

Programação Orientada a Objetos

Aula 06

Gustavo Willam Pereira

Introdução

- A POO encapsula os dados (atributos) e métodos (comportamentos) em objetos
- Os dados e métodos de um objeto estão intimamente amarrados entre si.
- Os objetos têm a capacidade de ocultar informações
- Embora os objetos se comuniquem através de interfaces bem definidas, os objetos não tem permissão para conhecer como os outros objetos são implementados
- Detalhes de implementação são ocultados dentro dos próprios objetos
- Em Java a unidade de programação é a classe a partir do qual objetos são instanciados (criados)
- As funções não desaparecem em Java; em vez disso, elas estão encapsuladas como métodos com os dados que elas processam

Introdução

- Programadores em C concentram-se em escrever funções – ações que realizam alguma tarefa são agrupadas para formar funções e as funções são agrupadas para formar programas
- Os dados são importantes em C, mas a idéia é que os dados existem principalmente para suportar as ações que as funções realizam.
- Os verbos em um documento de requisitos de sistema ajudam o programador em C a determinar o conjunto de funções que trabalharão juntas para implementar o sistema

Introdução

- Programadores em Java concentram-se em criar seus próprios tipos definidos pelo usuário chamados de classes.
- As classes também são referidas como tipos definidos pelo programador.
- Cada classe contém dados bem como o conjunto de métodos que manipulam os dados
- Os componentes de dados de uma classe são chamados de variáveis de instância (membros de dados em C++)
- Assim como uma instância de um tipo predefinido como `int` é chamado de variável, uma instância de um tipo definido pelo usuário (isto é, uma classe) é chamado de objeto.

Inicializando objetos de classe: construtores

- Quando um objeto é criado, seus membros podem ser inicializados por um método construtor
- Um construtor é um método com o mesmo nome da classe
- O programador fornece o construtor que é invocado automaticamente toda vez que um objeto dessa classe é instanciado
- Variáveis de instância podem ser inicializadas implicitamente com seus valores-padrão.
- Os construtores não podem especificar tipos de retorno nem valores de retorno
- Uma classe pode conter construtores sobrecarregados para fornecer uma variedade de maneiras de inicializar os objetos dessa classe.

Construtores

- Quando um objeto de uma classe é criado, inicializadores podem ser fornecidos entre parênteses a direita do nome da classe.
- Esses inicializadores são passados como argumentos para o construtor da classe.
 - `ref = new NomeDaClasse (argumentos);`
- Se nenhum construtor é definido para uma classe, o compilador cria um construtor-padrão que não aceita nenhum argumento

Utilizando os métodos get e set

- Variáveis de instância privadas podem ser manipuladas somente por métodos da classe.
- As classes frequentemente fornecem métodos public para permitir a clientes da classe configurar (set, isto é, atribuir valores a) ou obter (get, isto é obter valores de) variáveis de instância private.
- Esses métodos não precisam ser chamados set e get, mas isso ocorre frequentemente.
- Fornecer as capacidades de set e get é essencialmente o mesmo que tornar public as variáveis de instância.
- Embora esses métodos possam fornecer acesso a dados private, o acesso é restringido pela implementação dos métodos feita pelo programador.

Exemplo: Conta Bancária

// classe conta bancaria simples

```
public class Conta
{
    private
        int agencia; //agencia a qual a conta pertence
        int num_conta; // numero da conta
        double saldo; // saldo atual da conta

    public Conta( ) //construtor default da classe
    {
        setConta(0, 0, 0);
    }

    //métodos set

    public void setConta(int a, int nc, double s)
    {
        agencia = a;
        num_conta = nc;
        saldo = s;
    }
    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;
    }
}
```


Exemplo: Conta Bancária

```
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 void Deposito(double valor)
{
    saldo = saldo + valor;
}
public void Saque(double valor)
{
    saldo = saldo - valor;
}

} //fim da classe
```

Exemplo: Conta Bancária

```
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 != 7)
        {
            choice = Integer.parseInt(Instrua_Usuario());
            switch (choice)
            {
                case 1:
                    val = Double.parseDouble(JOptionPane.showInputDialog("Digite o valor a ser depositado"));
                    C1.Deposito(val);
                    break;
                case 2:
                    val = Double.parseDouble(JOptionPane.showInputDialog("Digite o valor a ser sacado"));
                    C1.Saque(val);
                    break;
                case 3:
                    val = C1.getSaldo();
                    JOptionPane.showMessageDialog(null, "Saldo de :" + val);
                    break;
```

Exemplo: Conta Bancária

case 4:

```
JOptionPane.showMessageDialog(null, "Número da Conta  : " + C1.getNum_Conta() +  
    "\nNúmero da Agência: " + C1.getAgencia() +  
    "\nSaldo da Conta  : " + C1.getSaldo());
```

```
break;
```

case 5:

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

```
C1.setAgencia(agenc);
```

```
break;
```

case 6:

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

```
C1.setNum_Conta(ct);
```

```
break;
```

case 7:

```
break;
```

default:

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

}

}

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

}

Exemplo: Conta Bancária

```
public static String Instrua_Usuario()
{
    String input;
    input = JOptionPane.showInputDialog("Tecle 1 para Depositar \n" +
        "Tecle 2 para Sacar \n" +
        "Tecle 3 para Saldo\n" +
        "Tecle 4 para Informações da conta \n" +
        "Tecle 5 para Alterar agência \n" +
        "Tecle 6 para Alterar conta \n" +
        "Tecle 7 para Sair");

    return input;

} //fim do método

} //fim da classe movimento
```

Exemplo: Conta Bancária

Input

?
Tecle 1 para Depositar
Tecle 2 para Sacar
Tecle 3 para Saldo
Tecle 4 para Informações da conta
Tecle 5 para Alterar agência
Tecle 6 para Alterar conta
Tecle 7 para Sair

OK Cancel

Opção 1

Input

? Digite o valor a ser depositado

500

OK Cancel

Opção 3

Message

i Saldo de :270.0

OK

Opção 2

Input

? Digite o valor a ser sacado

230

OK Cancel

Opção 4

Message

i Numero da Conta : 0
Numero da Agencia: 0
Saldo da Conta : 270.0

OK

Opção 5

Input

? Digite o numero da agência

02860

OK Cancel

Opção 6

Input

? Digite o numero da conta

21830

OK Cancel