Programação Orientada a Objetos (POO)

A **programação orientada a objetos (POO)** tem como principais objetivos reduzir a complexidade no desenvolvimento de software e aumentar a produtividade. O aumento da complexidade das aplicações informáticas implicou que o trabalho desenvolvido demorasse mais tempo e, por conseguinte, mais caro, acabando a programação estruturada por se revelar limitada e insuficiente para o desenvolvimento de projetos de software mais complexos, como é o caso das aplicações para ambientes gráficos, como o Microsoft Windows, Macintosh e Linux.
A análise, o projeto e a programação orientados a objetos são as respostas para o aumento da complexidade do software, que se caraterizam por sistemas heterogéneos baseados em interfaces gráficas.

A POO não tem a intenção de substituir a programação estruturada tradicional. Esta é uma evolução de práticas que são recomendadas na programação estruturada, mas não formalizadas. O modelo de objetos permite a criação de bibliotecas, de forma a partilhar e reutilizar o código, reduzindo o tempo de desenvolvimento e, principalmente, simplificando o processo de manutenção das aplicações.
A dificuldade que se coloca na POO é a diferença de abordagem do problema. Enquanto a programação estruturada tem como principal foco as ações (procedimentos e funções), a POO preocupa-se com os objetos e os seus relacionamentos. Além do conceito de objeto, a POO tem como fundamentos os conceitos de encapsulamento, classe, herança e polimorfismo.

NOÇÃO DE OBJETO

Um objeto é uma entidade abstrata ou existente no mundo real, sobre a qual se pretende incorporar num sistema de informação. O objeto é caraterizado por um conjunto de propriedades, um comportamento e uma identidade.

- As propriedades são as caraterísticas que definem o objeto - atributos;
- o comportamento é definido como as operações que o objeto pode efetuar sobre si próprio ou outros objetos – métodos;
- a identidade permite identificar um objeto em particular como único num conjunto de objetos idênticos - valores dos atributos

Por exemplo, considera os objetos da figura 1:

Fig. 1 - Exemplo de objetos — carros

- Apesar de serem do mesmo tipo — carros — possuem caraterísticas e funcionalidades que as distinguem.
- O objeto carro possui propriedades, como velocidade, cilindrada, altura, comprimento, cor, marca e matrícula.
- O objeto carro também possui procedimentos/comportamentos, como ligar, desligar, acelerar e parar.
- Finalmente, os valores do número de série, da marca, da cor, etc. constituem a identificação entre os demais objetos do mesmo tipo.

Objeto	Atributos	Valor do atributo	Métodos
Carro	Matrícula, marca, nº de série, modelo, cor	50-AM-60, Renault, 45327/15,Clio, verde	Acelerar, travar, buzinar, arrancar
Pessoa	Nome, idade, género, nacionalidade	Ana Vaz,43,F, Espanhola	falar, andar, pensar

NOÇÃO DE CLASSE

Uma classe define um conjunto de caraterísticas que são comuns a uma série de objetos com particularidades semelhantes. Para que um objeto seja de determinada classe, terá, obrigatoriamente, de respeitar a especificação da classe – tornando-se numa **instância da classe.**

Uma classe é como que um molde a partir do qual se criam objetos de um determinado tipo.

Cada instância de uma classe terá um valor diferente para cada um dos atributos definidos na especificação da classe.

Classes e objetos podem dizer respeito a qualquer tipo de entidades usadas em programação, tais como: janelas, menus, botões de comando, estruturas de dados, caixas de texto, imagens, etc.

- Os objetos, em programação, são unidades de código utilizadas no desenvolvimento de aplicações.
- Os atributos dos objetos são guardados em variáveis internas ao próprio objeto.
- Os valores dos atributos de um objeto podem ser colocados na altura da criação do objeto, sob forma de variáveis deste, e variar durante a execução do programa;
- Um método é semelhante a uma função – uma entidade lógica que aceita determinados parâmetros de entrada e que realiza uma determinada ação, podendo não devolver resposta) mas que é definido no interior de uma classe destinando-se a operar entre os objetos dessa classe.

Exemplos:

Código correspondente à classe pessoa que define os atributos dos objetos da classe Pessoa

```
class Pessoa {
//atributos
String nome, nacionalidade;
int idade;
char genero;
}
```

Código correspondente à criação de um objeto (instância da classe)

Sintaxe:

```
Nome_da_classe  objeto = new Nome_da_classe (argumento);
```

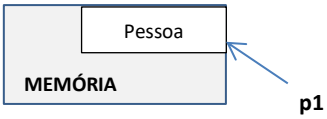
Os parêntesis à frente do nome da classe são obrigatórios. Em muitas classes, os objetos são criados sem passagem de argumentos.

Na realidade na sintaxe anterior a palavra *objeto* designa uma variável que faz referência a um objeto, isto é refere-se a um objeto. Uma variável declarada a partir da definição de uma classe não representa a informação propriamente dita, mas sim uma referência à informação.

Código correspondente à criação de um objeto da classe pessoa

```
Pessoa p1=new Pessoa();
```

Nota: Internamente, p1 vai guardar um número que identifica em que posição da memória aquela Pessoa se encontra. É parecido com um ponteiro, porém não se pode manipular como um número e nem utilizá-lo para aritmética.



Código correspondente à classe pessoa que define os métodos dos objetos da classe Pessoa

Os métodos são declarados dentro das classes que lhes dizem respeito, isto é, cujos objetos irão utilizar.

```
class Pessoa {
//Métodos
void falar(){
 System.out.println("Blá…Blá…Bla…");
 }
}
```

Método que possui retorno

```
int somar( int a, int b ) {
return a + b;
}
```

Este método retornou a soma de a+b.

Método que não possui retorno

```
void imprimirNaTela( String nome ) {

      System.out.println( "O meu nome é " + nome );
      }
```

Este método simplesmente imprimiu uma mensagem no ecrã sem retornar qualquer valor.

Notas:

1. String é uma classe do pacote java.lang que é importada automaticamente para o programa, tal como todas as classes deste pacote*.

2. Habitualmente a identificação da classe é iniciada com maiúscula, para serem distinguidos dos objetos que começam com minúscula.

3. Cada classe deve ficar num ficheiro individual, cujo nome deverá obedecer à mais simples regra de nomenclatura. O nome do ficheiro deverá ser igual ao nome da classe e sua extensão deve ser obrigatoriamente.java.

4. Uma classe pode ser implementada, tornando-se um objeto com várias caraterísticas que pode ser facilmente manipulado e incorporado por outros objetos. Isso é o que chamamos de reutilização de código.

Acesso a atributos e métodos

O acesso a atributos ou métodos de uma classe são permitidos, a partir do objeto instanciado, usando um separador que é o ponto (.)

Para aceder a qualquer membro da classe, basta que usemos o nome que faz referência ao objeto, mais um ponto e o nome do membro respetivo.

Exemplo:

```
class Pessoa {
//atributos
      String nome, nacionalidade;
      int idade;
      char genero;
}

Public static void main(String args[]){

      Pessoa p1=new Pessoa();

      p1.nome=”Ana”;
      p1.nacionalidade=”Brasileira”;
      p1.idade=29;
      p1.genero=’F’;

System.out.println(“Nome:“+p1.nome+”\nNacionalidade:“+p1.nacionalidade+”\nIdade: “+p1.idade+”\nGénero: “+p1.genero);

falar(); // execução do método falar

}
}
```

HERANÇA

No mundo real as pessoas herdam as caraterísticas dos seus pais que, por sua vez, as herdaram dos seus. Em POO é possível definir uma hierarquia de classe em que cada classe "herda" as caraterísticas das suas superiores

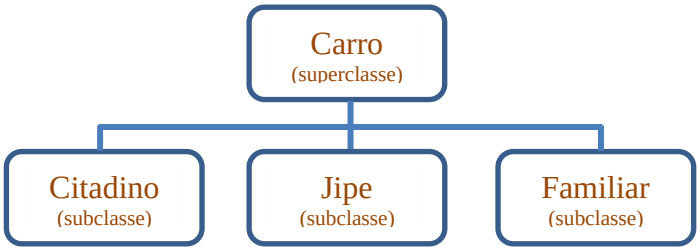


Fig. 2 – Estrutura hierárquica com a classe "carros" como superclasse

Na estrutura hierárquica da figura 2, a superclasse "Carro" possui os atributos e métodos comuns a todos os objetos Carro. Cada uma das subclasses possui os atributos e métodos específicos de cada uma delas, herdando os atributos e métodos da superclasse.

Se uma classe for declarada como uma subclasse de outra classe a sintaxe será a seguinte:

```
class nome_da_ classe extends Nome_da_superclasse {
…
/*código*/A
…
}
```

***** Pacotes **(packtage)** – conjunto de classes e interfaces** que podem ser importados e utilizados em programas Java.

** Conjunto de métodos usado numa classe, mas que não lhe pertencem, nem é acessível por herança de alguma superclasse. Duas classes completamente diferentes podem responder à chamada de um determinado método, cujo nome é reconhecido de formas idêntica.

As classes:

- Em Java os programas são constituídos por diversas classes
- Algumas classes são escritas por nós, outras fazem parte da biblioteca
- As classes são agrupadas em pacotes e possuem campos e métodos
- As classes são tipos
- Em Java cada objeto pertence a um determinado tipo
- O tipo de um objeto é a sua classe
- Algumas classes são *Applets* e podem ser executadas num *browser*

FUNÇÕES

Sempre que pretendemos usar a mesma codificação para algo específico, criamos uma função. Dessa forma, sempre que quisermos utilizar aquele código, não precisamos de criar outro igual novamente, simplesmente chamamos a função. Funções são extremamente úteis e adaptáveis, e o conceito de funções é importante para entendermos o funcionamento e criação dos métodos.
Para criar uma função, temos que ter sempre em mente que toda função é global, ou seja, é estática (static).

Exemplos:

1.

```
public class ExemploFuncao {
    //criar a função
       public static void mostrarMensagem() {
            System.out.println("Minha
            Mensagem");
       }
       public static void main(String[] args) {
            //chamar a função dentro do programa principal
            mostrarMensagem();
       }
  }
```

2.

```
public class FatorialComFuncao {
     public static void fatorar(int numero) {
          int fator = 1;
          for (int i = numero; i > 1; i--)
              fator *= i;
      System.out.println(numero + "! = " + fator);
 }
      public static void main(String args[]) {
          for (int x=1; x<=10; x++)
                fatorar (x);
      }
 }
```

3.

```
    public class Primo {
           public static boolean ehPrimo(long nr) {
            if (nr < 2)
                 return false;
            for (long i = 2; i < nr i++)
                 if (nr % i == 0)
                       return false;
            return true;
    }
 public static void main(String[] args) {
     long x = 5;
     if (ehPrimo(x)) // se for primo
       System.out.println(x + " é primo");
       else // se não for primo
          System.out.println(x + " não é primo");
  }
 }
```

Exercício: Cria uma classe chamada Conta que possua os atributos numero, cliente, saldo e limite. Cria uma instância da classe em que os atributos tenham os seguintes valores: 1, José Beto, 1000, 5000. Imprime no ecrã o valor atual do saldo relativo ao cliente.

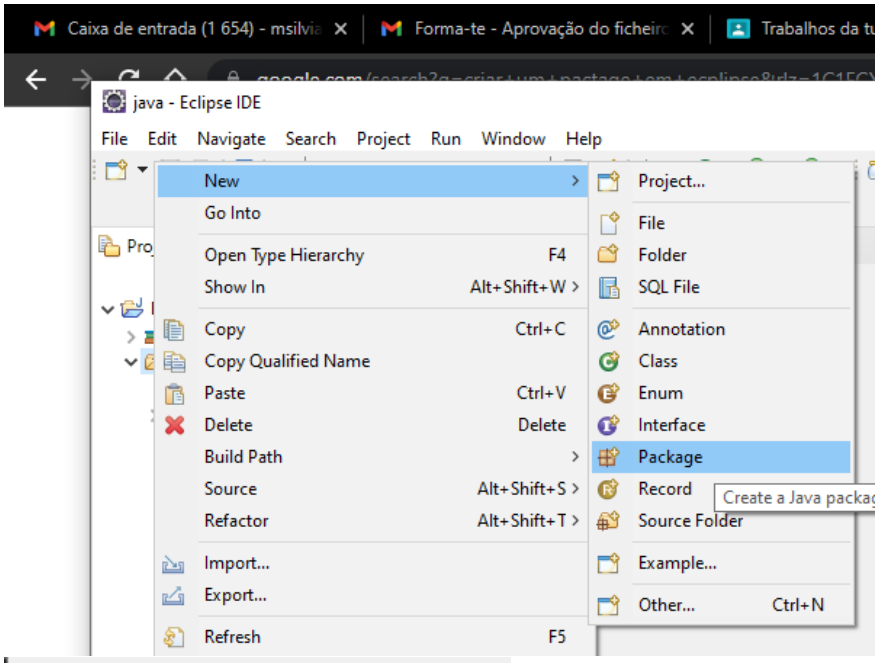
Bibliografia:

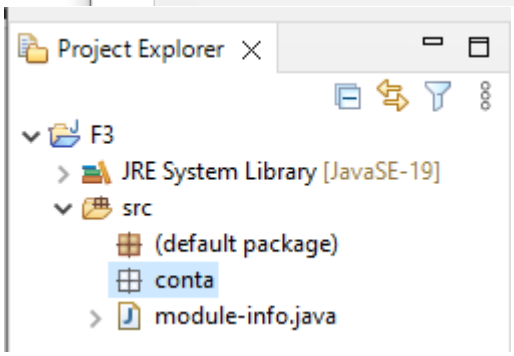
Objetos e classes em Java - Javatpoint

https://www.w3schools.com/java/java_oop.asp

Jesus, C. (2013). Curso Prático de Java.Lisboa:FCA

Coelho, P (2016). Programação em JAVA – Curso Completo. Lisboa: FCA


Caixa de entrada (1 654) - msilvia
Forma-te - Aprovação do ficheiro
Trabalhos da tu
java - Eclipse IDE
File Edit Navigate Search Project Run Window Help
New
Go Into
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Project...
File
Folder
SQL File
Annotation
Class
Enum
Interface
Package
Record
Source Folder
Example...
Other... Ctrl+N
Create a Java packag


Project Explorer
F3
JRE System Library [JavaSE-19]
src
(default package)
conta
module-info.java