        String cpf; // cpf da pessoa
```

```
        static int count; //contador de objetos na memória
```

```
    protected void finalize()
```

```
    {
```

```
        count = count -1; //decrementa contagem estática de pessoas
```

```
        System.out.println("Finalizador de objetos pessoa: " +
                             nome + " count = " + count);
```

```
    }
```

```
    public Pessoa( ) //construtor default da classe
```

```
    {
```

```
        setPessoa(" ", " ", " ");
```

```
        count++;
```

```
    }
```

```
    public static int getCount( )
```

```
    {
```

```
        return count;
```

```
    }
```

//métodos set

```
    public void setPessoa(String n, String e, String c)
```

```
    {
```

```
        nome = n;
```

```
        endereco = e;
```

```
        cpf = c;
```

```
    }
```

//métodos get

```
    public String getNome( )
```

```
    {
```

```
        return nome;
```

```
    }
```

} //fim da classe

# Finalizadores

```
// Testa Pessoa com variável estática em classe
import javax.swing.*;
```

```
public class PessoaTest
{
    public static void main( String args[] )
    {
        String output;

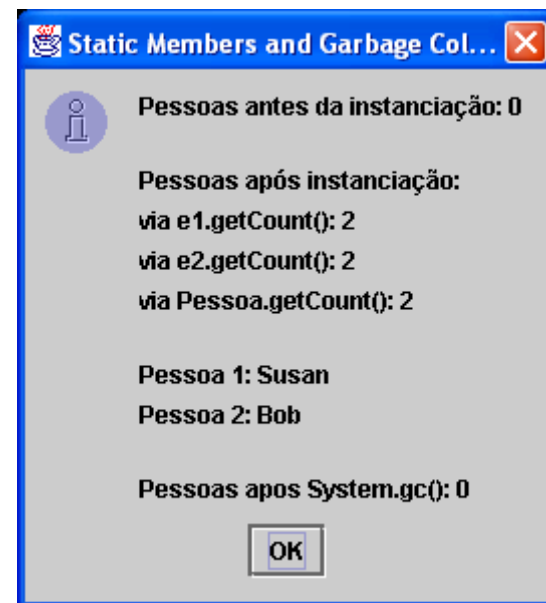
        output = "Pessoas antes da instanciação: " +
            Pessoa.getCount();

        Pessoa e1 = new Pessoa( );
        e1.setPessoa("Susan", "Rua X", "000");

        Pessoa e2 = new Pessoa( );
        e2.setPessoa("Bob", "Rua Z", "001");

        output += "\n\nPessoas após instanciação: " +
            "\nvia e1.getCount(): " + e1.getCount() +
            "\nvia e2.getCount(): " + e2.getCount() +
            "\nvia Pessoa.getCount(): " +
            Pessoa.getCount();

        output += "\n\nPessoa 1: " + e1.getNome() +
            "\nPessoa 2: " + e2.getNome();
    }
}
```



```
// mark objects referred to by e1 and e2 for garbage collection
e1 = null;
e2 = null;
```

```
System.gc(); // sugere que o garbage collector seja chamado
```

```
output += "\n\nPessoas apos System.gc(): " + Pessoa.getCount();
```

```
JOptionPane.showMessageDialog( null, output,
    "Static Members and Garbage Collection",
    JOptionPane.INFORMATION_MESSAGE );
System.exit( 0 );
```