



Programación 1

Diseño descendente

Diseño descendente

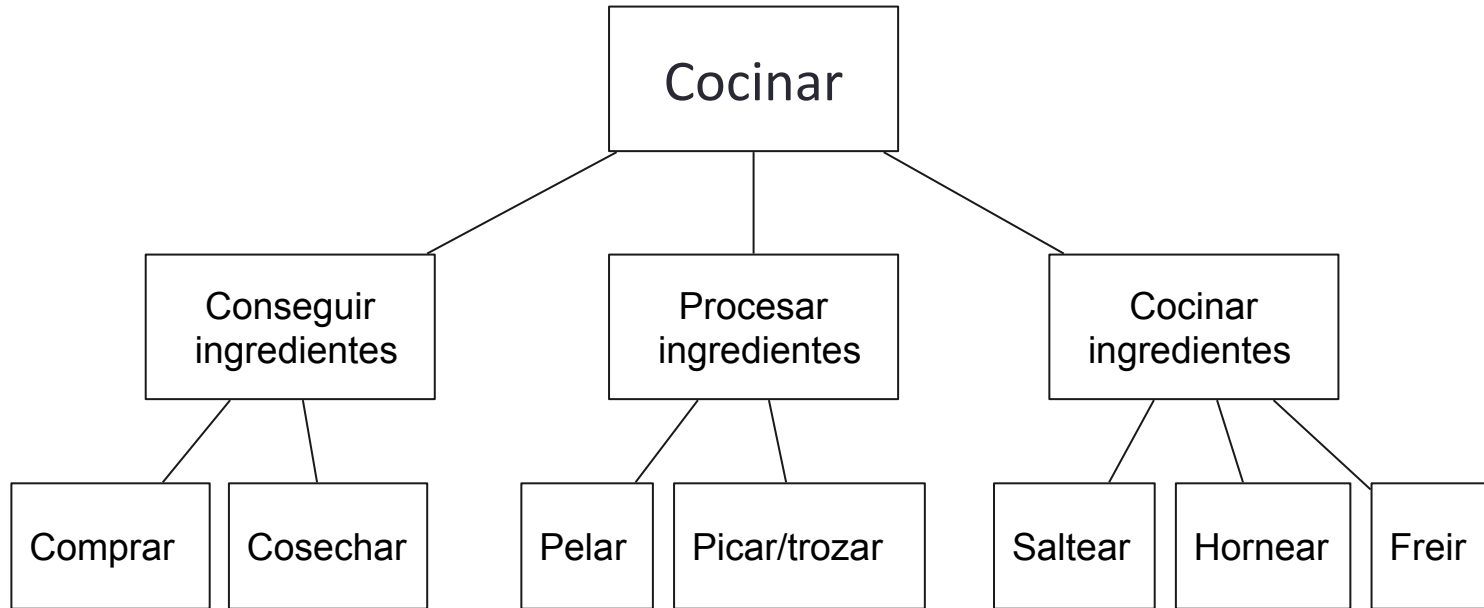
- Uno de los métodos fundamentales para resolver un problema es dividirlo en problemas más chicos o subproblemas.
- Está división se hace repetidamente hasta llegar a pequeños problemas fácilmente solucionables o ya solucionados.
- Es deseable que cada subproblema sea independiente de los restantes.



Ventajas de dividir el problema en subproblemas

- La implementación de un subproblema se puede ejecutar más de una vez **ahorrando tiempo** de programación y **cantidad de código**.
- La división de un problema en subproblemas facilita la **división** de tareas dentro de un grupo de trabajo.
- La resolución de cada subproblema se puede **comprobar individualmente**.
- La división de problemas en subproblemas hace al programa más **legible y modificable**

Diseño descendente



Pseudocódigo

La forma más sencilla de describir un problema con subproblemas es anotando los pasos a realizar con oraciones simples en pseudocódigo.

Escribir un diseño de programa en el que dado un número ingresado por el usuario, se chequee si es natural y en ese caso se imprima la tabla de multiplicar del 5.

```
public class Clase_3_Ejemplo_1 {  
    public static void main(String args[]) {  
        obtener un número por teclado  
        chequear si es natural  
        imprimir tabla de multiplicar del numero 5  
    }  
}
```

Pseudocódigo

```
try {
```

```
    BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
```

```
    System.out.println("Ingrese nro entero: ");
```

```
    entero = Integer.valueOf(entrada.readLine());
```

```
}
```

```
catch (Exception exc) {
```

```
    System.out.println("error");
```

```
}
```

```
public class Clase_3_Ejemplo_1 {
```

```
    public static void main(String args[]) {
```

```
        obtener un número por teclado
```

```
        chequear si es natural
```

```
        imprimir tabla de multiplicar del numero 5
```

```
    }
```

```
}
```

```
        if (entero>0){
```

```
            System.out.println("Tabla de multiplicar del 5");
```

```
            for (int i = 1 ; i <= 10; i++) {
```

```
                System.out.println(5*i);
```

```
            }
```

```
        }
```

Pseudocódigo



```
/*Escribir un diseño de un programa que mientras el usuario ingrese  
un número entero mayor que 0, imprima la tabla de multiplicar del  
10. Cuando salga del ciclo vuelva a imprimir la tabla de  
multiplicar del 10.
```

```
*/  
public class Clase_3_Ejemplo_2 {  
    public static void main(String[] args){  
        definir número entero  
        obtener un número entero por teclado  
        mientras numero entero sea mayor que 0  
            imprimir tabla de multiplicar de 10  
            obtener un número entero por teclado  
            imprimir tabla de multiplicar de 10  
    }  
}
```

¿Qué es un método?



Un método es una **porción** de **código** que lo puede llamar el main o cualquier otro método. Ej: calcular(); sumar(a,b);mostrar();

Se invoca por su nombre seguido de paréntesis con o sin parámetros. Estos parámetros son usados por el método para sus cálculos. Ej: sumar(a,b); mostrar();

Cuando el método termina, puede o no devolver algún resultado a quien lo llamó. Si no **devuelve nada** se dice que es un **procedimiento**. Ej: public static void mostrar()

Si **retorna un valor**, el tipo debe coincidir con el tipo declarado en la cabecera del método. Se dice que es una **función**. Ej: public static int sumar(int a, int b)

Declaración formal de métodos



modificadores -> ACCESO: **public** - **private** - **protected**

Ej: **public** static void mostrar()

```
[modificadores] tipoRetorno nombre (parámetros formales) {  
    declaraciones de variables/constantes locales;  
    sentencia_1;  
    ...  
    sentencia_n;  
[return]  
}
```

Declaración formal de métodos



modificadores -> ACCESO: `public` - `private` - `protected`

tipoRetorno -> si no retorno nada: `void`

Ej: `public` static `void` mostrar()

-> si retorno algo: tipo de lo que retorno

Ej: `public` static `int` sumar(int a, int b)

Declaración formal de métodos

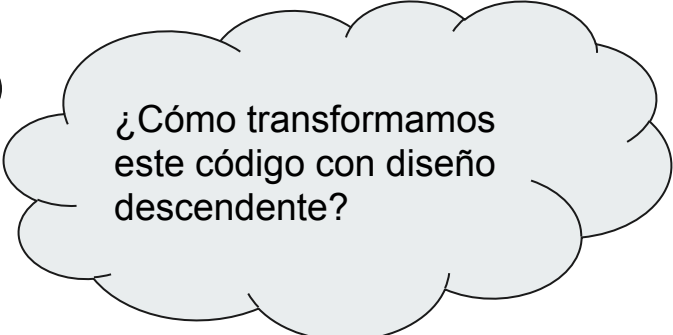
modificadores -> ACCESO: **public** - **private** - **protected**
tipoRetorno { -> si retornó nada: **void** (procedimiento)
 -> si retornó algo: tipo del retorno (función)
parámetros formales -> tipo y nombre de las variables necesarias

```
[modificadores] tipoRetorno nombre ([parámetros formales]) {  
    declaraciones de variables/constantes locales;  
    sentencia_1;  
    ...  
    sentencia_n;  
[return]  
}
```

public static int sumar(int a, int b) { **//declaración formal**

Ejemplo: Si se ingresa un nro > 0, imprime la tabla de multiplicar del 5

```
public class Clase_3_Ejemplo_3 {  
    public static void main (String [] args){  
        int entero=0;  
        try {  
            BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));  
            System.out.println("Ingrese un nro. entero: ");  
            entero = Integer.valueOf(entrada.readLine());  
            if (entero>0) {  
                System.out.println("Tabla de multiplicar del 5");  
                for (int i = 1 ; i <= 10; i++){  
                    System.out.println(5*i);  
                }  
            }  
        }  
        catch (Exception exc) {  
            System.out.println("error");  
        }  
    }  
} //fin del main  
} // fin del class
```



¿Cómo transformamos
este código con diseño
descendente?

Declaración formal de métodos (procedimiento)

```
public class Clase_3_Ejemplo_3 {  
    public static void main(String args[]) {  
        // damos por obtenido y verificado el número por teclado  
        System.out.println("Tabla de multiplicar del 5");  
        imprimirTabla5(); //declaración local  
    }  
    public static void imprimirTabla5() { //declaración formal  
        for (int i = 1 ; i <= 10; i++) {  
            System.out.println(5*i);  
        }  
    }  
}
```

¿Es un procedimiento o función?

Método agregado al Ejercicio

¿Cómo se ejecuta?

1



```
public class Clase_3_Ejemplo_3 {  
    public static void main (String [] args){  
        System.out.println("Tabla de multiplicar del 5");  
        imprimirTabla5(); //declaración local  
    }  
  
    public static void imprimirTabla5(){ //declaración formal  
        for (int i = 1 ; i <= 10; i++) {  
            System.out.println(5*i);  
        }  
    }  
}
```

¿Cómo se ejecuta?

2 →

```
public class Clase_3_Ejemplo_3 {  
    public static void main (String [] args){  
        System.out.println("Tabla de multiplicar del 5");  
        imprimirTabla5(); //declaración local  
    }  
  
    public static void imprimirTabla5(){ //declaración formal  
        for (int i = 1 ; i <= 10; i++) {  
            System.out.println(5*i);  
        }  
    }  
}
```

Consola

Tabla de multiplicar del 5

¿Cómo se ejecuta?

3 →

```
public class Clase_3_Ejemplo_3 {  
    public static void main (String [] args){  
        System.out.println("Tabla de multiplicar del 5");  
        imprimirTabla5(); //declaración local  
    }  
  
    public static void imprimirTabla5(){ //declaración formal  
        for (int i = 1 ; i <= 10; i++) {  
            System.out.println(5*i);  
        }  
    }  
}
```

Consola

Tabla de multiplicar del 5

¿Cómo se ejecuta?

```
public class Clase_3_Ejemplo_3 {  
    public static void main (String [] args){  
        System.out.println("Tabla de multiplicar del 5");  
        imprimirTabla5(); //declaración local  
    }  
}
```

4 →

```
    public static void imprimirTabla5(){ //declaración formal  
        for (int i = 1 ; i <= 10; i++) {  
            System.out.println(5*i);  
        }  
    }  
}
```

Consola

Tabla de multiplicar del 5

¿Cómo se ejecuta?

```
public class Clase_3_Ejemplo_3 {  
    public static void main (String [] args){  
        System.out.println("Tabla de multiplicar del 5");  
        imprimirTabla5(); //declaración local  
    }  
}
```

5 →

```
    public static void imprimirTabla5(){ //declaración formal  
        for (int i = 1 ; i <= 10; i++) {  
            System.out.println(5*i);  
        }  
    }  
}
```

Consola

Tabla de multiplicar del 5

¿Cómo se ejecuta?

```
public class Clase_3_Ejemplo_3 {  
    public static void main (String [] args){  
        System.out.println("Tabla de multiplicar del 5");  
        imprimirTabla5(); //declaración local  
    }  
  
    public static void imprimirTabla5(){ //declaración formal  
        for (int i = 1 ; i <= 10; i++) {  
            System.out.println(5*i);  
        }  
    }  
}
```

6 →

Consola

Tabla de multiplicar del 5
5

¿Cómo se ejecuta?



7..15 →

```
public class Clase_3_Ejemplo_3 {  
    public static void main (String [] args){  
        System.out.println("Tabla de multiplicar del 5");  
        imprimirTabla5(); //declaración local  
    }  
  
    public static void imprimirTabla5(){ //declaración formal  
        for (int i = 1 ; i <= 10; i++) {  
            System.out.println(5*i);  
        }  
    }  
}
```

Consola

```
Tabla de multiplicar del 5  
5  
10  
15  
20  
25  
30  
35  
40  
45  
50
```

¿Cómo se ejecuta?

```
public class Clase_3_Ejemplo_3 {  
    public static void main (String [] args){  
        System.out.println("Tabla de multiplicar del 5");  
        imprimirTabla5(); //declaración local  
    }  
  
    public static void imprimirTabla5(){ //declaración formal  
        for (int i = 1 ; i <= 10; i++) {  
            System.out.println(5*i);  
        }  
    }  
}
```

16 →

Consola

```
Tabla de multiplicar del 5  
5  
10  
15  
20  
25  
30  
35  
40  
45  
50
```

¿Cómo se ejecuta?

```
public class Clase_3_Ejemplo_3 {  
    public static void main (String [] args){  
        System.out.println("Tabla de multiplicar del 5");  
        imprimirTabla5(); //declaración local  
    }  
  
    public static void imprimirTabla5(){ //declaración formal  
        for (int i = 1 ; i <= 10; i++) {  
            System.out.println(5*i);  
        }  
    }  
}
```

17 →

Consola

```
Tabla de multiplicar del 5  
5  
10  
15  
20  
25  
30  
35  
40  
45  
50
```

¿Cómo se ejecuta?

18 →

```
public class Clase_3_Ejemplo_3 {  
    public static void main (String [] args){  
        System.out.println("Tabla de multiplicar del 5");  
        imprimirTabla5(); //declaración local  
    }  
  
    public static void imprimirTabla5(){ //declaración formal  
        for (int i = 1 ; i <= 10; i++) {  
            System.out.println(5*i);  
        }  
    }  
}
```

Consola

```
Tabla de multiplicar del 5  
5  
10  
15  
20  
25  
30  
35  
40  
45  
50
```

¿Cómo se ejecuta?

```
public class Clase_3_Ejemplo_3 {  
    public static void main (String [] args){  
        System.out.println("Tabla de multiplicar del 5");  
        imprimirTabla5(); //declaración local  
    }  
  
    public static void imprimirTabla5(){ //declaración formal  
        for (int i = 1 ; i <= 10; i++) {  
            System.out.println(5*i);  
        }  
    }  
}
```

19 →

FIN

Consola

```
Tabla de multiplicar del 5  
5  
10  
15  
20  
25  
30  
35  
40  
45  
50
```


Uso del return en funciones (métodos que retornan algo)




Ahora vamos a desarrollar la parte que nos faltaba para obtener el número por teclado pero con una **función**.

¿Qué tenemos que
agregar a este código?

```
public class Clase_3_Ejemplo_3 {  
    public static void main(String []args){  
        //damos por obtenido y verificado el número por teclado  
        System.out.println("Tabla de multiplicar del 5");  
        imprimirTabla5();//declaración local  
    }  
    public static void imprimirTabla5(){//declaración formal  
        for (int i = 1 ; i <= 10; i++) {  
            System.out.println(5*i);  
        }  
    }  
}
```

Función obtenerEntero() que hace uso de return



```
public static int obtenerEntero(){ <-- declaración
    int entero = -1;
    try {
        BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Ingresa un número entero: ");
        entero = Integer.valueOf(entrada.readLine());
    }
    catch (Exception exc) {
        System.out.println("error");
    }
    return entero; <-- retorna el valor
}
```

Ejercicio completo

```
public class Clase_3_Ejemplo_3 {  
    public static void main (String []args){  
        int numero=obtenerEntero();  
        if (numero!=-1){  
            System.out.println("Tabla del 5");  
            imprimirTabla5();//declaración local  
        }  
    }  
} // fin del main
```

// el código sigue al lado

Sigue

```
public static int obtenerEntero(){  
    int entero = -1;  
    BufferedReader entrada = new BufferedReader  
    (new InputStreamReader(System.in));  
    try{  
        System.out.println("Ingrese nro entero: ");  
        entero = new Integer(entrada.readLine());  
    }  
    catch (Exception exc){  
        System.out.println("error" + exc);  
    }  
    return entero;  
} // fin de la función  
public static void imprimirTabla5(){  
    for (int i = 1 ; i <= 10; i++) {  
        System.out.println(5*i);  
    }  
} // fin del procedimiento  
} // fin del class
```

Para pensar un ratito



Usando diseño descendente con al menos un método (función o procedimiento), escribir un programa que pida por teclado un carácter e imprima si es dígito (0..9), si es vocal o consonante (pueden suponer todas minúsculas, mayúsculas o ambas) o si es cualquier otro carácter. En todos los casos debe imprimir cuál fue el carácter ingresado.

```
public class Clase_3_Ejemplo_4 {
    public static void main(String []args){
        char letra= obtenerCaracter();
        if (letra>='0' && letra<='9'){
            System.out.println("Es el dígito: " + letra);}
        else if (letra>='a' && letra<='z'){
            switch(letra){
                case 'a':case 'e':case 'i':case 'o':case 'u':{
                    System.out.println("Es la vocal: " + letra);
                    Break;
                }
                default: System.out.println("Es la consonante: " + letra);
            }
        }
        else System.out.println("Es cualquier otro caracter " + letra);
    }
    public static char obtenerCaracter(){
        char caracter = ' ';
        try {
            BufferedReader entrada = new BufferedReader(new
            InputStreamReader(System.in));
            System.out.println("Ingrese un caracter: ");
            caracter = (char) (entrada.readLine().charAt(0));}
        catch (Exception exc) {
            System.out.println("error");
        }
        return caracter;
    }
}
```

/*Escribir un programa que llame un método que calcule el promedio de la suma de valores enteros entre 1 y 1000. Finalmente, el resultado debe mostrarse por pantalla.*/

```
public class Clase_3_Ejemplo_5 {  
    public static void main(String[] args) {  
        int promedio;  
        /*se invoca el método calcular_promedio_1_1000 por su nombre y sin parámetros,  
        es una función, entonces retorna un valor, y ese valor lo tengo que guardar para no perderlo,  
        si no guardo el valor, se pierde. El valor se guarda en una variable del mismo tipo que la función*/  
        promedio = calcular_promedio_1_1000();  
        System.out.println("El promedio es: "+promedio);  
    }  
    /* el encabezado del método es public static int calcular_promedio_1_1000 (),  
    es int porque retorna el promedio entero, y es función, el nombre del método tiene que ser un texto corto que  
    indique que hace, no tiene parámetros ya que no se menciona que se le pase alguna información */  
    public static int calcular_promedio_1_1000() {  
        final int MAX = 1000, MIN = 1;  
        int promedio, suma = 0;  
        for (int numero = MIN; numero <= MAX; numero++) {  
            suma += numero;  
        }  
        promedio = suma/(MAX-MIN+1);  
        return promedio;  
    }  
}
```

Ámbito de variables



- **Locales:** declarada dentro de un método. Sólo está disponible dentro de su bloque.
- **Globales:** definidas en el bloque que contiene al método que puede usarla.

Ámbito de variables

¿Se pueden tener dos variables con el mismo nombre?

```
public class Clase_3_Ejemplo_6 {  
    public static final int a = 2; // constante global  
    public static int b = 2; // variable global  
    public static void main(String []args) {  
        int a = 3; // local ← accesible sólo dentro de resolver  
        System.out.println ("a = " + a);  
        System.out.println ("b = " + b);  
    }  
}
```

¿Qué imprime por la consola?

Ámbito de variables



```
public class Clase_3_Ejemplo_7 {//BLOQUE del programa Clase_3_Ejemplo_7
    public static int numero = 2;//variable global a todos
    public static void main(String[] args) {//BLOQUE de main
        int a = 3;//variable local a main
        //un bloque se define por {}
        {//BLOQUE A
            System.out.println (a + ", " + numero);
            int b = 2; //variable local al BLOQUE A
            System.out.println (a + ", " + b);
            {//BLOQUE B
                int c = 3; //variable local al BLOQUE B
                System.out.println (a + ", " + b + ", " + c);
            }//FIN BLOQUE B
            //ERROR EN LA PROXIMA SENTENCIA, c SOLO PERTENCE AL BLOQUE B
            System.out.println (a + ", " + b + ", " + c);
        }//FIN BLOQUE A
    }//FIN BLOQUE main
}//FIN BLOQUE Clase_3_Ejemplo_7
```

Evitar la definición de bloques internos y variables globales

Definir constantes globales si hay más de un método que necesita usarlas

Definir las variables locales de cada método

Ámbito de variables



```
/*Escribir un programa que llame un método que calcule el promedio de la suma de valores enteros
entre 1 y 1000. Finalmente, el promedio debe mostrarse por pantalla.*/
public class Clase_3_Ejemplo_8 {
    /*Como usamos las mismas constantes en calcular_promedio_1_1000 y en main (en el mensaje de la
    impresión) definimos MAX Y MIN como constantes globales */
    public static final int MAX = 1000, MIN = 1;
    public static void main(String[] args) {
        int promedio;
        promedio = calcular_promedio_1_1000();
        System.out.println("El promedio de la suma entre " MIN " y " MAX " es "+promedio);
    }
    public static int calcular_promedio_1_1000() {
        int promedio, suma = 0;
        for (int numero = MIN; numero <= MAX; numero++) {
            suma += numero;
        }
        promedio = suma/(MAX-MIN+1);
        return promedio; //LA FUNCION DEBE RETORNAR ALGO DEL MISMO TIPO QUE FIGURA EN LA DECLARACION
    }
}
```



En resumen...

- En el desarrollo de una **solución**, uno de los pasos previos a la implementación es el **diseño**, guardar una idea y los pasos que habría que hacer.
- Un resultado ideal de un **diseño** es obtener un **pseudocódigo** (por ahora la única técnica que se explicó), que guarde la idea de los métodos que se van a tener que implementar.
- Generar un buen diseño permite **abstraer** del lenguaje, lo que se va a implementar.
- En programación 1 no se pedirán los **diseños** ni los pasos previos al desarrollo del programa, salvo que se mencione explícitamente. Solo se evaluarán los programas con los métodos correspondientes.
- El diseño con pseudocódigo debería **ayudar** al alumno/a a guardar la idea de lo que desea implementar.

Parámetros



- Un parámetro es una variable cuyo valor es proporcionado por la unidad invocadora.
- Es como un **hueco** donde puedo pasar un **dato** para que lo use otro método.
- Pueden pasarse 1 o más parámetros pero hay que respetar cantidad, posición y tipo.

```
public class Clase_3_Ejemplo_9 {  
    public static void main(String []args){  
        int n; // variable local a resolver  
        n = obtenerNatural();  
        imprimirTabla(n);  
    }  
}
```

Parámetros



```
public static int obtenerNatural() {  
    int valor = 0; // variable local a obtenerNatural  
    BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));  
    while(valor <= 0) {  
        try {  
            System.out.println ("Ingrese un número natural");  
            valor = Integer.valueOf(entrada.readLine());  
        }  
        catch (Exception exc ) {  
            System.out.println("Error: " + exc);  
        }  
    }  
    return valor;  
}
```

Parámetros



```
public static void imprimirTabla5(){//declaración formal
    for (int i = 1 ; i <= 10; i++) {
        System.out.println(5*i);
    }
}
```

¿Cuál es la diferencia entre lo que teníamos y lo nuevo?

Nombre del procedimiento

```
public static void imprimirTabla(int numero) {
    for (int i = 1 ; i <= 10; i++) {
        System.out.println(numero*i);
    }
}
```

Parámetro

Parámetros



Los métodos sin
parámetros



Realiza una tarea única



Un parámetro es una variable cuyo valor es proporcionado por la
unidad invocadora.



Parámetros en la
declaración de un método



Pueden no haber, tener
uno o más



Las variables que se colocan como parámetro cuando se
invoca un método tienen que coincidir en tipo y orden
según la declaración

Parámetros



- En **java** el pasaje de parámetros es por **copia**.
- Se replica la variable y dentro del método se trabaja sobre la réplica.
- Al finalizar el método se pierde la réplica y la unidad invocadora no percibe cambios en la variable utilizada como parámetro de invocación.

Parámetros

Dos parámetros enteros.

Importante: respetar el orden aunque en este caso son los dos enteros

```
public class Clase_3_Ejemplo_10{  
    public static int funcion(int a, int b) {  
        a = 1; //pierde el valor que tenía como parámetro  
        b = a + 5; //pierde el valor que tenía como parámetro  
        return b;  
    }  
    public static void main(String []args){  
        int m = 3;  
        int n = 7;  
        int x = funcion(m,n);  
        System.out.println("Valor de m: " + m);  
        System.out.println("Valor de n: " + n);  
        System.out.println("Resultado de la función: " + x);  
    }  
}
```

¿Cuál es la salida por consola?

Parámetros

```
public class Clase_3_Ejemplo_10{  
    public static int funcion(int a, int b) {  
        a = 1; //pierde el valor que tenía como parámetro  
        b = a + 5; //pierde el valor que tenía como parámetro  
        return b;  
    }  
    public static void main(Sting []args){  
        int m = 3;  
        int n = 7;  
        int x = funcion(m,n); // x vale 6  
        System.out.println("Valor de m:  " + m); // imprime 3  
        System.out.println("Valor de n:  " + n); // imprime 7  
        System.out.println("Resultado de la función: " + x); // imprime 6  
    }  
}
```

Parámetros

```
public class Clase_3_Ejemplo_11{  
    public static int funcion(int a, int b) {  
        a = 1; //pierde el valor que tenía como parámetro  
        b = a + 5; //pierde el valor que tenía como parámetro  
        return b;  
    }  
    public static void main(String []args){  
        int a = 3; //no importa que se llamen igual que en método  
        int b = 7;  
        int x = funcion(a,b); // x vale 6  
        System.out.println("Valor de a: " + a); // imprime 3  
        System.out.println("Valor de b: " + b); // imprime 7  
        System.out.println("Resultado de la función: " + x); // imprime 6  
    }  
}
```

Hasta ahora



- Pseudocódigo
- Diseño descendente con métodos: procedimientos y funciones
- Ámbito de una variable: local y global
- Parámetros

Errores comunes en métodos

- Una función **siempre necesita retornar algo**.

```
public static (tipo no void) nombre_funcion (...){  
...  
return ...;  
}
```

- Una función cuando se invoca sólo puede estar a la derecha de una asignación y dentro de una expresión lógica.

```
...  
numero = nombre_funcion(...);  
...  
If (MIN<nombre_funcion(...)){  
...  
}
```

- Los métodos sin parámetros resuelven una sola cosa. Los métodos tienen que tener parámetros para poder reutilizar código.

Flujos de retorno



```
...  
public static boolean esPositivo(int x) {  
    if (x<0)  
        return false;  
    if (x>0)  
        return true;  
  
}  
...
```

Para pensar otro ratito
¿Tiene algún problema esta función?

Flujos de retorno



```
public static boolean esPositivo(int x) {  
    if (x<0)  
        return false;  
    if (x>0)  
        return true;  
    // Error: retorno perdido si x es igual a cero.  
}
```

Flujos de retorno



```
public static boolean esPositivo(int x) {  
    if (x<=0)  
        return false;  
    else  
        return true;  
}
```


Flujos de retorno



```
public static boolean esPositivo(int x) {  
    if (x<=0)  
        return false;  
    else  
        return true;  
}
```

¿Qué cambió?

```
public static boolean esPositivo(int x) {  
    if (x<=0)  
        return false;  
  
    return true;  
}
```

Flujos de retorno



```
public static boolean esPositivo(int x){
    if (x<=0) return false;
    else return true;
}

public static boolean esPositivo(int x) {
    if (x<=0) return false;
    return true;
}

public static boolean esPositivo(int x) {
    return (x>0);
}
```

¿Y ahora?

Para resolver



Realizar un programa que dado un carácter ingresado desde teclado (**a** o **b**) permita realizar dos operaciones entre dos enteros **N** y **M** menores a 10 ingresados desde teclado también. Las operaciones son:


- _ Si el usuario ingresa **a** obtener la suma entre **N** y **M**.
- _ Si el usuario ingresa **b** obtener la resta entre **N** y **M**.

Un Pseudocódigo posible



```
public class Clase_3_Ejemplo_12 {  
    definir constante MAX = 10  
    public static void main(String[] args){  
        definir opcion, N y M  
        opcion = obtener caracter a o b()  
        N = obtener numero < MAX()  
        M = obtener numero < MAX()  
        resolver operaciones  
    }  
}
```

Una solución posible 1/3



```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_3_Ejemplo_12 {
    final int MAX=10; // defino la constante global
    public static void main(String []args){
        int N, M =0;
        char opcion='a';
        opcion=obtenerOpcionValida();
        N=obtenerNumeroValido();
        M=obtenerNumeroValido();
        if (opcion=='a'){
            System.out.println("La suma de " + M + " y " + N + " es: " + (N+M));
        }
        else {
            System.out.println("La resta de " + M + " y " + N + " es: " + (N-M));
        }
    }
}
```

Una solución posible 2/3



```
public static char obtenerOpcionValida(){
    char character = 'c';
    BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
    while (!(character=='a' || character=='b')){
        try {
            System.out.println("Ingrese una opcion valida entre a y b");
            character = (char)(entrada.readLine().charAt(0));
        }
        catch (Exception exc) {
            System.out.println("error");
        }
    }
    return character;
}
```

Una solución posible 3/3

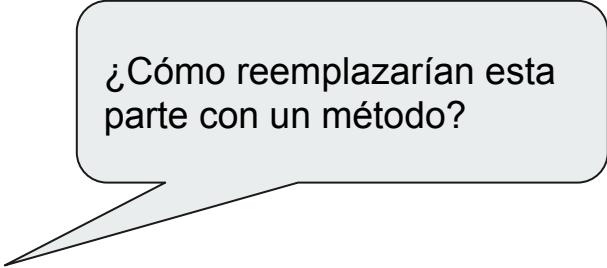


```
public static int obtenerNumeroValido() {  
    int valor = MAX + 1; // variable local a obtenerNatural  
    BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));  
    while (valor > MAX){ // se puede esperar que valor <= MAX  
        try{  
            System.out.println ("Ingrese un numero entre menor a " + MAX);  
            valor = new Integer(entrada.readLine());  
        }  
        catch (Exception exc ) {  
            System.out.println ("Error al ingresar número válido" + exc);  
        }  
    }  
    return valor;  
}  
} // fin del class
```



```
import java.io.BufferedReader;
import java.io.InputStreamReader;
```

```
public class Clase_3_Ejemplo_12 {
    final int MAX=10; // defino la constante global
    public static void main(String []args){
        int N, M =0;
        char opcion='a';
        opcion=obtenerOpcionValida();
        N=obtenerNumeroValido();
        M=obtenerNumeroValido();
        if (opcion=='a'){
            System.out.println("La suma de "+ M + " y " + N + " es: " + (N+M));
        }
        else {
            System.out.println("La resta de "+ M + " y " + N + " es: " + (N-M));
        }
    }
}
```



¿Cómo reemplazarían esta parte con un método?


```
//Realizar un programa que dado dos números enteros ingresados por el usuario, muestre por pantalla el resultado
//de las operaciones matemáticas básicas: la suma, la resta, la multiplicación y la división entre ambos números.
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_3_Ejemplo_13 {
    public static void main(String[] args) {
        int numero1, numero2;
        BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
        try{
            System.out.println("Ingrese un número entero: ");
            numero1 = Integer.valueOf(entrada.readLine());
            System.out.println("Ingrese otro número entero: ");
            numero2 = Integer.valueOf(entrada.readLine());
            //LAS VARIABLES INGRESADAS COMO PARAMETROS DEBEN SER DEL MISMO TIPO Y ORDEN
            //DE LA DEFINICION DEL METODO
            imprimir_resultados_operaciones_matematicas (numero1, numero2);
        }
        catch (Exception exc){
            System.out.println(exc);
        }
    }
    public static void imprimir_resultados_operaciones_matematicas (int numero1, int numero2){
        System.out.println(numero1+"+"+numero2+"="+ (numero1+numero2));
        System.out.println(numero1+"-"+numero2+"="+ (numero1-numero2));
        System.out.println(numero1+"*"+numero2+"="+ (numero1*numero2));
        System.out.println(numero1+"/"+numero2+"="+ (numero1/numero2));
    }
}
```

Algunos tips



- Usar pseudocódigo para detallar a grandes rasgos las soluciones a los problemas planteados
- Tratar de dividir el problema en subproblemas más pequeños que sean más sencillos de resolver (Modularizar las soluciones)
- Los métodos que retornan algo son funciones y los void son procedimientos
- Usar parámetros para reutilizar las soluciones desarrolladas
- Prestar atención en el llamado a métodos ya que si llevan parámetros deben respetar la cantidad y la posición.
- Si el método, en su definición formal, tiene más de un parámetro y son de diferentes tipos, cuando se llama se debe respetar el tipo que corresponde a cada parámetro. Si no se respeta puede generar errores sintácticos.
- En caso de tener algunos o todos los parámetros del mismo tipo hay que prestar atención como se hace el llamado ya que si me equivoco el orden y respeto los tipos puede ser un error semántico y eso no lo detecta ningún entorno.
- Siempre que llamo a una función debo guardar el retorno en una variable del mismo tipo o hacer uso del resultado para poner en una expresión lógica o como parámetro para algún método o simplemente como salida en la consola.
- Declarar variables lo más cerca de donde se van a usar (locales), evitar el uso de globales y por ahora solo constantes globales.