Introdução a OO

Instituto Federal do Espirito Santo – Ifes
Campus Colatina – Programação Orientada a Objetos I
Professor: Giovany Frossard Teixeira

Definições

- **Orientação a objetos:** é um paradigma de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas objetos.

- **Objetos:** são abstrações dos objetos reais existentes. Todo objeto possui as seguintes características:

  - **Estado:** conjunto de propriedades de um objeto (valores dos atributos).

  - **Comportamento:** conjunto de ações possíveis sobre o objeto (métodos da classe).

  - **Unicidade:** todo objeto é único (possui um endereço de memória).

Definições

- **Classe:** é um molde, um modelo, um protótipo a partir do qual objetos são criados.

- A partir de uma classe é possível criar quantos objetos forem desejados.

Definições

- **Métodos:** são funcionalidades ou comportamentos inerentes aos objetos ou a suas classes (métodos estáticos). Ex: getNome, getQuantidade.
- **Atributos:** são os valores das propriedades de um objeto ou de sua classe (atributos estáticos). Ex: nome, quantidade.
- Nota para programadores C:
  - Podemos visualizar os métodos como funções de objetos ou classes.
  - Os atributos de uma classe se assemelham muito aos campos de uma struct.

Declaração de uma classe

- Para declararmos uma classe utilizamos a seguinte sintaxe:

```
qualificador class nome_da_classe
{
    declaração de atributos
    declaração de métodos
}
```

O qualificador determinará a visibilidade da classe, caso seja suprimido a classe será privada e com isso será visualizada apenas dentro do pacote onde foi inserida, caso seja colocado o qualificador public a classe é acessível por qualquer classe de qualquer pacote.

Visibilidade de Classes

- Classe Pública:

```
package introducao_oo;

public class Produto1 {
    double preco;
    String nome;
}
```

- Classe Privada:

```
package introducao_oo;

class teste {
    int a, b;
}
```

- Classe em outro pacote:

```
package introducao_oo_teste_qualificador;


import introducao_oo.Produto1;
//import introducao_oo.teste;


public class teste_qualificador {


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

Obs. Não é possível importar a classe **teste** no pacote **introducao_oo** pois ela é privada

Utilização de objetos

- Para utilizar um objeto precisamos executar 3 tarefas:
  - **Declarar o objeto:** assim como fazemos com tipos primitivos é necessário declarar o objeto. Ex: Veiculo v1;
  - **Instanciação do objeto:** aloca o objeto em memória, o comando new é responsável por esse trabalho. Ex: v1 = new ...
  - **Inicialização do objeto:** define valores iniciais para o objeto de forma que ele fique logicamente consistente, a operação de inicialização está intimamente ligada com métodos construtores que serão discutidos ainda nessa aula. Ex: v1 = new Veiculo(“Ford”);

Construtores

- **Construtor:** é um método chamado no momento da criação do objeto, ou seja, no momento que é utilizado o comando new. Esse método promove a inicialização do objeto, de forma que, o objeto após essa inicialização já se encontra logicamente consistente.

Ex: Veiculo v1 = new Veiculo();

Nesse momento é chamado o construtor da classe Veículo para inicializar o objeto v1

Construtores

- É possível definir diversos construtores para a mesma classe através da sobrecarga de métodos (Aula Criação de Métodos em Java), tendo os tipos ou a quantidade de parâmetros diferentes para cada um deles.

- O próximo slide apresenta a classe Cliente e a classe UsaCliente, note que no momento da criação do objeto, o construtor adequado será acionado.

Construtores

```
package introducao_oo;

public class Cliente {

    String nome;


    public Cliente()

    {

    }



    public Cliente(String nome)

    {

        this.nome = nome;

    }
```

```
public class UsaCliente {

    public static void main(String[] args) {

        Cliente c1 = new Cliente();

        Cliente c2 = new Cliente("giovany");

    }

}
```

Obs1. Note que o na criação do objeto c1 foi chamado o construtor sem parâmetros e na construção de c2 o construtor com o parâmetro String.

Obs2. A palavra **this** é utilizada para fazer referência a atributos do objeto corrente (nesse caso o objeto que está sendo construído). No exemplo estamos fazendo o atributo nome de c2 ser "giovany".

Declaração de atributos de uma classe

- Declaração de atributos:

qualificador tipo_atributo nome_atributo1, nome_atributo2;

Quando o qualificador é omitido ele é chamado *default*. Um atributo de visibilidade *default* é acessível apenas por classes do mesmo pacote.

O tipo_atributo é um tipo primitivo ou classe que define o atributo.

nome_atributo1 e nome_atributo2 são atributos de mesmo tipo e visibilidade.

Declaração de atributos de uma classe

- Atributos estáticos: assim como é possível ter métodos de classe, ou seja, métodos que não necessitam de um objeto para serem acionados, existem atributos de classe, esses atributos são chamados atributos estáticos. Sua declaração é da seguinte forma:

qualificador static tipo_atributo nome_atributo1, nome_atributo2;

Declaração de atributos de uma classe - Exemplo

```
public class Cliente {

    private int codigo;

    String nome;

    public String cidade;

    public static int cont;

    private static int cont2;

}

public class UsaCliente {

    public static void main(String[] args)

   {

        Cliente c1 = new Cliente();

        c1.cont = 0;

        Cliente.cont = 0;

        c1.nome = "giovany";

    }

}
```

Obs. Não é válido acessar diretamente codigo e cont2, dessa forma seria necessário implementar métodos públicos que façam essa tarefa.

Obs2. O atributo cont2 é estático, o método que atuará sobre cont2 deve ser estático, caso isso não ocorra deve-se verificar se existe realmente a necessidade de cont2 ser estático.

Declaração de métodos

- Já aprendemos a declarar métodos na Aula Criação de Métodos em Java, entretanto naquele contexto trabalhamos basicamente com métodos estáticos. Vejamos agora como utilizar métodos implementados.

- Para tanto utilizaremos novamente as classes Cliente e UsaCliente.

Declaração de métodos

```
public class Cliente {
    private int codigo;
    String nome;
    public String cidade;
    public static int cont = 0;
    private static int cont2 = 1;
    public int getCodigo() {
        return codigo;
    }

    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }
    public static int getCont2() {
        return cont2;
    }
    public static void setCont2(int cont2) {
        Cliente.cont2 = cont2;
    }
}
public class UsaCliente {
    public static void main(String[] args) {
        Cliente c1 = new Cliente();
        Cliente.cont = Cliente.getCont2();
        c1.setCodigo(10);
    }
}
```

Declaração de métodos

- Note que para os métodos estáticos é possível fazer referência a classe, mas que nos métodos não estáticos é necessário possuir uma instância de objeto, no nosso caso c1.

- O comportamento estático só é interessante quando é uma característica que esteja ocorrendo com o grupo de objetos instanciados de uma classe.

Dúvidas?