APOYA

Introducción a la Programación

Comisión “B”

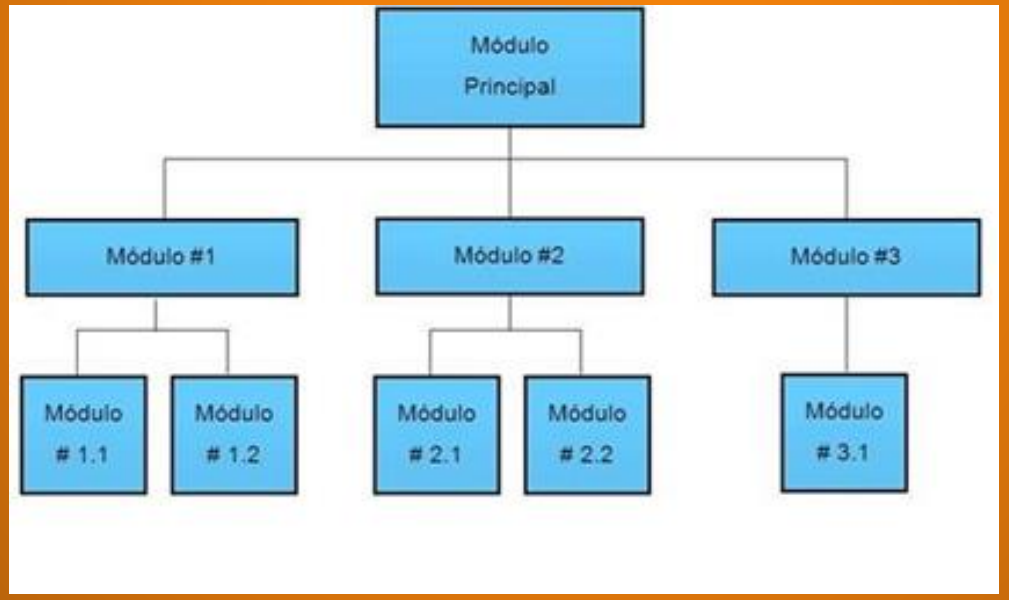
Año 2020

Funciones y Procedimientos

Profesor: Ing. Gabriel Guismin

PROGRAMACIÓN MODULAR

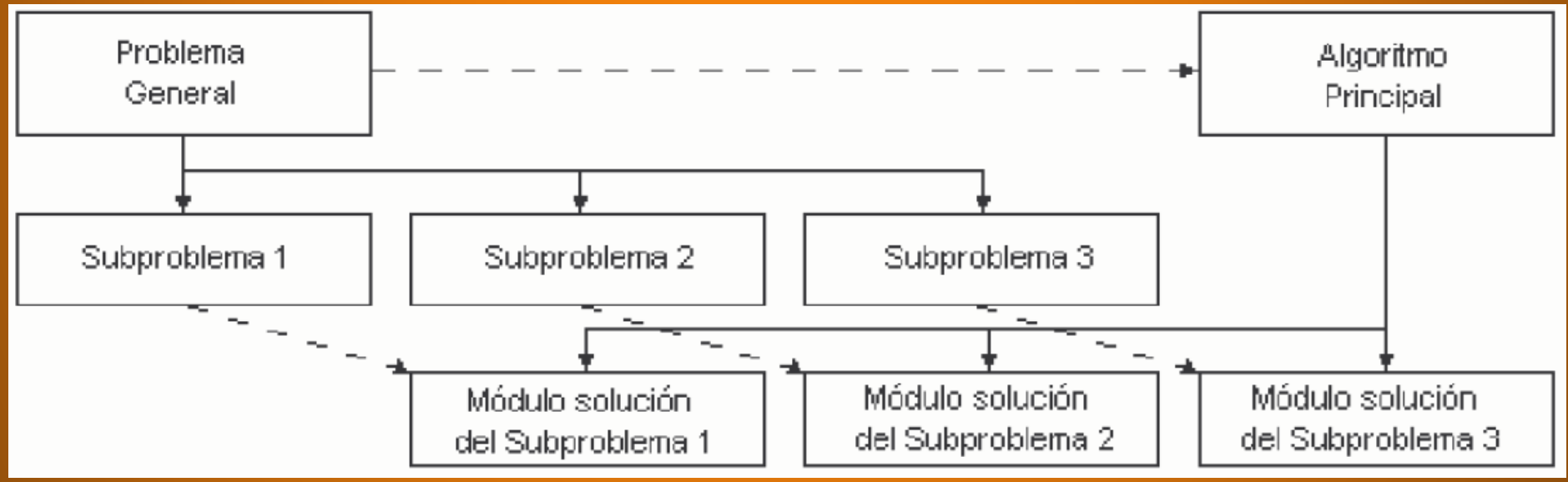
La programación modular permite plantear este concepto a través de la división del problema principal en varios problemas menores



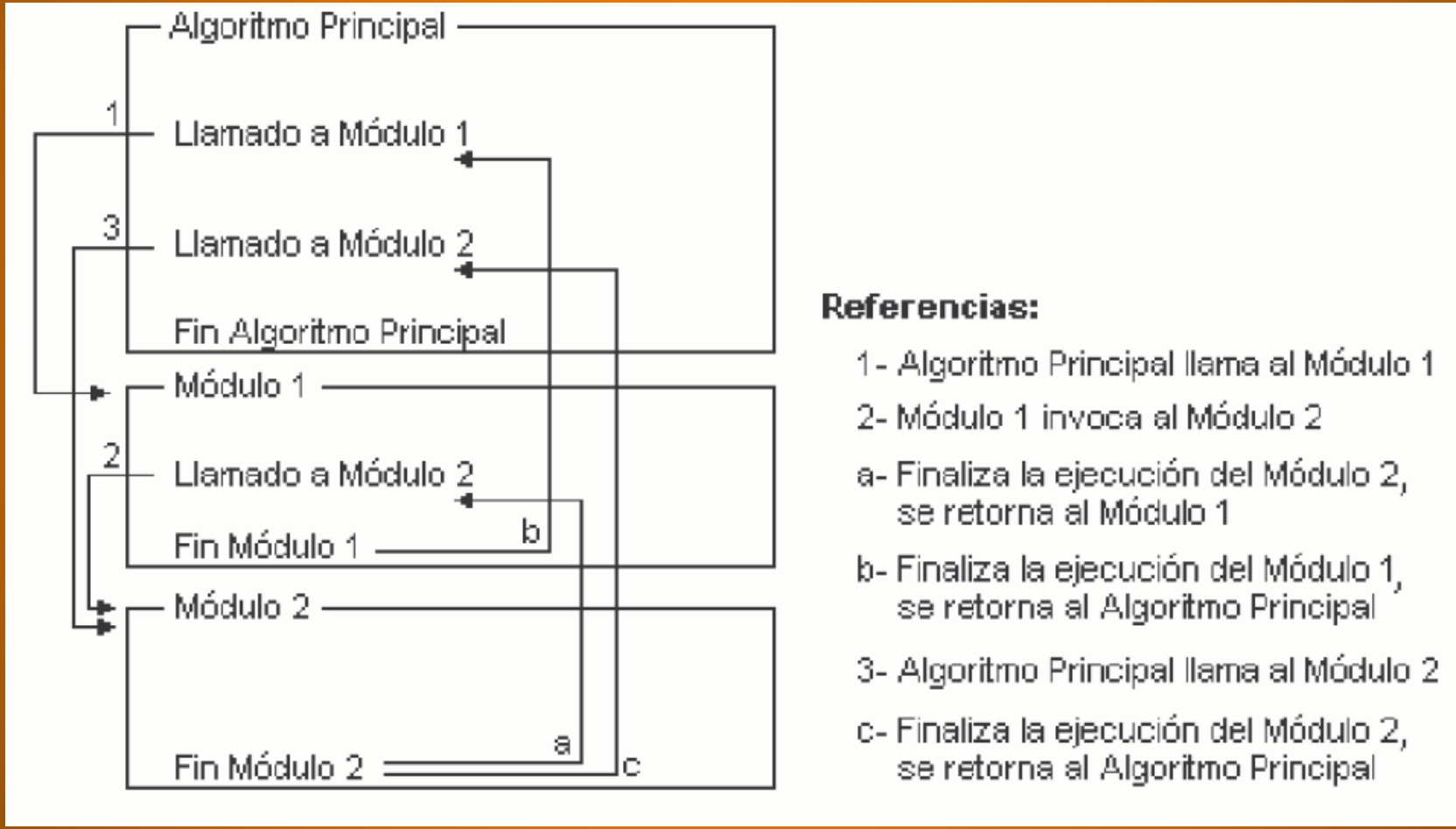
Ing. Gabriel Guismin

DISEÑO DESCENDENTE

El diseño descendente consiste en dividir un problema complejo en varios problemas menores, más sencillos de resolver.



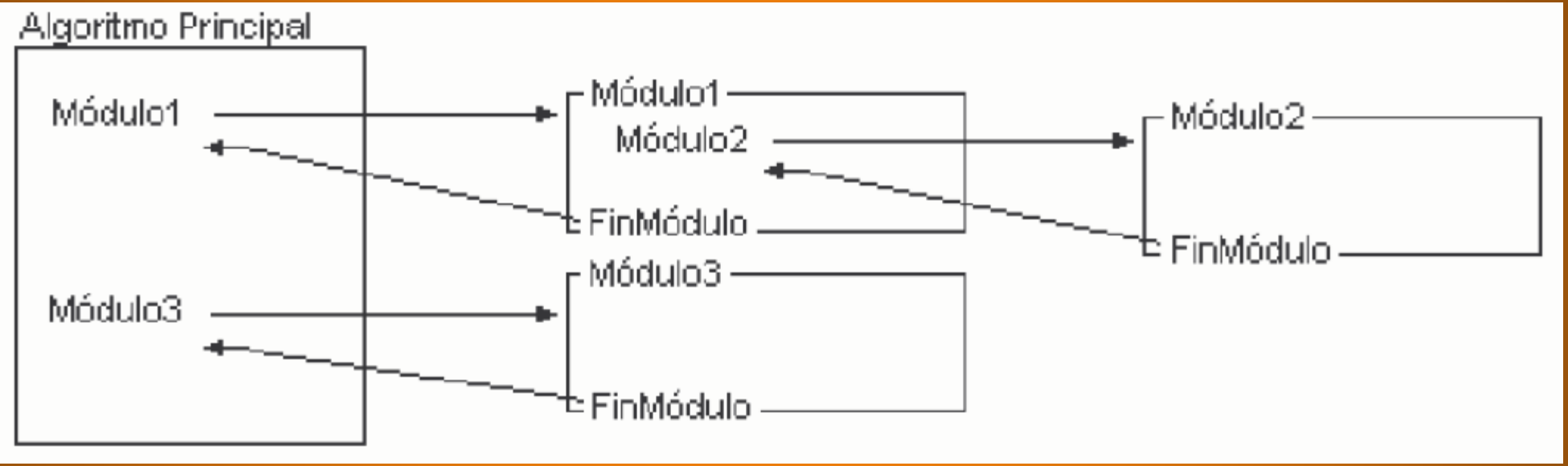
Módulos y llamados (invocaciones)



Ejemplo 1: Secuencia de llamados a módulos y sus retornos.

Ing. Gabriel Guismin

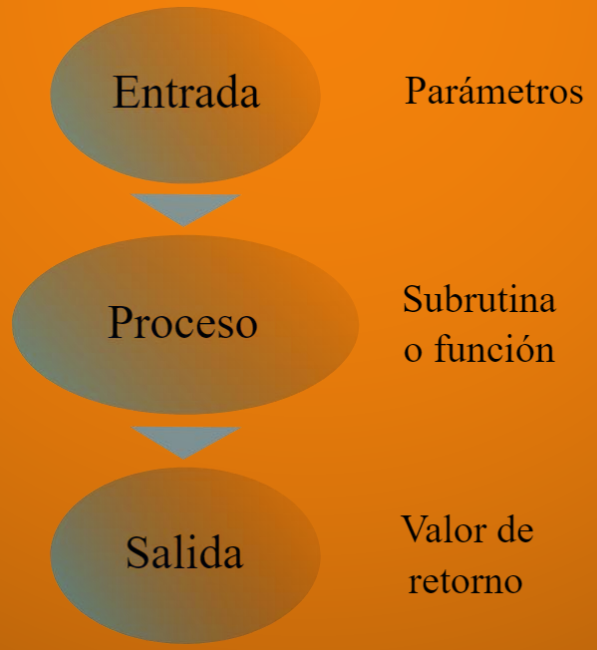
Módulos y llamados (invocaciones)



Ejemplo 2: Secuencia de llamados a módulos y sus retornos.

¿Cuándo utilizar módulos?

Estos se utilizan no solo para simplificar la codificación a través del diseño descendente, sino también cuando la solución del problema requiere realizar varias veces, en distintos puntos del algoritmo, las mismas acciones.
En estos casos, las acciones son candidatas evidentes a estar escritas en un módulo y éste será invocado desde los distintos puntos del algoritmo principal.



Algunas ventajas

- Permite reducir complejidad del programa, siguiendo el lema: “Divide y vencerás”.
- Se elimina código duplicado
- Mejorar y facilitar la legibilidad del código
- Disminuir el impacto de los cambios
- Reutilización de código y factorización
- Facilita el mantenimiento

Funciones

Una función es un conjunto de acciones (módulo) que realizan una tarea específica y como resultado siempre devuelve un valor.

```
<tipo de datos> funcion nombre([declaración de parámetros])
    [declaración de variables locales]

inicio
    <acción1>                                  { Cuerpo de la Función }
    <acción2>
        ...
    <acciónn>
    retornar(valor)                            { Retorno de resultados }
finfuncion
```

Funciones

```
Funcion variable_de_retorno <- Nombre ( Argumentos )

    //Instrucciones de la Funcion

Fin Funcion
```

Función en PSeInt

Funciones - Ejemplo

```
Funcion resultado <- sumando ( num1, num2 )

    resultado = num1 + num2

    //Retornar resultado
    //Devolver resultado

Fin Funcion


//-----------------------------------------------------

Algoritmo sumandonumeros

    Definir num1,num2,resultado como Real

    Escribir "Ingresar 2 numeros para sumar"
    Leer num1, num2

    //Se invoca a la Funcion sumando
    resultado <- sumando ( num1, num2 )

    Escribir "El resultado de la suma es: " resultado



FinAlgoritmo
```

Ejemplo de función suma

Ing. Gabriel Guismin

Procedimientos

Un procedimiento es un conjunto de acciones (módulo) que realizan una tarea específica y no devuelve valores.

```
procedimiento nombre([declaración de parámetros])
    [declaración de variables locales]

inicio
    <acción1>                                  { Cuerpo del procedimiento }
    <acción2>
        ...
    <acciónn>
finprocedimiento
```

Procedimientos

```
SubProceso procedimiento ()

    //Instrucciones del procedimiento


FinSubProceso
```

Procedimiento en PSeInt

Procedimientos - Ejemplo

```
Subproceso sumando (n1, n2)

    resultado = n1 + n2

    Escribir "El resultado de la suma es: " resultado

Fin Funcion


//-----------------------------------------------------

Algoritmo sumandonumeros

    Definir num1,num2,resultado como Real

    Escribir "Ingresar 2 numeros para sumar"
    Leer num1, num2

    sumando(num1, num2)



FinAlgoritmo
```

Ejemplo de procedimiento suma

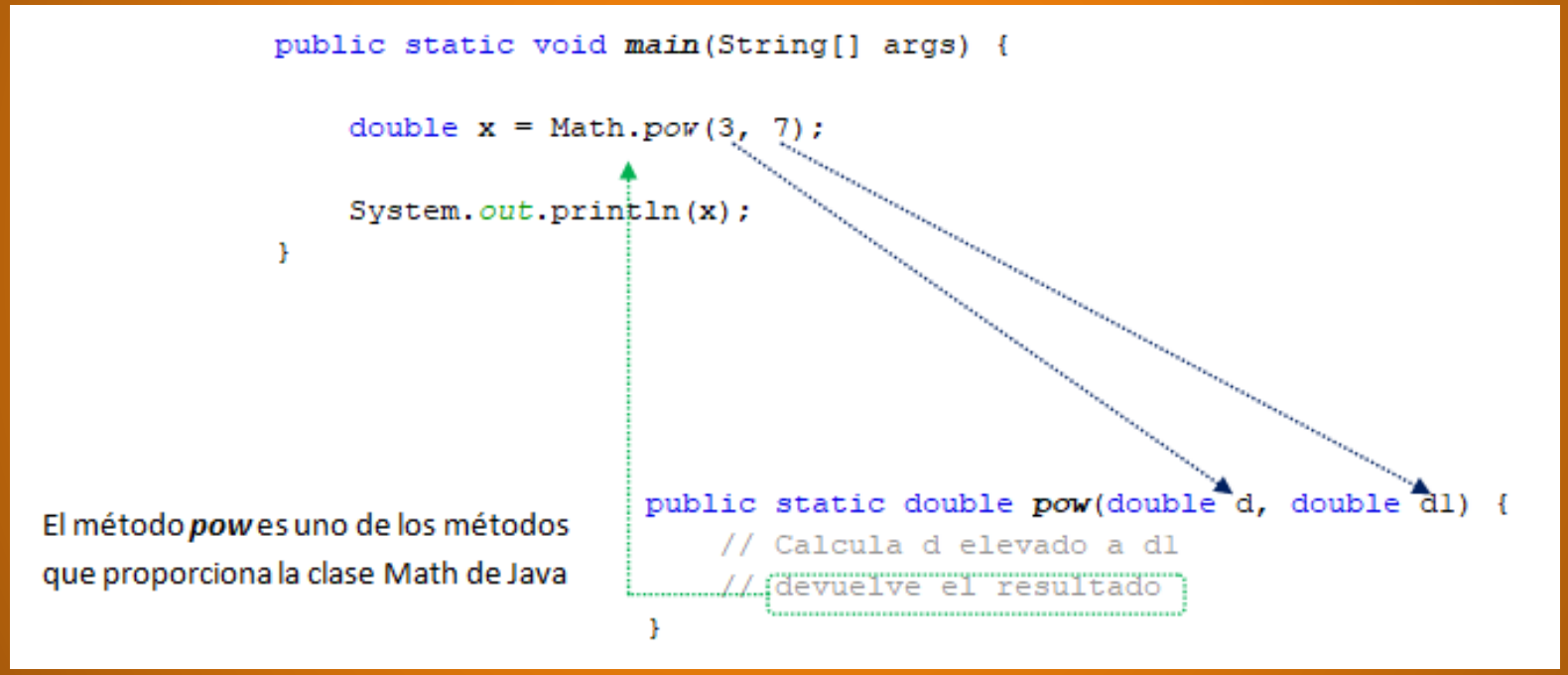
Ing. Gabriel Guismin

Ámbito de las variables

- VARIABLES GLOBALES: Una variable global es aquella que está declarada para el programa o algoritmo principal, del que dependen todos los subprogramas

- VARIABLES LOCALES: Una variable local es aquella que está declarada y definida dentro de un subprograma, en el sentido de que está dentro de ese subprograma

Para conocer: en JAVA



Ejemplo de llamado a un **método** (función/procedimiento) en Java