

CIFP César Manrique.

Programación 1º de Desarrollo de Aplicaciones Web

Profesor: José David Díaz Díaz

Actividades de la Unidad 4: Funciones

Juan Carlos Francisco Mesa



Esta obra está licenciada bajo la Licencia Creative Commons Atribución 4.0 Internacional.
Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by/4.0/> o
envíe una carta a Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Índice

Actividades.....	1
Desarrollo.....	2
Actividades de comprobación.....	2
1. 4.1. Los parámetros en la llamada a una función en Java pueden ser opcionales si:.....	2
4.2. Una variable local (declarada dentro de una función) puede usarse :.....	2
4.3. El tipo devuelto de todas las funciones definidas en nuestro programa tiene que ser siempre:.....	2
4.4. ¿Qué instrucción permite a una función devolver un valor?.....	3
4.5. La forma de distinguir entre dos o más funciones sobrecargadas es:.....	3
4.6. ¿Cuál es la definición de una función recursiva?.....	3
4.7. El paso de parámetros a una función en Java es siempre:.....	3
4.8. En el caso de que una función devuelva un valor, ¿cuál es la recomendación con respecto a la instrucción return?.....	4
4.9. ¿Cuáles de las siguientes operaciones se pueden implementar fácilmente mediante funciones recursivas?.....	4
4.10. En los identificadores de las funciones, al igual que en los de las variables, se recomienda utilizar la siguiente nomenclatura:.....	4
Actividades de Aplicación.....	5
4.11. Diseña una función que calcule y muestre la superficie y el volumen de una esfera.....	5
4.12. Implementa la función que calcula y devuelve la distancia euclídea que separa los puntos (x1 , y1) y (x2 , y2).....	6
4.13. Crea la función muestraPares(int n) que muestre por consola los primeros n números pares.....	7

4.14. Escribe una función a la que se pase como parámetros de entrada una cantidad de días, horas y minutos. La función calculará y devolverá el número de segundos que existen en los datos de entrada.....	9
4.15. Diseña una función a la que se le pasan las horas y minutos de dos instantes de tiempo, con el siguiente prototipo:.....	10
4.16. Implementa la función divisoresPrimos() que muestra , por consola , todos los divisores primos del número que se le pasa como parámetro.....	11
4.17. Escribe una función que decida si dos números enteros positivos son amigos	12
4.18. Crea una función que muestre por consola una serie de números aleatorios enteros.....	13
4.19. Sobrecarga la función realizada en la Actividad de aplicación 4.18 para que el único parámetro sea la cantidad de números aleatorios que se muestra por consola.....	14

Actividades

Actividades de la Unidad 4: Funciones.

En este documento se detallan las actividades a realizar. Se entregará al profesor en la plataforma digital dos ficheros. Un primer fichero pdf con todas las actividades a realizar, el nombre del fichero será “unidad2 + nombre del alumno.pdf”. Añadir en el fichero pdf por cada actividad de programación dos capturas de pantalla, una del código y otra de su ejecución. También en el fichero pdf copiar todas las preguntas y las respuestas correctas de las actividades de comprobación. Además, entregar un segundo fichero comprimido con todos los códigos fuentes de cada actividad de programación realizada.

Todas las actividades resueltas se deberán de analizar y no se entregarán.

A continuación, detallamos las actividades a realizar:

- **Actividades propuestas.** Esta unidad no tiene dichas actividades.
- **Actividades de comprobación.** Realizarlas todas. Copiar todas las preguntas y sus respuestas correctas.
- **Actividades de aplicación.** Realizar las siguientes **4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.18 y 4.19.**
- **Actividades de ampliación.** No realizar ninguna.

[Volver al índice](#)

Desarrollo

Actividades de comprobación.

1. 4.1. Los parámetros en la llamada a una función en Java pueden ser opcionales si:

- a) Todos los parámetros son del mismo tipo.
- b) Todos los parámetros son de distinto tipo.
- c) Nunca pueden ser opcionales.
- d) Siempre que el tipo devuelto no sea void.

[Volver al índice](#)

4.2. Una variable local (declarada dentro de una función) puede usarse :

- a) En cualquier lugar del código.
- b) Solo dentro de main().
- c) Solo en la función donde se ha declarado.
- d) Ninguna de las opciones anteriores es correcta.

[Volver al índice](#)

4.3. El tipo devuelto de todas las funciones definidas en nuestro programa tiene que ser siempre:

- a) int.
- b) double.
- c) void.
- d) Ninguna de las opciones anteriores es correcta.

[Volver al índice](#)

4.4. ¿Qué instrucción permite a una función devolver un valor?

- a) value.
- b) return.**
- c) static.
- d) function.

[Volver al índice](#)

4.5. La forma de distinguir entre dos o más funciones sobrecargadas es:

- a) Mediante su nombre.
- b) Mediante el tipo devuelto.
- c) Mediante el nombre de sus parámetros.
- d) Mediante su lista de parámetros: número o tipos.**

[Volver al índice](#)

4.6. ¿Cuál es la definición de una función recursiva?

- a) Es aquella que se invoca desde dentro de su propio bloque de instrucciones.**
- b) Es aquella cuyo nombre permite la sobrecarga y además realiza alguna comprobación mediante if.
- c) Es aquella cuyo bloque de instrucciones utiliza alguna sentencia if (lo que llamamos caso base).
- d) Es aquella que genera un bucle infinito.

[Volver al índice](#)

4.7. El paso de parámetros a una función en Java es siempre:

- a) Un paso de parámetros por copia.**
- b) Un paso de parámetros por desplazamiento.
- c) Un paso de parámetros recursivo.
- d) Un paso de parámetros funcional.

[Volver al índice](#)

4.8. En el caso de que una función devuelva un valor, ¿cuál es la recomendación con respecto a la instrucción return?

- a) Utilizar tantos como hagan falta.
- b) Emplear tantos como hagan falta , pero siempre que se encuentren en bloques de instrucciones distintas.
- c) Usar solo uno.
- d) Utilizar solo uno, que será siempre la primera instrucción de la función.

[Volver al índice](#)

4.9. ¿Cuáles de las siguientes operaciones se pueden implementar fácilmente mediante funciones recursivas?

- a) $a^n = a \times a^{n-1}$
- b) esPar (n) = esImpar (n - 1) y esImpar (n) = esPar (n - 1).
- c) suma (a , b) = suma (a + 1 , b - 1).
- d) Todas las respuestas anteriores son correctas.

[Volver al índice](#)

4.10. En los identificadores de las funciones, al igual que en los de las variables, se recomienda utilizar la siguiente nomenclatura:

- a) suma_notas_alumnos().
- b) sumanotasalumnos().
- c) SumaNotasAlumnos().
- d) sumaNotasAlumnos().

[Volver al índice](#)

Actividades de Aplicación.

4.11. Diseña una función que calcule y muestre la superficie y el volumen de una esfera.

$$\text{Superficie} = 4 \pi \cdot \text{radio}^2$$

$$\text{Volumen} = \frac{4 \pi \cdot \text{radio}^3}{3}$$

```
package main;
import java.util.Scanner;

/*
 * @author juancfm
 */
public class Main {
    public static void main(String[] args) {
        double radio;
        String option = "";
        Scanner sc = new Scanner(System.in);

        System.out.print("Por favor indique el radio de la esfera: ");
        radio = sc.nextDouble();
        System.out.println("\Ahora seleccione una opcion para:");
        System.out.println("1 Calcular la superficie o área de la esfera.");
        System.out.println("2 Calcular el volumen de la esfera.");
        System.out.println("3 Calcular ambos.\n");
        System.out.print("Opcion: ");
        option = sc.nextLine();
        option = option.substring(0, 1);
        System.out.println("\n");

        switch (option) {
            case "1" ->
                superficie(radio);
            case "2" ->
                volumen(radio);
            case "3" -> {
                superficie(radio);
                volumen(radio);
            }
        }
    }
}
```

```
    }

    /**
     * Calcula la superficie de la circunferencia a partir de su radio
     *
     * @param radio radio de la circunferencia
     */
    static void superficie(double radio) {
        System.out.println("El área de la circunferencia es: "
            + (4 * Math.PI * Math.pow(radio, 2)));
    }

    /**
     * Calcula el volumen de una esfera a partir de su radio
     *
     * @param radio radio de la esfera
     */
    static void volumen(double radio) {
        System.out.println("El volumen de la esfera es: "
            + ((4 / 3) * Math.PI * Math.pow(radio, 3)));
    }
}
```

Output - Main (run)

```
RUN:
Por favor indique el radio de la esfera: 3
Ahora seleccione una opcion para:
1 Calcular la superficie o área de la esfera.
2 Calcular el volumen de la esfera.
3 Calcular ambos.

Opción: 3

El área de la circunferencia es: 113.09733552923255
El volumen de la esfera es: 84.82300164692441
BUILD SUCCESSFUL (total time: 29 seconds)
```

[Volver al índice](#)

Actividades de la Unidad 4: Funciones

4.12. Implementa la función que calcula y devuelve la distancia euclídea que separa los puntos (x1 , y1) y (x2 , y2).

```
static double distancia (double x1, double y1, double x2, double y2)
```

La fórmula para calcular esta distancia es :

$$\text{distancia} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

```
package main;
import java.util.Scanner;

/*
 * 
 * @author juancfm
 */
public class Main {

    /**
     * 4.12. Implementa la función que calcula y devuelve la distancia euclídea
     * que separa los puntos ( x1 , y1 ) y ( x2 , y2 ).
     */
    public static void main(String[] args) {
        int x1, x2, y1, y2;
        Scanner sc = new Scanner(System.in);

        System.out.println("Para calcular la distancia euclídea entre dos"
                           + " puntos, vamos a ingresar sus coordenadas: ");
        System.out.print("Coordenada x del punto 1: ");
        x1 = sc.nextInt();
        System.out.print("Coordenada y del punto 1: ");
        y1 = sc.nextInt();
        System.out.print("Coordenada x del punto 2: ");
        x2 = sc.nextInt();
        System.out.print("Coordenada y del punto 1: ");
        y2 = sc.nextInt();

        System.out.println("\nLa distancia euclídea entre (" +
                           + x1 + "," + y1 + ") y (" + x2 + "," + y2 + ") es: "
                           + distancia(x1, y1, x2, y2));
    }
}
```

```
/*
 * Función que calcula la distancia euclídea entre dos puntos dados
 *
 * @param x1 abscisa punto 1
 * @param y1 ordenada punto 1
 * @param x2 abscisa punto 2
 * @param y2 ordenada punto 2
 * @return result Devuelve la distancia euclídea entre los dos puntos
 */
public static double distancia(int x1, int y1, int x2, int y2) {
    double result;
    result = Math.sqrt(Math.pow((x1 - x2), 2) + Math.pow((y1 - y2), 2));
    return result;
}
```

```
Output - Main (run) ×
RUN:
Para calcular la distancia euclídea entre dos puntos, vamos a ingresar sus coordenadas:
Coordenada x del punto 1: 3
Coordenada y del punto 1:
5
Coordenada x del punto 2: 2
Coordenada y del punto 1: 3
La distancia euclídea entre (3,5) y (2,3) es: 2.23606797749979
BUILD SUCCESSFUL (total time: 34 seconds)
```

[Volver al índice](#)

4.13. Crea la función muestraPares(int n) que muestre por consola los primeros n números pares.

```
package main;

import java.util.Scanner;

/**
 * 
 * @author juancfm
 */
public class Main {

    /**
     * 4.13. Crea la función muestraPares(int n) que muestre por consola los
     * primeros n números pares.
     */
    public static void main(String[] args) {
        int n;
        Scanner sc = new Scanner( source: System.in);

        System.out.println("Vamos a calcular los primeros n números pares"
                           + " a partir de 0.");

        System.out.print( s: "\n¿Cuántos números desea conocer: ");
        n = sc.nextInt();

        muestraPares(n);
    }

    /**
     * Muestra los primeros n números pares.
     *
     * @param n Cantidad de números pares a mostrar.
     */
    private static void muestraPares(int n) {
        for (int i = 1; i <= n; i++) {
            System.out.println(i + " - " + i * 2);
        }
    }
}
```

Actividades de la Unidad 4: Funciones

```
Output - Main (run) ×
▶ run:
▶ Vamos a calcular los primeros n números pares a partir de 0.

➡ ¿Cuántos números desea conocer: 6
1.-      2
2.-      4
3.-      6
4.-      8
5.-     10
6.-     12
BUILD SUCCESSFUL (total time: 8 seconds)
```

[Volver al índice](#)

Actividades de la Unidad 4: Funciones

4.14. Escribe una función a la que se pase como parámetros de entrada una cantidad de días, horas y minutos. La función calculará y devolverá el número de segundos que existen en los datos de entrada.

```
package main;
import java.util.Scanner;

/*
 * @author juancfm
 */
public class Main {
    /**
     * 4.14. Escribe una función a la que se pase como parámetros de entrada una
     * cantidad de días, horas y minutos. La función calculará y devolverá el
     * número de segundos que existen en los datos de entrada.
     */
    public static void main(String[] args) {
        int d, h, m;
        Scanner sc = new Scanner(source.System.in);

        System.out.println("Vamos a calcular los segundos que hay en los datos "
                           + "proporcionados por el usuario.");
    }
}
```

```
System.out.print( s: "\nDías: ");
d = sc.nextInt();
System.out.print( s: "\nHoras: ");
h = sc.nextInt();
System.out.print( s: "\nMinutos: ");
m = sc.nextInt();

calcularSegundos(d,h,m);

/**
 * Calcula los segundos que hay en los parámetros introducidos
 * @param d Cantidad de días
 * @param h Cantidad de horas
 * @param m Cantidad de segundos
 */
private static void calcularSegundos(int d, int h, int m) {
    int result = ((d * 24 + h) * 60 + m) * 60;
    System.out.println("\nHay " + result + " segundos.");
}

}
```

```
Output - Main (run) ×
run:
Vamos a calcular los segundos que hay en los datos proporcionados por el usuario.

Días: 1
Horas: 12
Minutos: 23

Hay 130980 segundos.
BUILD SUCCESSFUL (total time: 10 seconds)
```

[Volver al índice](#)

Actividades de la Unidad 4: Funciones

4.15. Diseña una función a la que se le pasan las horas y minutos de dos instantes de tiempo, con el siguiente prototipo:

```
static int diferenciaMin ( int horal , int minutol , int hora2 , int minuto2 )
```

La función devolverá la cantidad de minutos que existen de diferencia entre los dos instantes utilizados.

```
package main;
import java.util.Scanner;

/*
 * @author juancfm
 */
public class Main {

    /**
     * 4.15. Diseña una función a la que se le pasan las horas y minutos de dos
     * instantes de tiempo.
     * La función devolverá la cantidad de minutos que existen de diferencia
     * entre los dos instantes utilizados.
     */
    public static void main(String[] args) {
        int h0, h1, m0, m1;
        Scanner sc = new Scanner(source.System.in);

        System.out.println("Vamos a calcular la cantidad de minutos que existen
                           + "de diferencia entre dos instantes utilizados.\n");
        System.out.print("Hora(s) del primer instante: ");
        h0 = sc.nextInt();
        System.out.print("Minuto(s) del primer instante: ");
        m0 = sc.nextInt();
        System.out.print("\nHora(s) del segundo instante: ");
        h1 = sc.nextInt();
        System.out.print("\nMinuto(s) del segundo instante: ");
        m1 = sc.nextInt();
    }
}
```

```
    /**
     * Calcula los minutos de diferencia entre dos instantes dados
     * @param h0 Hora del primer instante
     * @param m0 Minuto del primer instante
     * @param h1 Hora del segundo instante
     * @param m1 Minuto del segundo instante
     */
    private static void calcularMinutos(int h0, int m0, int h1, int m1) {
        int result = Math.abs((h0 * 60 + m0) - (h1 * 60 + m1));
        System.out.println("\nEntre los dos instantes hay " + result
                           + " minutos de diferencia.");
    }
}
```

```
Output - Main (run) ×
run:
Vamos a calcular la cantidad de minutos que existen de diferencia entre dos instantes utilizados.

Hora(s) del primer instante: 15
Minuto(s) del primer instante: 12

Hora(s) del segundo instante: 11
Minuto(s) del segundo instante: 23

Entre los dos instantes hay 229 minutos de diferencia.
BUILD SUCCESSFUL (total time: 24 seconds)
```

[Volver al índice](#)

Actividades de la Unidad 4: Funciones

4.16. Implementa la función divisoresPrimos() que muestra , por consola , todos los divisores primos del número que se le pasa como parámetro.

```
package main;
import java.util.Scanner;

/*
 * @author juancfm
 */
public class Main {

    /**
     * 4.16. Implementa la función divisoresPrimos( ) que muestra, por consola,
     * todos los divisores primos del número que se le pasa como parámetro.
     */
    public static void main(String[] args) {
        int n;
        Scanner sc = new Scanner(System.in);

        System.out.println("Vamos a mostrar todos los números primos que "
            + "dividen a un número dado");
        System.out.print("Indique el número a analizar: ");
        n = sc.nextInt();

        divisoresPrimos(n);
    }

    /**
     * Presenta por consola los números primos divisores de un número
     * @param n Número a analizar
     */
}
```

```
private static void divisoresPrimos(int n) {
    if (esPrimo(n)) {
        System.out.println(n + " es un número primo");
    } else {
        System.out.println("Los divisores primos de " + n + " son:");
        for (int i = 2, aux = n; aux >= 2; aux--) {
            if (esPrimo(n / aux) && n % aux == 0) {
                System.out.println(" " + aux);
            }
        }
    }
}

/**
 * Identifica cuando un número es primo
 * @param n número sujeto a análisis
 * @return Boolean true si es primo
 */
private static boolean esPrimo(int n) {
    boolean flag = true;
    int i = 2;
    while (i <= (int) Math.sqrt(n) && flag == true) {
        if (n % i == 0) {
            flag = false;
        }
        i++;
    }
    return flag;
}
```

Output x
Debugger Console x Main (run) x
run:
Vamos a mostrar todos los números primos que dividen a un número dado
Indique el número a analizar: 1298
Los divisores primos de 1298 son:
59
11
2
BUILD SUCCESSFUL (total time: 5 seconds)

[Volver al índice](#)

Actividades de la Unidad 4: Funciones

4.17. Escribe una función que decida si dos números enteros positivos son amigos .

Dos números a y b son amigos si la suma de los divisores propios (distintos de él mismo) de a es igual a b . Y viceversa . Para probar se pueden usar los números 220 y 284 , que son amigos.

```
package main;

import java.util.Scanner;

/**
 * @author juancfm
 */
public class Main {

    /**
     * 4.17. Escribe una función que decida si dos números enteros positivos son
     * amigos . Dos números a y b son amigos si la suma de los divisores propios
     * (distintos de él mismo) de a es igual a b . Y viceversa . Para probar se
     * pueden usar los números 220 y 284 , que son amigos.
     */
    public static void main(String[] args) {
        int num1, num2;
        Scanner sc = new Scanner(System.in);

        System.out.println("Vamos a mostrar si dos números son amigos.");
        System.out.print("\nIndique el primer número a analizar: ");
        num1 = sc.nextInt();
        System.out.print("\nIndique el siguiente número: ");
        num2 = sc.nextInt();

        System.out.println(
            sonAmigos(num1, num2)
            ? "\nExcelente, Son amigos"
            : "\nLo siento, no son amigos");
    }
}
```

```
/*
 * Compara dos números para ver si son amigos
 * @param num1 primer número a comparar
 * @param num2 segundo número a comparar
 * @return boolean devuelve true si son amigos
 */
private static boolean sonAmigos(int num1, int num2) {
    boolean flag = false;

    if ((sumaDivisores(num2) == num1)
        && (sumaDivisores(num1) == num2)) {
        flag = true;
    }

    return flag;
}

/*
 * Suma los divisores propios de un número aparte de si mismo
 * @param num número bajo análisis
 * @return int devuelve la suma de los divisores propios de un número
 */
private static int sumaDivisores(int num) {
    int suma = 0;

    for (int i = 1; i < num; i++) {
        if (num % i == 0) {
            suma += i;
        }
    }

    return suma;
}
```

Output - Main (run)

```
RUN:
Vamos a mostrar si dos números son amigos.

Indique el primer número a analizar: 284

Indique el siguiente número: 220

Excelente, Son amigos
BUILD SUCCESSFUL (total time: 18 seconds)
```

Output - Main (run)

```
RUN:
Vamos a mostrar si dos números son amigos.

Indique el primer número a analizar: 265

Indique el siguiente número: 468

Lo siento, no son amigos
BUILD SUCCESSFUL (total time: 7 seconds)
```

[Volver al índice](#)

Actividades de la Unidad 4: Funciones

4.18. Crea una función que muestre por consola una serie de números aleatorios enteros.

Los parámetros de la función serán: la cantidad de números aleatorios que se mostrarán y los valores mínimos y máximos que estos pueden tomar.

```
package main;
import java.util.Scanner;

/**
 * @author juancfm
 */
public class Main {
    /**
     * 4.18. Crea una función que muestre por consola una serie de números
     * aleatorios enteros. Los parámetros de la función serán: la cantidad de
     * números aleatorios que se mostrarán y los valores mínimos y máximos que
     * estos pueden tomar.
     */
    public static void main(String[] args) {
        int n, min, max;
        Scanner sc = new Scanner(System.in);

        System.out.println("Vamos a mostrar n números aleatorios.");
        System.out.print("Indique la cantidad de números que desea ver: ");
        n = sc.nextInt();
        System.out.print("Indique el mínimo de los valores que desea: ");
        min = sc.nextInt();
        System.out.print("Indique el máximo de los valores que desea: ");
        max = sc.nextInt();
    }
}
```

```
System.out.println("\nLos valores obtenidos son:");
valoresAleatorios(n, min, max);

/**
 * Obtiene una serie de valores aleatorios comprendidos entre dos números
 * @param n número de elementos a mostrar
 * @param min valor mínimo de la serie
 * @param max valor máximo de la serie
 */
private static void valoresAleatorios(int n, int min, int max) {
    for (int i = 1; i <= n; i++) {
        System.out.println("\n" + i + ".-" + Math.round(Math.random() * (max - min)) + min);
    }
}
```

Output - Main (run) ×

run:
Vamos a mostrar n números aleatorios.
Indique la cantidad de números que desea ver: 5
Indique el mínimo de los valores que desea: 3
Indique el máximo de los valores que desea: 5
Los valores obtenidos son:
1.- 4
2.- 5
3.- 3
4.- 5
5.- 5
BUILD SUCCESSFUL (total time: 15 seconds)

[Volver al índice](#)

Actividades de la Unidad 4: Funciones

4.19. Sobrecarga la función realizada en la Actividad de aplicación 4.18 para que el único parámetro sea la cantidad de números aleatorios que se muestra por consola.

Los números aleatorios serán reales y estarán comprendidos entre 0 y 1.

```
package main;
import java.util.Scanner;

/**
 * @author juancfm
 */
public class Main {

    /**
     * 4.19. Sobrecarga la función realizada en la Actividad de aplicación 4.18
     * para que el único parámetro sea la cantidad de números aleatorios que se
     * muestra por consola. Los números aleatorios serán reales y estarán
     * comprendidos entre 0 y 1.
     */
    public static void main(String[] args) {
        int n, min;
        int max = 0;
        Scanner sc = new Scanner(System.in);
        System.out.println("Vamos a mostrar n números aleatorios.");
        System.out.print("Indique la cantidad de números que desea ver: ");
        n = sc.nextInt();
        System.out.print("Indique el mínimo de los valores que desea: "
                + "(pulse 0 si no desea fijar topes): ");
        min = sc.nextInt();

        if (min != 0) {
            System.out.print("Indique el máximo de los valores que desea: ");
            max = sc.nextInt();
        }

        System.out.println("Los valores obtenidos son: ");
    }
}
```

```
if (min == 0) {
    valoresAleatorios(n);
} else {
    valoresAleatorios(n, min, max);
}

/*
 * Obtiene una serie de valores aleatorios comprendidos entre dos números
 *
 * @param n número de elementos a mostrar
 * @param min valor mínimo de la serie
 * @param max valor máximo de la serie
 */
private static void valoresAleatorios(int n, int min, int max) {
    for (int i = 1; i <= n; i++) {
        System.out.println("\n" + i + ".-" + Math.round(Math.random() * (max - min) + min));
    }
}

/*
 * Sobre carga de la función valoresAleatorios(n, min, max)
 *
 * @param n Cantidad de números a mostrar
 * @see valoresAleatorios(n, min, max)
 */
private static void valoresAleatorios(int n) {
    for (int i = 1; i <= n; i++) {
        System.out.println("\n" + i + ".-" + Math.random());
    }
}
```

```
Output - Main (run) ×
run:
Vamos a mostrar n números aleatorios.

Indique la cantidad de números que desea ver: 5

Indique el mínimo de los valores que desea (pulse 0 si no desea fijar topes): 0

Los valores obtenidos son:

1.-      0.8377921506889878
2.-      0.8681747163409917
3.-      0.6006904223898933
4.-      0.9511745032602433
5.-      0.6199087561574337
BUILD SUCCESSFUL (total time: 11 seconds)
```

Actividades de la Unidad 4: Funciones

```
Output - Main (run) ×
▶ run:
Vamos a mostrar n números aleatorios.

▶ Indique la cantidad de números que desea ver: 5
▶ Indique el mínimo de los valores que desea (pulse 0 si no desea fijar topes): 2
▶ Indique el máximo de los valores que desea: 3
Los valores obtenidos son:

1.- 2
2.- 2
3.- 3
4.- 2
5.- 3
BUILD SUCCESSFUL (total time: 7 seconds)
```

[Volver al índice](#)