di
departamento de informática

dwdm
desenvolvimento para a web e dispositivos móveis

Técnicas Avançadas de Programação

Desenvolvimento para Web e Dispositivos Móveis

1º Ano, 2º Semestre

Joana Fialho
E-mail: jfialho@estgv.ipv.pt
Nuno Costa
E-mail: ncosta@estgv.ipv.pt
Carlos Simões
E-mail: csimoes@estgv.ipv.pt

Escola Superior de Tecnologia e Gestão de Viseu
2019-2020

Programação orientada a objetos

• Em POO, existem duas grandes categorias: instâncias (objetos que realizam as operações pretendidas) e as classes (definem a estrutura e o comportamento das instâncias criadas por elas). As instâncias criadas pelas classes são designadas por objetos. Para criar um objeto, usa-se a palavra reservada New e um método especial que pode ou não ter parâmetros (construtor). Os nomes dos objetos identificam-nos e representam os seus endereços.

• Para os objetos executarem os respetivos métodos, são-lhes enviadas mensagens e é esse o paradigma da POO. Genericamente, em Java, será objeto.mensagem(). Pode haver algum resultado associado (que deve ser devidamente alocado) e/ou envio de parâmetros na mensagem.

• Os métodos que não produzem resultado são modificadores do estado interno do objeto. Os que produzem resultado são interrogadores ou de consulta. Em POO, um método não deve ser, simultaneamente, modificador e de consulta, para melhor clareza de semântica de código.

Encapsulamento e modularização de código

Encapsulamento representa uma agregação, numa estrutura computacional, dados e operações que manipulam esses dados, de modo a que esses dados sejam acessíveis através de operações próprias e se possa esconder, do exterior, as estruturas de dados e a implementação das operações. Este conceito caracteriza os objetos, que são concretizações/instanciação de classes. A modularização do código tem a ver com a estrutura das classes e, consequentemente, da manipulação dos objetos

Classes e Objetos

Uma classe representa um conceito, com atributos/ campos/variáveis, construtor(es) e operações/ métodos. Os objetos são a concretização/instanciação desse conceito/classe.

Por exemplo, a classe Pessoa tem, como atributos, nome, nif, etc. Quando instanciamos/criamos objetos dessa classe, criamos variáveis do tipo Pessoa, mas cada uma delas com nome, nif, etc. diferentes. No entanto, ambas têm a mesma estrutura e acedem às mesmas funcionalidades (métodos/operações).

Atributos e visibilidade

A classe, por norma, começa com letra Maiúscula e é de acesso público. Os atributos, por norma, começam com letra minúscula e são de acesso privado (restrito à classe).

```
public class Pessoa {
    //Variáveis de instância/atributos/características
    private String nome;
    private String nif;
    private int anoNasc;
    private String contacto;
```

Métodos

Os métodos, por norma, começam com letra minúscula e são de acesso público. Como os atributos/variáveis de instância são, em geral, privados, os métodos de consulta (get) e alteração (set) dos atributos tornam-se "obrigatórios".

```
//Métodos/Operações
//Métodos de instância

public int getanoNasc() { return anoNasc; }

public String getContacto() { return contacto; }

public String getNif() { return nif; }

public String getNome() { return nome; }

public void setAnoNasc(int n) { this.anoNasc = n; }

public void setContacto(String contacto) { this.contacto = contacto;

public void setNif(String nif) { this.nif = nif; }

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

Métodos

No exemplo acima, a palavra reservada “this” serve para distinguir o atributo da variável de entrada no método. Sempre que os nomes são iguais, o atributo pode ser identificado como tal, através do “this”.

Para além destes exemplos, qualquer método pode ser definido na classe com o objetivo de executar algum conjunto de operações.

```
//Outros métodos
public int idade()
{
    Calendar cal=Calendar.getInstance();
    int year = cal.get(Calendar.YEAR);
    System.out.println(year-anoNasc);
    return  year-anoNasc;
}
```

Métodos

Os métodos são chamados associados aos objetos criados, através do operador Ponto (.)

Dentro da classe, podem existir métodos privados que poderão funcionar como auxiliares para fazer, por exemplo, cálculos intermédios. Dentro da classe, não é necessária a invocação do Ponto(.) para executar os métodos da classe.

```
Pessoa p1=new Pessoa( nome: "joana", nif: "3333", nasc: 1981, contact: "viseu");
System.out.println("Idade: "+p1.idade());
```

Construtores

Os construtores de uma classe criam instâncias (objetos), reproduzindo as variáveis e métodos de instância. Devem inicializar as variáveis de instância.

Os construtores não devolvem resultados de execução.

```
//construtor(es)
public Pessoa(String nome, String nif, Date nasc, String conc,
              String contact)
{
    this.nome=nome;
    this.nif=nif;
    dataNasc=nasc;
    concelho=conc;
    contacto=contact;
}
```

Construtores

Quando um construtor não recebe dados iniciais, o objeto é instanciado com construtor sem parâmetros (Ex. Pessoa p=new Pessoa();).

Podem existir vários construtores (sobrecarga/ overloading), desde que recebam argumentos em número e/ou tipo diferentes.

```
public Pessoa(String nif)
{
    this( nome: "",nif, nasc: 0, contact: "");
}
```

Classes sem construtor são não instanciáveis, sendo apenas prestadores de serviços (ex. classe Math)

Bibliografia da Unidade Curricular

1. **Diapositivos que cobrem a totalidade dos conteúdos lecionados**
2. **Fichas de trabalho para apoio das aulas**
3. **Diversos livros versando JAVA e outros:**

- Fundamentos Programação Java, 4ª Ed., A. J. Mendes, M. J. Marcelino, FCA Editora, 2012. COTA: 004.43JAVA MEN 4ªed
- Java 8 : POO + construções funcionais, F. M. Martins, Lisboa : FCA - Editora de Informática, 2017. COTA: 004.43JAVA MAR
- Fundamental de UML, M. Nunes, H. O'Neill, 2001, FCA Editores. COTA: [004.41 NUN]