Mecanismos de controlo de acesso

- Especificam "quem" tem acesso a cada entidade, isto é, quem tem acesso a cada classe e cada membro da classe (dados e métodos)

Modificadores de acesso:

public

protected

private

"por omissão"

Regras de acesso a classes:

R1: Uma classe é sempre acessível a todas as outras classes do mesmo package (qualquer que seja o modificador de acesso).

R2: Se nenhum modificador de acesso é usado, a classe apenas pode ser acedida dentro do seu package.

R3: Quando uma classe é declarada como "public" pode ser acedida por qualquer classe que tenha acesso ao seu package.

R4: Quando uma classe é não pública apenas é acessível dentro do seu package.

Regras de acesso a variáveis e métodos:

"norma:" variáveis são privadas,

métodos de interface são públicos,

métodos auxiliares são privados

.

R1: Um método declarado como "public" é acessível de qualquer ponto de qualquer programa.

Designa-se por API ("Application Programming Interface") de uma classe, o conjunto de métodos de instância que não forem declarados como "private".

R2: Um método sem modificador de acesso é acessível a qualquer classe do mesmo package.

R3: Métodos ou variáveis declarados como "private" são apenas acessíveis dentro da própria classe.

R4: Métodos ou variáveis declarados como "protected" são acessíveis na própria classe, de outra classe dentro do mesmo package e nas subclasses da classe (!).

Definição da classe Contador

```
public class Contador  {
    // variáveis de instância
    private int conta;
    // construtores
    public Contador () {
            conta = 0;
     }
    public Contador ( int conta) {
            this.conta = conta;
     }
    // métodos de instância
    public int getConta(){
        return conta;
    }
    public void incConta () {
        conta ++;
    }
    public void incConta (int inc) {
        conta = conta + inc;
    }
    public void decConta () {
        conta --;
    }
    public void decConta (int dec) {
        conta = conta - dec;
    }
    public String toString () {
        return ("Contador: " + conta );
    }
}
```

Classes de teste

- Uma classe de teste serve para testar toda a funcionalidade de uma classe

```
public class TesteContador {

   public static void main ( String[] args) {

      // criar instâncias da classe
      Contador c1, c2;
      c1 = new Contador();
      c2= new Contador (10);

      // enviar mensagens às instâncias criadas
      // obter o valor do contador
      int i1, i2;
      i1 = c1.getConta();
      i2 = c2.getConta();
      // verificar os resultados
      System.out.println ("c1 = " + i1 );
      System.out.println ("c2 = " + i2 );
      // alterar valores
      c1.incConta();
      c2.incConta(10);
      System.out.println ("c1= " + c1.getConta() +
                          "\n" + "c2 = " + c2.getConta() );
      c1.decConta();
      c2.decConta (2);
```

A execução de qualquer programa em Java inicia-se por um método com esta assinatura.

```
        // converter para String
                String s = c1.toString();
                 System.out.println (s);
                 System.out.println ( c2.toString() );

          System.out.println ( c2 );  (??)

    } // main
} // class TesteContador
```

Exercício: Qual o output deste programa?

Verifique a sua solução implementando o programa.

c1 = 0
c2 = 10

c1= 1
c2 = 20

Contador: 0
Contador: 18
Contador: 18

(??) Numa instrução de escrita, o método toString definido pelo utilizador será usado para converter o objecto para texto mesmo quando não é explicitamente invocado.

Variáveis e Métodos de Classe

Em Java, quer as classes quer as instâncias das classes são objectos.

Onde está o estado da classe?

Com que operações é manipulado?

Variáveis de classe - representam a estrutura interna de uma dada classe

Métodos de classe - métodos que implementam o comportamento da classe.

- São invocados através de mensagens enviadas à classe.

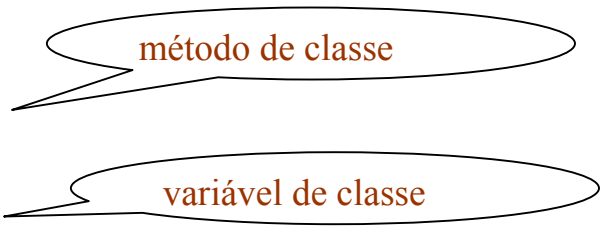
Como se declaram?

- com o identificador static

Ex.lo

```
public static int  metodoA()

private static String texto;
```



Para que servem?

Usam-se variáveis de classe para armazenar valores que digam "respeito" a todos os objectos da classe.

Exemplo:

- se quiséssemos saber, em determinado instante, quantos objetos do tipo Contador já tinham sido instanciados.

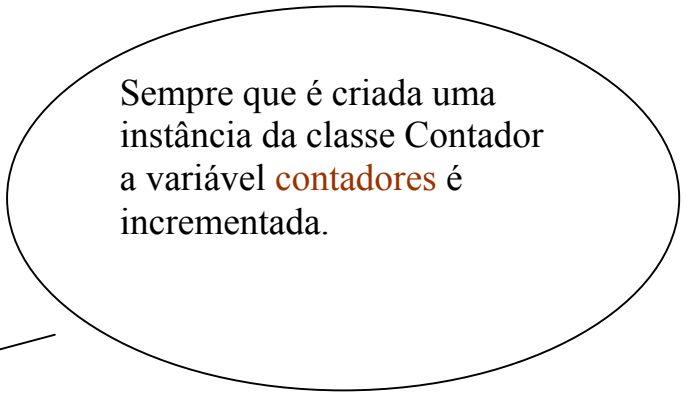
```
public class Contador {
    // declarar uma variável de classe que vai conter o nº de objetos instanciados
    private static int contadores = 0;

    // método de classe
    public static int getContadores () {
        return (contadores) ;
    }
   // variável de instância
   private int conta;

   // reescrever os construtores
   public Contador () {
      conta = 0;
      contadores ++;
   }

   public Contador ( int conta) {
      this.conta = conta;
      contadores ++;

   }
    …
```



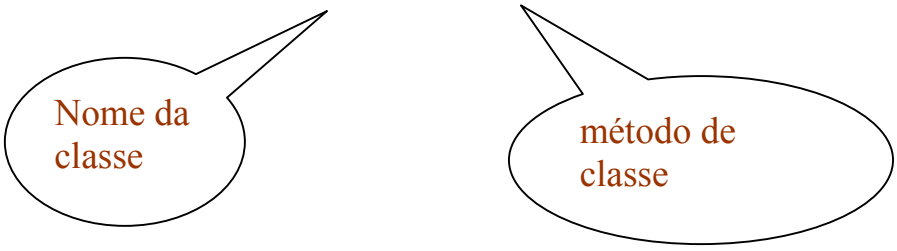
Na classe TesteContador:

```
public class TesteContador {

   public static void main ( String[] args) {
          System.out.println ("Nº de objetos do tipo Contador" );
          System.out.println (Contador.getContadores());

          Contador c1 = new Contador();
          Contador c2 = new Contador(10);

          System.out.println ("Nº de objetos do tipo Contador" );
          System.out.println (Contador.getContadores());
   }
```



- Variáveis de classe podem ser usadas mesmo que nunca tenha sido instanciado um objeto da classe.

- Métodos de classe são acessíveis às instâncias da classe, isto é um método de instância pode invocar um método de classe.

- Métodos de classe não podem invocar métodos de instância.

Questões:

- Quantas e quais as variáveis que existem ao longo da execução do programa anterior?

- Qual o output do programa anterior?

- Em que situações anteriores já usou valores ou métodos de classe?

```
…
public static void main(String[] args);
System.out ;
System.in ;
Calendar.HOUR_OF_DAY;
Math.random();
…
```

Classes não instanciáveis:

São classes só com variáveis e métodos de classe
(diferente de classes com apenas uma instância)

- Não especificam a estrutura nem o comportamento de qualquer instância.
- Representam "centros de serviços" que não faz sentido replicar

Ex.lo

Classe Math:

Math.mensagem();

```
double x, y, z;
…
double x = Math,sqrt( y );
double y = Math.cos (x );
double z = Math.random();
        …
```

___

Quando definimos uma classe, alguns atributos dessa classe podem ser objetos de uma qualquer classe já definida.

Trata-se de uma forma de Composição de Classes

➔ aula teórico-prática