	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERÍA** **ESCUELA DE COMPUTACIÓN** **TÉCNICO EN DESARROLLO DE APLICACIONES MÓVILES**
Ciclo I	**Programación Orientada a Objetos** **Guía de Laboratorio No. 5** **Métodos/Funciones**

I. OBJETIVOS.

- Introducir al alumno en el desarrollo de aplicaciones en JAVA.
- Que el alumno conozca los principios básicos de la programación estructurada a partir de los siguientes conceptos:
  - Métodos
  - Funciones

II. INTRODUCCIÓN.

¿Qué es una Función?

Una función es un módulo / sección de código independiente y separado del cuerpo principal, que realiza una tarea específica y que opcionalmente puede regresar un valor a la parte principal del programa u otra función o procedimiento que la llame.

Métodos tipo procedimiento y métodos tipo función

Los procedimientos y las funciones Java son un conjunto de instrucciones definidas dentro de una clase, que realizan una determinada tarea y a las que podemos invocar mediante un nombre.
Cuando se llama a un método o función, la ejecución del programa pasa a éste y cuando éste acaba, la ejecución continúa a partir del punto donde se produjo la llamada.

Utilizando métodos (o procedimientos):

- Podemos construir programas modulares.
- Se consigue la reutilización de código. En lugar de escribir el mismo código repetido cuando se necesite, por ejemplo para validar una fecha, se hace una llamada al método que lo realiza.

En Java un método siempre pertenece a una clase.

Todo programa java tiene un método llamado main. Este método es el punto de entrada al programa y también el punto de salida.

Estructura general de los procedimientos en java.

```
[especificadores] nombreMétodo([lista parámetros]) [throws listaExcepciones]
{
   // instrucciones
}
```

especificadores (opcional): determinan el tipo de acceso al método. Se verán en detalle más adelante.

nombreMetodo: es el nombre que se le da al método. Para crearlo hay que seguir las mismas normas que para crear nombres de variables.

Lista de parámetros (opcional): después del nombre del método y siempre entre paréntesis puede aparecer una lista de parámetros (también llamados argumentos) separados por comas. Estos parámetros son los datos de entrada que recibe el método para operar con ellos. Un método puede recibir cero o más argumentos. Se debe especificar para cada argumento su tipo. Los paréntesis son obligatorios aunque estén vacíos.

throws listaExcepciones (opcional): indica las excepciones que puede generar y manipular el método.

Los métodos no retornan ningún valor

Estructura general de las funciones en java.

```
[especificadores] tipoDevuelto nombreFuncion([lista parámetros]) [throws
listaExcepciones]
{
   // instrucciones
  [return valor;]
}
```

Los elementos que aparecen entre corchetes son opcionales.

especificadores (opcional): determinan el tipo de acceso a la función. Se verán en detalle más adelante.

tipoDevuelto: indica el tipo del valor que devuelve la función. En Java es imprescindible que en la declaración de un método, se indique el tipo de dato que ha de devolver. El dato se devuelve mediante la instrucción return.

nombreFuncion: es el nombre que se le da a la función. Para crearlo hay que seguir las mismas normas que para crear nombres de variables.

Lista de parámetros (opcional): después del nombre de la función y siempre entre paréntesis puede aparecer una lista de parámetros (también llamados argumentos) separados por comas. Estos parámetros son los datos de entrada que recibe la función para operar con ellos. Un método puede recibir cero o más argumentos. Se debe especificar para cada argumento su tipo. Los paréntesis son obligatorios aunque estén vacíos.

throws listaExcepciones (opcional): indica las excepciones que puede generar y manipular el método.

return: se utiliza para devolver un valor. La palabra clave return va seguida de una expresión que será evaluada para saber el valor de retorno. Esta expresión puede ser compleja o puede ser simplemente el nombre de un objeto, una variable de tipo primitivo o una constante.

El tipo del valor de retorno debe coincidir con el tipoDevuelto que se ha indicado en la declaración de la función.

Una función puede devolver un tipo primitivo, un array, un String o un objeto.

Una función tiene un único punto de inicio, representado por la llave de inicio {. La ejecución de una función termina cuando cuando se ejecuta la instrucción return.

La instrucción return puede aparecer en cualquier lugar dentro de la función, no tiene que estar necesariamente al final.

Implementando funciones y métodos en java

Pasos para implementar un método:

1. Describir lo que la función o método debe hacer
2. Determinar las entradas de la función o método
3. Determinar los tipos de las entradas
4. Determinar el tipo del valor retornado (en el caso de las funciones)
5. Escribir las instrucciones que forman el cuerpo de la función o método.

6. Prueba de las funciones y los métodos: diseñar distintos casos de prueba

III. Material y equipo necesario.

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Requerimiento	Cantidad
1	Guía de Laboratorio #4 de PRO	1
2	Computadora con Eclipse IDE	1

IV. Procedimiento.

Crear el proyecto Guia4

Ejemplo01: Ejemplo01.java

Método de tipo función: Comprobar año bisiesto.

```
import java.util.Scanner;


public class Ejemplo01 {

        public static void main (String [] args){


                Scanner sc = new Scanner(System.in);
    int año;
    System.out.print("Introduce año: ");
    año = sc.nextInt();
    if(esBisiesto(año))  //llamada al método
      System.out.println("Bisiesto");
    else
      System.out.println("No es bisiesto");


        }


  /**
   * método que calcula si un año es o no bisiesto
   */
  public static boolean esBisiesto(int a){
    if(a%4==0 && a%100!=0 || a%400==0)
      return true;
```

```
        else
          return false;
    }

}
```

Ejemplo02: Ejemplo02.java
Método que crea una caja de texto.

```
import java.util.Scanner;

public class Ejemplo02 {

        public static void main(String[] args) {
              Scanner sc = new Scanner(System.in);
               String cadena;
               System.out.print("Introduce una cadena de texto: ");
               cadena = sc.nextLine();
               cajaTexto(cadena); //llamada al método


               System.out.print("Introduce otra cadena de texto: ");
               cadena = sc.nextLine();
               cajaTexto(cadena); //llamada al método

        }


   /**
    * método que muestra un String rodeado por un borde
    */
    public static void cajaTexto(String str){
          int n = str.length();
          for (int i = 0; i < n + 4; i++){
              System.out.print("#");
          }
          System.out.println();
          System.out.println("# " + str + " #");
          for (int i = 0; i < n + 4; i++){
              System.out.print("#");
          }
          System.out.println();
    }
}
```

Ejemplo 3: Ejemplo03.java
Tabla de multiplicar. (Importante, debe haber creado el Ejemplo02.java)

```
import java.util.Scanner;

public class Ejemplo03 {

        public static void main(String[] args) {

                Ejemplo02 micajaTexto = new Ejemplo02();

                micajaTexto.cajaTexto("TABLA DE MULTIPLICAR");

           cargarValor();
        }

        public static void cargarValor() {
      Scanner teclado=new Scanner(System.in);
      int valor;
      do {
         System.out.print("\nIngrese valor (0 para salir):");
         valor=teclado.nextInt();
         if (valor>0) {
            calcular(valor);
         }
      } while (valor>0);
   }

   public static void calcular(int v) {
         int contador=1;
      for(int f=v;f<=v*10;f=f+v) {
         System.out.println( v+"x"+contador+"="+f );
         contador++;
      }
   }

}
```

Ejemplo 4: Ejemplo04.java
Funciones

Función que calcular base^{n}.

```
import java.util.Scanner;
```

```
public class Ejemplo04 {

	public static void main(String[] args) {
		Ejemplo02 miCajaTexto = new Ejemplo02();

		miCajaTexto.cajaTexto("Cálculo de potencia");


		Scanner teclado=new Scanner(System.in);

		System.out.print("Ingrese el valor de la base: ");

		int base=teclado.nextInt();


		System.out.print("Ingrese el valor de la potencia: ");

		int potencia=teclado.nextInt();

		miCajaTexto.cajaTexto(String.valueOf( miPotencia(base,potencia) ) );



		miCajaTexto.cajaTexto("Cálculo de área de un cuadrado");

		System.out.print("Ingrese el valor de un Lado del cuadrado: ");

		int lado=teclado.nextInt();

		miCajaTexto.cajaTexto(String.valueOf(miPotencia(lado,2)));

		miCajaTexto.cajaTexto("Un cubo con el mismo valor de lado tiene");

		miCajaTexto.cajaTexto(String.valueOf(miPotencia(lado,3)));
	}

	public static int miPotencia(int base, int potencia){

		if(potencia == 0) return 1;

		if(potencia < 0) {
			System.out.println("No soportado");
			return 0;
		}
```

```
		int resultado=1;

		for (int i=1; i <=potencia; i++)
			resultado=resultado*base;

		return resultado;
	}

}
```

Ejemplo 5:

```
import java.util.Scanner;

public class Ejemplo05 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		Scanner reader = new Scanner(System.in);

		System.out.println(
		"Escriba tres valores de punto flotante, separados por espacios: "
				);

		double numero1 = reader.nextDouble();
		double numero2 = reader.nextDouble();
		double numero3 = reader.nextDouble();

		double valorMaximo=maximo(numero1,numero2,numero3);

		System.out.println("El maximo de los valores es "+valorMaximo);



	}

	public static double maximo(double x, double y, double z){
		//asumiremos de entrada que x es el mayor maximo
		double valorMaximo = x;

		if( y > valorMaximo )
			valorMaximo = y;

		if( z>valorMaximo )
```

```
                        valorMaximo = z;

                return valorMaximo;
        }

}
```

Ejemplo 06:

```
import java.util.Scanner;

public class Ejemplo06 {
        public static void main (String [] args){

                Scanner reader = new Scanner(System.in);


                Scanner reader_double = new Scanner(System.in);

                String continuar="y";
                String opcion="1";

                do{
                        System.out.println("Que desea hacer?");
                        System.out.println("1. Convertir de grados fahrenheit a celsius");
                        System.out.println("2. Convertir de grados celsius a fahrenheit");

                        opcion = reader.nextLine();

                        Double grados = 0.0;

                        if (opcion.equals("1")){
                        System.out.println("Ingrese el valor de los grados fahrenheit");
                                grados=reader_double.nextDouble();
                        System.out.println(grados+" fahrenheit son "+ celsius(grados) +
" grados celsius" );
                        }

                        else if (opcion.equals("2")){
                        System.out.println("Ingrese el valor de los grados celsius");
                        grados=reader_double.nextDouble();
                        System.out.println(grados+" celsius son "+ fahrenheit(grados) +
" grados fahrenheit" );
                        }
                        else{
```

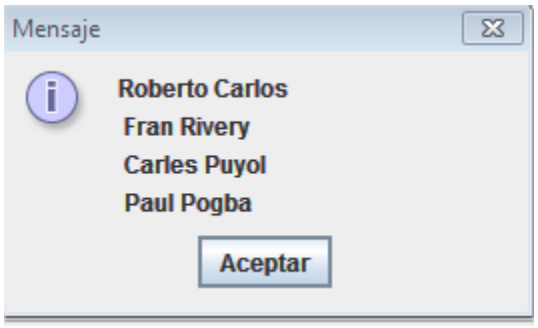
```
                        System.out.println("Opción desconocida");

                        }

                        System.out.println("Desea continuar y/n?");
                        continuar=reader.nextLine();

                }while(continuar.equals("y"));
        }

        private static double fahrenheit(Double p_celsius){

                Double fahrenheit_grad = (p_celsius*9/5)+32;

                return fahrenheit_grad;
        }

private static double celsius(Double p_fahrenheit){

                Double celsius_grad = (p_fahrenheit-32)*5/9;

                return celsius_grad;
        }

}
```

V. EJERCICIOS COMPLEMENTARIOS.

1) Cree una **función** que permita calcular el área de un círculo. Recuerde: πr^2 El valor de PI es constante (3.14159) y el valor del radio será ingresado manualmente.

2) Investigue el uso de la clase Math de Java y realice un ejemplo para potencia y raíz cuadrada.

3) Crear un método que permita concatenar los nombres de varios estudiantes. El ingreso de datos debe ser mediante ingreso de datos gráfico (JOptionPane.showInputDialog) y el resultado debe mostrarse **concatenado (Todos los nombres juntos)** en un mensaje gráfico (showMessageDialog)



4) Crear una función que reciba como parámetro el sueldo de un empleado. Con esta información, imprima las deducciones de ley. Invoque el método con 3 sueldos diferentes en el método main.

VI. REFERENCIA BIBLIOGRAFICA.

http://puntocomnoesunlenguaje.blogspot.com/2012/04/metodos.html

Java : Cómo Programar ./ Paul J. Deitel y Harvey M. Deitel Pauld Deitel
MEXICO, MEXICO : PEARSON, 2012
9a. ed. 2012