Diseño descendente

Programación I

Objetivos del tema

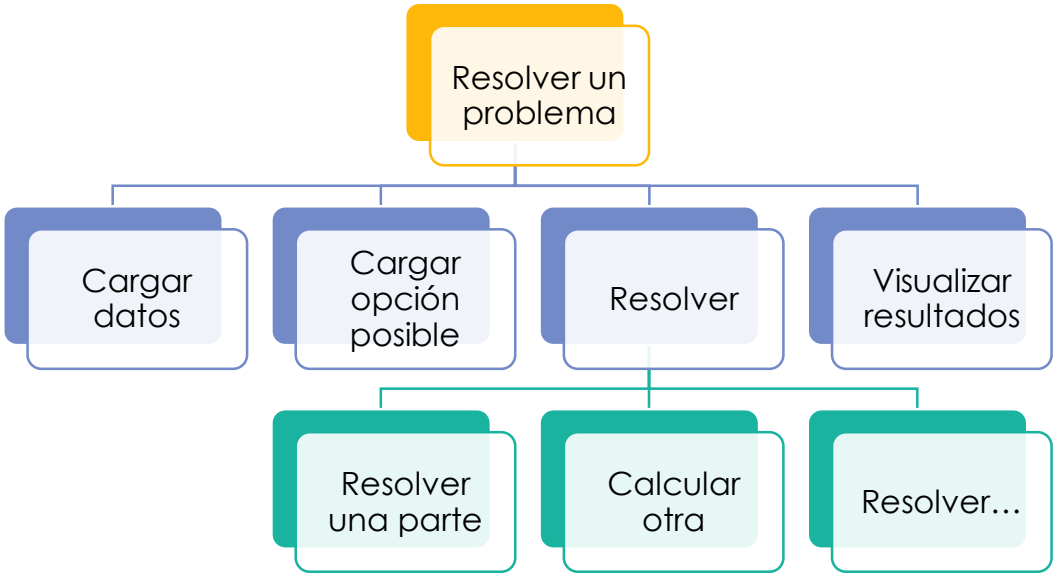
- Resolver problemas aplicando un diseño descendente
- Aplicar diseño descendente utilizando métodos (procedimientos y funciones) y parámetros

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 tares 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

Ejemplo de diseño por pseudocódigo

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

```
/*Escribir un diseño de un programa que solicite al usuario el ingreso de
  un número entero mayor que 0 e imprima la tabla de multiplicar de dicho
  número.
*/
/*Un diseño con pseudocodigo puede describirse en el cuerpo del programa
  {…} con frases cortas, vinculados con los pasos (resolución de
  subproblemas) a realizar para resolver el enunciado
*/
public class Ejemplo_diseño {
  public static void main(String[] args){
        definir una variable numero
        obtener desde teclado un numero mayor que 0
        imprimir tabla de multiplicar de numero
  }
}
```

Video de explicación
https://drive.google.com/drive/folders/1wWyFJz2KcuY_c0kUqtvCXyWG0yzZQmGz?usp=sharing

Ejemplo de diseño por pseudocódigo

- Los pseudocódigos son casi códigos. Y a veces se le puede dar más detalle para que luego sea más fácil de programar. Por ejemplo, para el enunciado anterior.

```
/*Escribir un diseño de un programa que solicite al usuario el ingreso de un
  número entero mayor que 0 e imprima la tabla de multiplicar de dicho número.
*/
/*Un diseño con pseudocodigo puede describirse en el cuerpo del programa {…}
  con frases cortas, vinculados con los pasos (resolución de subproblemas) a
  realizar para resolver el enunciado
*/
public class Ejemplo_diseño {
  public static void main(String[] args){
        definir una variable numero
        obtener desde teclado un numero
        si numero mayor que 0
              imprimir tabla de multiplicar de numero
  }
}
```

Ejemplo de diseño por pseudocódigo

```
/*Escribir un diseño de un programa que mientras el usuario ingrese
  de un número entero mayor que 0, imprima la tabla de multiplicar
  de dicho número
*/
/*Un diseño con pseudocodigo puede describirse en el cuerpo del
  programa {…} con frases cortas, vinculados con los pasos
  (resolución de subproblemas) a realizar para resolver el enunciado
*/
public class Ejemplo_diseño {
  public static void main(String[] args){
        definir numero entero
        obtener un numero por teclado
        mientras numero sea mayor que 0
              imprimir tabla de multiplicar de numero
              obtener un numero por teclado
  }
}
```

Práctico – primera parte

1. Escribir un diseño de programa que mientras que el usuario ingrese un número distinto de 0, pida ingresar otro numero y lo imprima.
2. Escribir un diseño de programa que mientras que el usuario ingrese un caracter dígito o caracter letra minúscula, imprima dicho caracter, y si es caracter letra minúscula, imprima si es vocal o consonante.

Métodos

Un método es una sección de código que puede ser llamado desde el bloque principal **main(){…}** u otros métodos, para realizar alguna tarea específica o resolver un subproblema.

El método es llamado por su nombre seguido por una secuencia de parámetros o argumentos (datos utilizados por el propio método para sus cálculos) entre paréntesis.

Un método puede o no retornar un valor a la unidad (principal u otro método) que lo llama.

Un método puede retornar un valor a través de la sentencia return

Declaración de métodos

```
[modificadores] tipo_de_dato nombre_de_metodo (parametros formales){
 sentencias de declaracion de variables locales;
 sentencia_1;
 ...
 sentencia_n;
}
```

- La primera línea es el encabezado del método.
- Los modificadores especifican cómo puede llamarse al método (principalmente quienes los pueden usar o no).
- El tipo de dato se refiere al valor que puede retornar el método.
- El nombre de un método se utiliza para invocarlo y es una identificación del mismo respecto de los otros métodos.
- Los parámetros (entre paréntesis) se utilizan para introducir y/u obtener información con el método.
- La declaración de un método se denomina declaración formal.

Método main

- Hasta ahora hemos venido utilizando un método que no lo conocíamos como tal.
- main es un método que usamos para resolver los ejercicios colocando sentencias en orden dentro de sus **{…}**.

```
/*Dado un numero entero con valor inicial 1, hacer una iteración que haga
  incrementar numero de a uno hasta un valor MAX = 4 (constante). Mientras
  itera deberá imprimir numero
*/
public class Clase_6_Ejemplo_1 {
 public static void main (String [] args) {
        final int MAX = 4;
        for (int numero = 1; numero <= MAX; numero++){
                System.out.println("El numero es: " + numero);
        }
  }
}
```

Ejemplo 1: corresponde al ejemplo 6 de la clase 5

```
/* public static void main (String [] args) es un método cuyo encabezado es
public static void main (String [] args).
Los modificadores son public static, el tipo de dato es void, el nombre del
método es main, y los parámetros son (String [] args).
La declaración de un método (public static void main (String [] args){…}) se
denomina declaración formal.
*/
/* Se puede decir que main es el método principal del programa
Clase_5_Ejemplo_6, y se utiliza para resolver el ejercicio.
*/
public class Clase_6_Ejemplo_1 {
  public static void main (String [] args) {
        final int MAX = 4;
        for (int numero = 1; numero <= MAX; numero++){
               System.out.println("El numero es: " + numero);
        }
  }
}
```

Ejemplo 1: corresponde al ejemplo 6 de la clase 5

```
/* Los modificadores son public static hacen al método accesible al resto.
public y static permite que el método sea accesible desde afuera del programa
y/o como va ser su acceso.
LOS MODIFICADORES SON MUY ÚTILES EN PROGRAMACIÓN 2 (PROGRAMACIÓN ORIENTADA A
OBJETOS). EN PROGRAMACIÓN 1 SOLO SE USARÁN COMO PARTE DE LA DECLARACIÓN Y NO
VAN A MODIFICARSE.
*/

public class Clase_6_Ejemplo_1 {
  public static void main (String [] args) {
        final int MAX = 4;
        for (int numero = 1; numero <= MAX; numero++){
              System.out.println("El numero es: " + numero);
        }
  }
}
```

Ejemplo 1: corresponde al ejemplo 6 de la clase 5

```
/* Los métodos pueden retornar valores a través de su invocación o a través de
sus parámetros.
void es un tipo de dato que no hemos visto hasta ahora y que se utiliza para
indicar que el método NO va a retornar un valor a través de su invocación.
main, que es el nombre de método, se utiliza para invocarlo, y es una
identificación del mismo respecto de los otros métodos.
*/

public class Clase_6_Ejemplo_1 {
  public static void main (String [] args) {
        final int MAX = 4;
        for (int numero = 1; numero <= MAX; numero++){
               System.out.println("El numero es: " + numero);
        }
  }
}
```

Ejemplo 1: corresponde al ejemplo 6 de la clase 5

```
/* Dentro de los paréntesis se ubican los parámetros. En este caso el único
parámetros es args de tipo String []. Este tipo se desarrollará en una clase
posterior.
*/

public class Clase_6_Ejemplo_1 {
  public static void main (String [] args) {
        final int MAX = 4;
        for (int numero = 1; numero <= MAX; numero++){
              System.out.println("El numero es: " + numero);
        }
  }
}
```

Definición de métodos

- Dejando para más adelante en esta misma clase el uso de parámetros, vamos a definir y usar métodos.

```
/*Hacer un método que dado un numero entero con valor inicial 1, haga una
  iteración incrementando numero de a uno hasta un valor MAX = 4 (constante).
  Mientras itera deberá imprimir numero. Luego invocarlo desde el programa
  principal.
*/
public class Clase_6_Ejemplo_2 {
  public static void main (String [] args) {
        imprimir_1_a_4 ();
        System.out.println("Termino");
  }

  public static void imprimir_1_a_4 () {
        final int MAX = 4;
        for (int numero = 1; numero <= MAX; numero++){
              System.out.println("El numero es: " + numero);
        }
  }
}
```

Ejemplo 2

```
/*Hacer un método que dado un numero entero con valor inicial 1, haga una iteración incrementando
  numero de a uno hasta un valor MAX = 4 (constante). Mientras itera deberá imprimir numero. Luego
  invocarlo desde el programa principal.
*/
public class Clase_6_Ejemplo_2 {
/*main y imprimir_1_a_4 son métodos del programa Clase_6_Ejemplo_2 y están dentro de sus {…}
*/
  public static void main (String [] args) {
  /*main es el método principal del programa e invoca al método imprimir_1_a_4 por su nombre y sin
  parámetros como en la declaración; cuando se lo llama se dice que es una declaración o invocación
  local
  */
        imprimir_1_a_4 ();
        System.out.println("Termino");
  }
  public static void imprimir_1_a_4 () {
  /*el encabezado del método es public static void imprimir_1_a_4 ();
  es void porque NO retorna nada, solo debe hacer que imprima por pantalla;
  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;
  la definición del método se dice que es una declaración formal
  */
        final int MAX = 4;
        for (int numero = 1; numero <= MAX; numero++){
                System.out.println("El numero es: " + numero);
        }
  }
}
```

Ejemplo 2

¿Qué hace el programa Clase_6_Ejemplo_2 cuando se ejecuta?

Primero ejecuta el método principal, main, y se ejecutan sentencias que incluyen en el orden en que aparecen.

Llama o invoca al método imprimir_1_a_4 () //no tiene parámetros;
Estando en el método imprimir_1_a_4 () define una constante MAX = 4;
Para el for define una variable numero = 1;

Pregunta numero<=MAX, en valores sería si (1<=4) es verdadero, entonces
Imprime El numero es: 1
Luego ejecuta numero++, que es igual a numero=numero+1, por lo tanto numero=2
Por estar dentro de la sentencia for(...){...} se vuelve al comienzo de dicha sentencia

Pregunta numero<=MAX, en valores sería si (2<=4) es verdadero, entonces
Imprime El numero es: 2
Luego ejecuta numero++, que es igual a numero=numero+1, por lo tanto numero=3
Por estar dentro de la sentencia for(...){...} se vuelve al comienzo de dicha sentencia

Pregunta numero<=MAX, en valores sería si (3<=4) es verdadero, entonces
Imprime El numero es: 3
Luego ejecuta numero++, que es igual a numero=numero+1, por lo tanto numero=4
Por estar dentro de la sentencia for(...){...} se vuelve al comienzo de dicha sentencia

Pregunta numero<=MAX, en valores sería si (4<=4) es verdadero, entonces
Imprime El numero es: 4
Luego ejecuta numero++, que es igual a numero=numero+1, por lo tanto numero=5
Por estar dentro de la sentencia for(...){...} se vuelve al comienzo de dicha sentencia

Pregunta numero<=MAX, en valores sería si (5<=4) es falso,
entonces pasa a la próxima sentencia después de las llaves {...} del for

Como no hay más sentencias para ejecutar sale de imprimir_1_a_4 y vuelve a la próxima sentencia que sigue de la invocación desde main, en este caso System.out.println("Termino").
Imprime por pantalla Termino y como en main no hay más sentencias el programa termina.

```
public class Clase_6_Ejemplo_2 {

  public static void main (String [] args) {
  imprimir_1_a_4 ();
  System.out.println("Termino");
  }

  public static void imprimir_1_a_4 () {
  final int MAX = 4;
  for (int numero=1;numero<=MAX;numero++){
    System.out.println("El numero es: "+numero);
  }
  }
}
```

Métodos

- Los métodos pueden ser dos: procedimientos y funciones.
- Las funciones se utilizan para retornar un solo valor a quien lo invoca, y generalmente ese valor es el resultado de un cálculo (obtener menor, calcular promedio, …).
- Una función retorna el valor que calculó usando la sentencia **return nombre_variable;**. Por ejemplo, si dentro de la función el resultado está en una variable suma, al final del método entre {…} se coloca **return suma;**

```
public static tipo_de_dato nombre_de_metodo (parametros formales){
  sentencia_1;
  ...
  return suma;
}
```

- Los procedimientos se utilizan para resolver un problema concreto, que no es realizar un calculo o un recorrido para retornar un valor (imprimir una serie de numero, …)

Ejemplo 3

```
/*Escribir un programa que llame un método que calcule el promedio de la suma de valores enteres entre 1 y 1000.
Finalmente, el resultado debe mostrarse por pantalla.
*/
public class Clase_6_Ejemplo_3 {
   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 valor, y ese valor lo tengo que guardar para no perderlo,
            si no guardo el valor estaría mal porque es como que no hace nada, lo que devuelve 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;
            final int MIN = 1;
            int promedio;
            int suma = 0;
            for (int numero = MIN; numero <= MAX; numero++) {
                        suma += numero;
            }
            promedio = suma/(MAX-MIN+1);
            return promedio;
   }
}
```

Video de explicación

https://drive.google.com/drive/folders/1wWyFJz2KcuY_c0kUqtvCXyWG0yzZQmGz?usp=sharing

Ejemplo 4

```
/*Escribir un programa que mientras el usuario cargue desde teclado un numero distinto de 0, llame a un método que imprima por pantalla los numeros
   entre 1 y 4 de forma creciente
*/
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_6_Ejemplo_4 {
   public static void main (String [] args) {
            int numero;
            BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
            try{
                        System.out.println("Ingrese un entero (0 para salir): ");
                        numero = Integer.valueOf(entrada.readLine());
                        while(numero!=0){
                                    imprimir_1_a_4 ();
                                    System.out.println("Ingrese un entero (0 para salir): ");
                                    numero = Integer.valueOf(entrada.readLine());
                        }
            }
            catch(Exception exc){
                        System.out.println(exc);
            }
   }
   public static void imprimir_1_a_4 (){
            final int MAX = 4;
            final int MIN = 1;
            for (int i = MIN;i <= MAX;i++){
                        System.out.println("numero es: "+i);
            }
    }
}
```

Video de explicación

https://drive.google.com/drive/folders/1wWyFJz2KcuY_c0kUqtvCXyWG0yzZQmGz?usp=sharing

Repasando diferencias entre funciones y procedimientos

Funciones:

- Calcular… promedio, cantidad, operación compleja
- Obtener…mayor, menor, un índice, un valor
- Retornan un valor de tipo primitivo
- Puede ser parte de una expresión lógica o siempre estar a la derecha de una asignación

Procedimientos:

- Generar salida de resultados, imprimir
- Procesar o resolver un problema

Práctico – segunda parte

3. Escribir un programa que mientras el usuario cargue desde teclado un caracter letra minúscula, llame a un método que imprime por pantalla la tabla de multiplicar de 9.
4. Escribir un programa que mientras el usuario cargue desde teclado un numero entero distinto de 0, imprima por pantalla la suma que se obtiene de invocar un método que calcula la sumatoria de los primeros 200 números naturales (son números enteros en 1 y 200).

Ámbito o alcance de variables y constantes

- Según donde se ubiquen las variables y constantes se clasifican en:
  - Locales: declarada dentro de un método. Sólo está disponible para el mismo.
  - Globales: declaradas fuera de los métodos, y pueden ser usadas por los distintos métodos (***no usar variables globales***).

```
public class Ejemplo {
  //constante global a todos los metodos del programa Ejemplo
  public static final int MIN = 2;
  //variable global a todos los metodos del programa Ejemplo
  public static int b = 2;//SE RECOMIENDA NO USAR
  public static void main(String[] args) {
        //constante local del metodo main
        final int MAX = 2;
        //variable local del metodo main
        int a = 3;
        System.out.println ("a = "+a);
        System.out.println ("b = "+b);
  }
}
```

Ámbito o alcance de variables y constantes

El ámbito o alcance de una variable es la zona accesible.

Variables "globales": disponibles a todos.	Variables locales • Están disponibles desde su declaración hasta el final del método. • No son visibles desde otros métodos métodos. • Distintos métodos pueden contener variables con el mismo nombre. El nombre de una variable local debe ser único dentro de su ámbito.	Variables de bloque {…} • Están disponibles desde su declaración hasta el final del bloque. • No son visibles desde otros bloques. • Distintos bloques pueden contener variables con el mismo nombre. Si un bloque contiene otro bloque, en el bloque interior no se puede declarar una variable con el mismo nombre.	En todos los casos las variables no tienen un valor inicial por defecto. El programador es el encargado de asignarles valores.

Ámbito o alcance de variables y constantes

```
public class Ejemplo {//BLOQUE del programa Ejemplo
  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 Ejemplo
```

Evitar 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

Ejemplo 5

```
/* Similar a Clase_6_Ejemplo_3. Escribir un programa que llame un método que calcule el promedio de la suma
   de valores enteres entre 1 y 1000. Finalmente, el promedio debe mostrarse por pantalla.
*/
public class Clase_6_Ejemplo_5 {
   /*Como usamos las mismas constantes en calcular_promedio_1_1000 y en main (en el mensaje de la impresion)
   definimos MAX Y MIN como constantes globales
   */
   public static final int MAX = 1000;
   public static final int 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;
          int 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
   }
}
```

Diseño descendente y métodos

- En este punto conocemos un poco de diseño descendente, métodos, y alcance de variables/constantes.
- Ya vinculamos los métodos con el alcance, y nos falta asociar el diseño con los métodos y utilizar parámetros.
- Dejando la utilización de parámetros para los último, ahora vamos asociar el diseño descendente con métodos.

Ejemplo de diseño por pseudocódigo

```
/*Escribir un diseño de un programa que mientras el usuario ingrese
  de un número entero mayor que 0, imprima la tabla de multiplicar
  de 10. Cuando salga del ciclo vuelva a imprimir la tabla de
  multiplicar de 10.
*/
public class Ejemplo_diseño {
  public static void main(String[] args){
        definir numero entero
        obtener un numero por teclado
        mientras numero sea mayor que 0
              imprimir tabla de multiplicar de 10
              obtener un numero por teclado
        imprimir tabla de multiplicar de 10
  }
}
```

Ejemplo de diseño por pseudocódigo

- Teniendo la serie de pasos de como se va a resolver el problema (pseudocodigo o no) el paso siguiente es identificar posibles métodos
- Para identificarlos vamos a buscar y marcar frases que pueden indicar hacer una tarea o subproblema que se repite, y/o que puede no se sencilla de poder implementar en una etapa posterior.

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

Ejemplo de diseño por pseudocódigo

- **obtener un numero por teclado** puede ser algo simple de resolver, por ejemplo con un mensaje al usuario sobre los que tiene que ingresar y la sentencia de carga desde teclado.
- Sin embargo otros podrán hacer un método que realice más validaciones sobre lo ingresado.
- Ambas soluciones son factibles.

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

Ejemplo de diseño por pseudocódigo

- **imprimir tabla de multiplicar de 10** es seguro de que es un método que necesita iterar y mientras itera imprime el resultado de una multiplicación.
- Como no devuelve ningún valor es un procedimiento.

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

Ejemplo de diseño por pseudocódigo

- Otro cosa para analizar es si al haber descripto los pasos, los mismos se pueden agrupar en un método.
- En este caso el método itera, mientras itera pide ingresar un numero e imprime una tabla de multiplicar.
- Como haría muchas cosas que no implican una tarea concreta no es común asociar dichos pasos a un método.

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

Ejemplo de diseño por pseudocódigo

- El resultado final tendría que ayudar a identificar los métodos, y en cada caso los pasos a realizar.

```
public class Ejemplo_diseño {
  public static void main(String[] args){
        definir numero entero
        obtener un numero por teclado
        mientras numero sea mayor que 0
                imprimir tabla de multiplicar de 10
                obtener un numero por teclado
        imprimir tabla de multiplicar de 10
  }
  public static void imprimir tabla de multiplicar de 10 (){
        definir multiplicador entero
        mientras multiplicador sea menor igual que 10
                imprimir multiplicador x 10
                incrementar multiplicador
  }
}
```

Ejemplo 6

```
/*Escribir un diseno de un programa que mientras el usuario ingrese de un numero entero mayor que 0, imprima la tabla de multiplicar de 10. Cuando salga del ciclo vuelva a imprimir
    la tabla de multiplicar de 10.
*/
import java.io.BufferedReader;
import java.io.InputStreamReader;
class Clase_6_Ejemplo_6 {
    public static final int MIN = 0;
    public static final int MINMULTIPLICADOR = 10;
    public static final int MAXMULTIPLICADOR = 10;
    public static final int VALOR = 10;
    public static void main(String[] args) {
                int numero;
                BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
                try{
                                System.out.println("Ingrese un numero entero natural");
                                numero = Integer.valueOf(entrada.readLine());
                                while (numero > MIN){
                                    imprimir_tabla_de_multiplicar_de_10();
                                    System.out.println("Ingrese un numero entero natural");
                                    numero = Integer.valueOf(entrada.readLine());
                                }
                                imprimir_tabla_de_multiplicar_de_10 ();
                }
                catch (Exception exc){
                                System.out.println(exc);
                }
    }
    public static void imprimir_tabla_de_multiplicar_de_10 (){
                for (int multiplicador = 1; multiplicador <= MAXMULTIPLICADOR; multiplicador++){
                                System.out.println(multiplicador +" * "+VALOR+" = "+(multiplicador*VALOR));
                }
    }
}
```

Video de explicación

https://drive.google.com/drive/folders/1wWyFJz2KcuY_c0kUqtvCXyWG0yzZQmGz?usp=sharing

Ejemplo de diseño por pseudocódigo

- Como las resoluciones pueden llegar a tener muchas variables, en general no se mencionan las declaraciones de variables y constantes.

```
/*Escribir un diseño de programa que llame un método que calcule el promedio de
  la suma de valores enteres entre 1 y 1000. Finalmente, el resultado debe
  mostrarse por pantalla.
*/
public class Ejemplo_diseño {
  public static void main(String[] args){
        promedio = calcular_promedio_1_1000()
        imprimir promedio
  }
  public static int calcular_promedio_1_1000 (){
        mientras un numero recorra de 1 a 1000
                acumulo en una suma el numero
        calculo el promedio de lo que sumo y lo retorno
  }
}
```

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 pseudocodigo (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 pseudocodigo debería ayudar a alumno/a a guardar la idea de lo que desea implementar.

Ejemplo 7: Paramétros

- Escribir un diseño de programa y luego implementar: dado un número ingresado por el usuario, si el numero es natural imprima la tabla de multiplicar de ese número.

```
public class Clase_6_Ejemplo_7 {
  public static void main(String[] args){
        obtener numero_natura
        si es numero_natural
              imprimir tabla de multiplicar (numero_natural)
  }
}
```

- Para que el método pueda imprimir la tabla de multiplicar del numero_natural ingresado debo informárselo por parámetro.
- Si el método imprimir tabla de multiplicar está pensado además para leer de consola ese número estaría MAL, ya que el método haría más que imprimir.

Ejemplo 7: Paramétros

- En el siguiente diseño se puede ver que el método imprimir tabla de multiplicar no genera siempre lo mismo, dependiendo del numero ingresado por el usuario imprime la tabla de multiplicar de ese numero.
- Cuando se diseña el método imprimir tabla de multiplicar se entiende que va a recibir un numero como parámetro, y no necesariamente se tiene que llamar numero_natural.

```
public class Clase_6_Ejemplo_7 {
  public static void main(String[] args){
        obtener numero_natural
        si es numero_natural
              imprimir tabla de multiplicar (numero_natural)
  }
  public static void imprimir tabla de multiplicar (numero){
        definir multiplicador entero
        mientras multiplicador sea menor igual que 10
              imprimir multiplicador x numero
              incrementar multiplicador
  }
}
```

Ejemplo 7: Paramétros

```
/*Escribir un programa que dado un numero ingresado por el usuario, si el numero es natural imprima la tabla de multiplicar de ese numero.
*/
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_6_Ejemplo_7 {
   public static final int MIN = 1;
   public static final int MAXMULTIPLICADOR = 10;
   public static void main(String[] args) {
            int numero_natural;
            BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
            try{
                        System.out.println("Ingrese un numero natural");
                        numero_natural = Integer.valueOf(entrada.readLine());
                        if (numero_natural >= MIN){
                            imprimir_tabla_de_multiplicar (numero_natural);
                        }
            }
            catch (Exception exc){
                        System.out.println(exc);
            }
   }

   public static void imprimir_tabla_de_multiplicar (int numero){
            for (int multiplicador = 1; multiplicador <= MAXMULTIPLICADOR; multiplicador++){
                System.out.println(multiplicador +" * "+numero+" = "+(multiplicador*numero));
            }
   }
}
```

Video de explicación

https://drive.google.com/drive/folders/1wWyFJz2KcuY_c0kUqtvCXyWG0yzZQmGz?usp=sharing

Ejemplo 7

¿Qué hace el programa Clase_6_Ejemplo_7 cuando se ejecuta?

Primero ejecuta el método principal, main, y se ejecutan sentencias que incluyen en el orden en que aparecen.
Define un numero_natural y un buffer entrada
Ingresa al try{}catch{}, imprime el mensaje ingrese numero natural, y el usuario ingresa un numero_natural, supongamos el 8.
Si numero_natural es >0 llama a imprimir_tabla_de_multiplicar (numero_natural)

Empieza en el encabezado de imprimir_tabla_de_multiplicar (int numero).
El tipo del parámetro numero que tiene que coincidir en tipo con el valor que se le paso como parámetro. Ahora tiene una copia del valor que le pasaron desde main a través de la variable numero_natural. Entonces numero tiene un 8.

Se define la constante MAXMULTIPLICADOR (local al método pero podría ser global si la usa otro), y entra al for. Define multiplicador=1.
Pregunta multiplicador<=MAXULTIPLICADOR, en valores sería si (1<=10) es verdadero, entonces imprime un mensaje junto con 1x8=8
Luego ejecuta multiplicador++, por lo tanto vale 2
Por estar dentro de la sentencia for(...){...} se vuelve al comienzo de dicha sentencia
Pregunta multiplicador<=MAXMULTIPLICADOR, en valores sería si (2<=4) es verdadero, entonces Imprime un mensaje junto con 2x8=16
…CONTINUA HASTA QUE multiplicador VALE 11…

Cuando termina el for no hay más sentencias para ejecutar, sale de imprimir_tabla_de_multiplicar y vuelve a la próxima sentencia que sigue de la invocación desde main. Como en main no hay más sentencias el programa termina.

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_6_Ejemplo_7 {
  public static void main(String[] args) {
   int numero_natural;
   BufferedReader entrada = new BufferedReader(new
   InputStreamReader(System.in));
   try{
    System.out.println("Ingrese un numero
             natural");
    numero_natural =
             Integer.valueOf(entrada.readLine());
    if (numero>0){
     imprimir_tabla_de_multiplicar (numero_natural);
    }
   }
   catch (Exception exc){
    System.out.println(exc);
   }
  }

  public static void imprimir_tabla_de_multiplicar
    (int numero){
     final int MAXMULTIPLICADOR = 10;
     for (int multiplicador = 1; multiplicador <=
             MAXMULTIPLICADOR; multiplicador++){
      System.out.println(multiplicador +" *
             "+numero+" = "+(multiplicador*numero));
     }
  }
}
```

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

- En java el pasaje de parámetros es por copia. Se pasan copias de los valores de los parámetros del llamado y dentro del método se trabaja con esos valores asociados a los nombres de las variables figuran en la declaración de parámetros.

Ejemplo 8

```
/* Escribir un programa que dado un numero natural (entero > 0 o >=1) ingresado por el usuario llame un metodo que imprima de forma decreciente
   los numeros naturales menores a ese numero ingresado. Finalmente, imprima el numero natural ingresado
*/
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_6_Ejemplo_8 {
   public static final int MIN = 1;
   public static void main(String[] args) {
            int numero_natural;
            BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
            try{
                        System.out.println("Ingrese un numero natural");
                        numero_natural = Integer.valueOf(entrada.readLine());
                        if (numero_natural >= MIN){
                                    imprimir_decreciente(numero_natural);
                        }
                        System.out.println("Numero natural ingresado: "+numero_natural);
            }
            catch (Exception exc){
                        System.out.println(exc);
            }
   }
   public static void imprimir_decreciente(int numero){
            while (numero >= MIN){
                        System.out.println("El natural es: "+numero);
                        numero--;
            }
   }
}
```

Video de explicación

https://drive.google.com/drive/folders/1wWyFJz2KcuY_c0kUqtvCXyWG0yzZQmGz?usp=sharing

Errores comunes en métodos

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

  public static (tipo no void) nombre_funcion (…){

  …

  return …;

  }

- Una función cuando se invoca solo 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.

Errores comunes en métodos

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

```
public static (tipo no void) nombre_funcion (…){
   …
   If (){
     …
     return …;
   else if (){
     …
     return …;
   }
   //ERROR, hay casos donde la función no retorna nada
}
```

Práctico – tercera parte

5. Escribir un programa que mientras el usuario ingresa un caracter distinto del caracter '*', invoque a un método que imprima si es caracter dígito o caracter letra minúscula, y si es letra minúscula imprimir si es vocal o consonante.

6. Escribir un programa que mientras que el usuario ingrese un número entero natural, llame a un método que calcule la sumatoria entre 1 y dicho numero y se lo retorne como resultado.

7. Escribir un programa que mientras el usuario ingresa un numero de mes (entero) entre 1 y 12 inclusive, muestre por pantalla la cantidad de días del mes ingresado (suponer febrero de 28 días) (***¿mostrar por pantalla la cantidad de días del mes debería realizarse con un método? Debería***).

8. Escribir un programa que mientras que el usuario ingrese un caracter letra minúscula, pida ingresar un numero entero. Si el número ingresado está entre 1 y 5 inclusive deberá imprimir la tabla de multiplicar de dicho numero.

Ejemplo 9: problema con más de un parámetro

- 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.

```
/* el diseño seria
*/
public class Clase_6_Ejemplo_9 {
  public static void main(String[] args){
        obtener numero1 entero
        obtener numero2 entero
        imprimir resultados operaciones matematicas (numero1, numero2)
  }
  public static void imprimir resultados operaciones matematicas (numero1, numero2){
        imprimir numero1 + numero2
        imprimir numero1 - numero2
        imprimir numero1 * numero2
        imprimir numero1 / numero2
  }
}
```

Ejemplo 9

```
/* 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_6_Ejemplo_9 {
   public static void main(String[] args) {
            int numero1;
            int numero2;
            BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
            try{
                        System.out.println("Ingrese un numero entero: ");
                        numero1 = Integer.valueOf(entrada.readLine());
                        System.out.println("Ingrese otro numero 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));
   }
}
```

Ejemplo 10: problema con más de un parámetro

- Realizar un programa que dado dos números enteros y un carácter, todos ingresados por el usuario, muestre por pantalla el resultado de la operación matemática básica considerando el valor del carácter. Si ingreso el caracter:
  - 'a' la suma, 'b' la resta, 'c' la multiplicación y 'd' la división entre ambos números.

```
/* el diseño seria
*/
public class Clase_6_Ejemplo_10 {
   public static void main(String[] args){
            obtener numero1 entero
            obtener numero2 entero
            obtener caracter
            imprimir resultado operacion matematica (carácter, numero1, numero2)
   }
   public static void imprimir resultado operacione matematica (opción, numero1, numero2){
            si la opción es
                        a imprimir numero1 + numero2
                        b imprimir numero1 - numero2
                        c imprimir numero1 * numero2
                        d imprimir numero1 / numero2
   }
}
```

Práctico – cuarta parte

9. Escribir un programa para el Ejemplo 10.

10. Escribir un programa que mientras el usuario ingrese un número entero entre 1 y 10, pida ingresar un caracter, y por cada caracter ingresado imprima:
   - "letra minúscula" si el caracter es una letra del abecedario en minúscula;
   - "letra mayúscula" si el caracter es una letra del abecedario en mayúscula;
   - "dígito" si el caracter corresponde a un carácter número;
   - "otro" para los restantes casos de caracteres.

11. Escribir un programa que mientras el usuario ingrese un número entero entre 1 y 10 realice:
   - Si el numero ingresado es múltiplo de 3 pida ingresar un caracter, y para el caracter ingresado imprima a que tipo de carácter esta asociado:
     - "letra minúscula" si el caracter es una letra del abecedario en minúscula;
     - "letra mayúscula" si el caracter es una letra del abecedario en mayúscula;
     - "dígito" si el caracter corresponde a un carácter número;
     - "otro" para los restantes casos de caracteres.
   - Si el numero ingresado es múltiplo de 5 imprima la tabla de multiplicar del numero ingresado.

Práctico – cuarta parte

12. Siguiendo la Clase 4 Ejercicio 4, hacer un programa completo (usando métodos donde se requiera) en el cual se solicite de teclado el ingreso de un día, un número de mes, y un año; luego calcule la cantidad de días desde la última luna nueva (edad lunar), e informe por pantalla en cual de las 4 fases se corresponde para esa fecha.