

# Top\_Down Design - Stepwise Refinement

- Uma metodologia de resolução de problemas muito usada na programação estruturada baseia-se na decomposição descendente do problema nas suas funcionalidades (**Top-Down** – do global para o particular) com a sucessiva decomposição de cada uma dessas funcionalidades, até o procedimento estar completamente detalhado (**Stepwise Refinement**).
- Começa-se por descrever o procedimento de resolução (programa) em termos das suas funcionalidades gerais. Cada uma dessas funcionalidades é sucessivamente decomposta em funcionalidades mais simples até estar todo o programa detalhado.

# Decomposição “Top-Down” - Síntese

**O método de decomposição tem as seguintes fases:**

1. Decompor o problema em módulos integrando-os numa estrutura de controlo

**(Descrever o respetivo algoritmo de integração dos módulos)**

2. Para cada módulo

Se (o módulo é simples)  
então

Programar o módulo

senão

Decompor o módulo como referido em 1.

# Noção de Módulo

- Um módulo é um bloco de instruções que executa uma tarefa específica.
- De um modo geral, cada **módulo**
  - recebe **dados**,
  - processa-os e
  - devolve **resultados** retornando **explicitamente valores ou não**
  - Pode haver **módulos** que não recebem explicitamente **dados de entrada** e / ou
  - não devolvem **resultado** como um valor de retorno
- A codificação de um módulo não deve conter muitas linhas de código

# Tipos de Módulos

- Existem dois tipos de **módulos**:
  - Função** – módulo que executa um bloco de instruções e **retorna um valor**, para a linha de código que o chamou, através da instrução **return** valor
  - Procedimento** – módulo que executa um bloco de instruções **não retornando** qualquer valor
- Na programação OO os **módulos** designam-se por métodos.
  - Há métodos que **retornam** valor  
EX:  

```
public long factorial ( long numero ) { ... }
```

← **long** retorna valor
  - Há métodos que **não retornam** valor  
EX:  

```
public void mostrar ( int valor ) { ... }
```

← **void** não retorna valor

Parâmetros  
de entrada

# Métodos em Java

## Definição de método

public  
**private**  
Protected  
...

Assinatura do método

modificadores **tipo\_retorno** nome (**list\_params**)

{

variáveis locais

instruções

}

Os parâmetros são os meios de  
comunicação com o método

Os parâmetros funcionam como variáveis locais inicializadas com os valores dos argumentos de chamada


# Invocar métodos - Correspondência Argumentos - Parâmetros

- Os métodos podem ser chamados permitindo que o programa execute a funcionalidade do método sempre que necessário

```
double a=12.8, b=7, c=3.5;
```

```
double res= maior3( a, b, c);
```

```
private static double maior3(double x, double y, double z)
{
    return (Math.max(Math.max(x,y),z)
}
```



# Passagem de Parâmetros para os métodos

Exemplos:

**Para chamar métodos estáticos de uma outra classe**

```
String res= JOptionPane.showInputDialog (null,"Qual o número");
```

Nome da Classe      Nome do Método      Lista de argumentos

```
int n = Integer.parseInt("124");
```

**Na chamada de métodos estáticos da própria classe podemos omitir o nome da classe**

Cabeçalho do método      private static float area\_r(float l1,float l2){ }

Chamada do método      float res= area\_r(lado1, lado2);