FICHA DIDÁCTICA DE LA UNIDAD 02

Bloque	1	**Metodología de la programación**		
UNIDAD	2	**Estructuras de control**	**Nº SESIONES:**	28

Objetivos didácticos específicos de la unidad

9. Utilizar entornos integrados de desarrollo.
10. Escribir programas simples en Java.
11. Escribir llamadas a métodos estáticos.
12. Utilizar parámetros en la llamada a métodos.
13. Incorporar y utilizar librerías de objetos.
14. Utilizar un IDE en la creación y compilación de programas simples en Java.
15. Utilizar estructuras de selección.
16. Utilizar estructuras de repetición.
17. Utilizar estructuras de salto.
18. Escribir código utilizando control de excepciones.
19. Comentar el código.
20. Probar, documentar y depurar programas.

Objetivos generales: j), q), w)	*Capacidades PPS:* i), j), t) , w)

Resultado de aprendizaje

RA 3: escribe y depura código, analizando y utilizando las estructuras de control del lenguaje.
RAT1, RAT2, RAT3

Contenidos didácticos de la unidad

- Utilización de los entornos integrados de desarrollo.
- Codificación, edición y compilación de programas simples orientados a objetos.
- Uso de estructuras de control: selección, repetición y salto.
- Control de excepciones.
- Uso de operaciones de entrada y salida.
- Codificación, edición y compilación de programas con estructuras de control.

Criterios de evaluación

- 19. Se ha escrito y probado el código de un programa que haga uso de estructuras de selección.
- 20. Se han utilizado estructuras de repetición.
- 21. Se han reconocido las posibilidades de las sentencias de salto.
- 22. Se ha escrito código utilizando control de excepciones.
- 23. Se han creado programas utilizando diferentes estructuras de control.
- 24. Se han probado y depurado los programas.
- 25. Se ha comentado y documentado el código.
- 36. Se ha utilizado la consola para realizar operaciones de entrada y salida de información.
- 38. Se han reconocido las posibilidades de entrada/salida del lenguaje y las librerías asociadas.
- RT1, RT2, RT3, RT4

Actividades generales			
• Realización de la reflexión al alumnado de la necesidad de los condicionales y estructuras repetitivas. • Realización y prueba de programas resueltos, identificando los elementos y estructuras del lenguaje. • Realización de programas que resuelvan problemas propuestos, identificando para su resolución la utilización de alguna de las estructuras vistas: de selección simple, multiple o repetitivas. • Realización de programas donde hagan uso de diferentes estructuras de control para la resolución de problemas planteados. • Realización de programas donde se realice control de excepciones.			
GUÍA ACTIVIDADES			
Actividad	**Tiempo**	**¿Cómo? / ¿Qué?**	**CE**
IM: Introducción/motivación	5 min	Reflexión inicial: debate de la necesidad de utilizar estructuras condicionales y repetitivas.	-
T1, T2. Introducción de datos mediante **Scanner**	5 min	Explicación teórico práctica del funcionamiento y componentes a través de los ejemplos.	24, 25
T3, T4. Introducción de datos mediante **JOptionPane**	10 min	Explicación teórico práctica del funcionamiento y componentes a través de los ejemplos.	24, 25
T5. Introducción de datos mediante **BufferedReader**	5 min	Explicación teórico práctica del funcionamiento y componentes a través de los ejemplos.	24, 25
T6. Estructuras básicas de control	5 min	Exposición. Sentencias de selección y de repetición. Qué son y para qué se utilizan.	-
T7. Estructura selección **if** (simple y compuesta)**,** ejemplos **T8** y **T9**	10 min	Análisis y sintáxis de la estructura condicional if,. Explicación del ordinograma asociado para representarlo gráficamente. Demostración de uso a través de ejemplos realizados con el alumnado.	19, 23, 24, 25
T10. Estructura de selección múltiple **switch**, ejemplo **T11**	5 min	Análisis y sintáxis de la estructura condicional switch. Explicación del ordinograma asociado para representarlo gráficamente. Demostración de uso a través de ejemplos realizados con el alumnado.	19, 23, 24, 25
T12. Estructuras de repetición	5 min	Introducción a las distintas estructuras de repetición en Java.	-
T13. Estructura repetitiva **while.** y ejemplo **T14**	10 min	Análisis y sintáxis de la estructura repetitiva while. Explicación del ordinograma asociado para representarlo gráficamente. Demostración de uso a través de ejemplos realizados con el alumnado.	19, 20, 23, 24, 25

T15. Estructura repetitiva **do..while** y ejemplo **T16.**	10 min	Análisis y sintáxis de la estructura repetitiva do while. Explicación del ordinograma asociado para representarlo gráficamente. Demostración de uso a través de ejemplos realizados con el alumnado.	19, 20, 23, 24, 25
T17. Estructura repetitiva **for** y ejemplo **T18.**	5 min	Análisis y sintáxis de la estructura repetitiva for. Explicación del ordinograma asociado para representarlo gráficamente. Demostración de uso a través de ejemplos realizados con el alumnado.	19, 20, 21, 23, 24, 25
T19. Break y **continue**	5min	Explicación de uso.	19, 20, 21, 23, 24, 25
T20. Ejemplo de continue con etiquetas	5 min	Ejemplo teórico/práctico de uso de continue.	19, 20, 21, 23, 24, 25
T21. Ejemplo de break	10 min	Ejemplo teórico / práctico que dibuja pirámide. Uso de break, continue y etiquetas.	19, 20, 21, 23, 24, 25
T22. La sentencia **return**	2 min	Exposición de su utilización.	-
T23. Control de excepciones y ejemplo **T24**	15 min	Explicación de la necesidad del control de excepciones y ejemplo práctico de la división por cero.	19, 20, 21, 22, 23, 24, 25
T25. Prueba y depuración de programas	10 min	Exposición de la prueba y depuración de los programas.	19, 24, 25
T26. Documentación de programas	10 min	Exposición de la documentación necesaria de las aplicaciones.	19, 24, 25
TP27. Ejemplo salida por pantalla System.out.println	5 min	Ejemplo guiado de salida por pantalla utilizando System.out.println.	36, 38
TP28. Ejemplo salida por pantalla JOptionPane.showMessageDialog	5 min	Ejemplo guiado de salida por pantalla utilizando System.out.println.	36, 38
TP29. Ejemplo lectura de datos JOptionPane.showInputDialog	5 min	Ejemplo guiado de salida por pantalla utilizando System.out.println.	36, 38
TP30 y TP31. Análisis de ejemplos completos	35 min	El alumnado analizará y codificará los ejemplos completos. Planteando las dudas y comentando en clase sobre los mismos.	36,38

PRÁCTICA **EJERCICIOS RESUELTOS (**Se dejará unos minutos para que el alumnado realice los ejercicios y se realizará la resolución y explicación en clase conforme al feedback recibido).			
R1. Área de un círculo	10 min	Practicar lectura de datos por teclado, salida por pantalla y estructura secuencial.	24, 25, 36, 38
R2. Dados dos números ver si uno es múltiplo de otro	10 min	Practicar lectura/escritura de información y estructura condicional simple.	19, 23, 34, 25, 36, 38
R3. Ordenar dos números enteros de menor a mayor	10 min	Practicar lectura/escritura de información y estructura condicional simple.	19, 23, 34, 25, 36, 38
R4. Pedir número y mostrar cifras al revés	15 min	Practicar entrada y salidas de datos con estructura secuencial.	24, 25, 36, 38
R5. Lucky number	35 min	Aplicar estructuras de repetición do while y for.	19, 20, 21, 23, 24, 25, 36, 38
R6. Sumar valores introducidos por teclado y decir si la suma es menor, igual o mayor que 0	15 min	Aplicar estructuras de selección y repetición.	19, 20, 21, 23, 24, 25, 36, 38
R7. Número de Armstrong	40 min	Aplicar estructuras de selección y repetición. Explicar las llamadas a métodos.	19, 20, 21, 22, 23, 24, 25, 36, 38
EJERCICIOS PROPUESTOS (quedan 20 sesiones para la realizar los 23 ejercicios). Se indica un tiempo estimado, pero el alumnado irá a su propio ritmo, con supervisión de la docente.			
P1. Decir si un entero es par	10 min	Aplicar estructura condicional simple. E/S datos por pantalla.	19, 24, 25, 36, 38
P2. Mayor de dos enteros	10 min		
P3. Decir si un número es múltiplo de otro	10 min		
P4. Decir si un número es múltiplo de 10, si lo es, pedir otro número y decir si también lo es	10 min	Aplicar estructura condicional simple. E/S datos por pantalla.	19, 22, 23, 24, 25, 36, 38
P5. Multiplicar dos números e informar si es cero u otra cosa	5 min	Aplicar estructura condicional simple. E/S datos por pantalla.	
P6. Dividir dos números si el segundo no es cero	5 min	Aplicar estructura condicional simple. E/S datos por pantalla.	
P7. Decir si entero es múltiplo de 2 o de 3	10 min	Aplicar estructura condicional simple. E/S datos por pantalla.	
P8. Decir si entero es múltiplo de 2 y de 3 simultáneamente	5 min	Aplicar estructura condicional simple. E/S datos por pantalla.	

P9. Decir si entero es múltiplo de 2 pero no de 3	5 min	Aplicar estructuras condicionales adecuadas a la resolución de cada problema.	19, 22, 23, 24, 25, 36, 38
P10. Decir si entero es múltiplo de ni de 2 ni de 3	5 min		
P11. Decir si dos enteros son pares	5 min		
P12. Decir si al menos uno de dos números es par	5 min		
P13. Decir si uno y sólo un número de los dos introducidos es par	5 min		
P14. Dados dos enteros decir si uno, los dos o ninguno es positivo	10 min		
P15. Mayor de tres números	15 min		
P16. Dados dos números decir si son iguales, sino, decir el mayor	5 min		
P17. Dados tres números, ordenar de mayor a menor	30 min		
P18. Notas en formato letras	40 min		
P19. Dado dia, mes, anio. Decir si fecha correcta	40 min		
P20. Pedir número de 1 a 99 y escribirlo en letra	20 min		
P21. Transformar un bucle for dado en un bucle while	5 min	Aplicar estructuras de selección y repetición.	19,20, 21, 23, 24, 25
P22. Triángulo con asteriscos alineado a la izquierda	30 min		
P23. Mostrar los cinco primeros números pares	10 min		
P24. Tablas multiplicar del 1 al 10	30 min		
P25. Pasar número de decimal a romano	1,2 sesiones		
P26. Pirámide de n asteriscos alineada a la derecha	1 sesión		
TEST/CUESTIONARIO	45 min	Test/ Cuestionario de la unidad.	19, 20, 21, 23, 24, 25
ACTIVIDADES DE AMPLIACIÓN (opcionales)			
ISA DESAFÍA: ¡TE RETO!	-	Actividades de motivación y diversión. Toma de contacto con el juez del concurso de programación *Programame.*	

Cuestionario Unidad 02

1. **¿Qué diferencia hay entre bucle while y un bucle for?:**
   a) El bucle for puede no llegar a ejecutarse nunca pero el while siempre se ejecuta al menos una vez.
   b) El bucle for se ejecuta un número determinado de veces y el while un número indeterminado de veces.
   c) El bucle for no puede convertirse en un bucle while pero sí al contrario.
   d) El bucle while permite su incialización pero el bucle for no.

2. **De acuerdo a la sintaxis del bucle while:**

```
while (condición) {
 sentencias
 }
```

   ¿Qué es falso en relación a la condición?
   a) La condición es una variable booleana.
   b) La condición sólo se evalúa al principio de la ejecución del bucle.
   c) Si la condición es verdadera, se ejecuta el bloque de sentencias, y se vuelve al principio del bucle.
   d) Si la condición es falsa, no se ejecuta el bloque de sentencias.

3. **Dado el siguiente código escrito en Java, que imprime los números múltiplos de cuatro menores o iguales que 100:**

```
int x = 0;
        while (x++ <= 100){
            System.out.print(x);
            if (x%4 == 0) System.out.print(" es múltiplo de cuatro");
           System.out.println();
        }
```

   Programa un código equivalente que use un bucle do-while.

4. **La famosa SERIE DE FIBONACCI se construye de tal manera que cada término de la misma es igual a la suma de los dos anteriores. Es decir, si comenzamos con el número 1, esta serie tiene la siguente apariencia:**

   1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4081, 6665, 10746, ...

ⓘ Nota

Por cierto, esta serie tiene una serie de propiedades tales como que los números consecutivos de Fibonacci son primos entre si, aunque la más curiosa de todas es que el cociente de dos números consecutivos de la serie se aproxima a un número "Fi", 1.618033988 .... Durante los últimos siglos se ha venido considerando que el número Fi , también llamado "divina proporción" o "razón áurea", era un baremo de equilibrio y belleza en cuanto lo que a proporciones se refiere.

Programa un código que imprima por pantalla la serie de números de Fibonacci menores de 1000

5. ¿Qué es una variable centinela?

a) Es una variable de tipo entero (int) que permite contar el número de veces que se ejecuta el bucle.

b) No existen este tipo de variables.

c) Es un objeto de la clase "Guard" que tiene métodos para controlar bucles.

d) Es una variable booleana (boolean) a la que asignaremos valores y que nos ayudará en el control del bucle.

6. De acuerdo a la sintaxis del bucle do-while:

```
do{
   sentencias
} while (condición)
```

Señalar cuál es la afirmación falsa:

a) Si condición = = true, entonces el bucle se sigue ejecutando.

b) Aunque se cumpla condición = = false, el bucle se llega a ejecutar alguna vez.

c) Si condición = = false, el bucle no se llega a ejecutar nunca.

d) Ninguna de las anteriores es falsa.

7. Transforma este bucle for:

```
for( a=0,b=0; a < 7; a++,b+=2 )
```

en su equivalente bucle while.

8. De acuerdo a la sintaxis del bucle for

```
for (inicialización ; condición ; actualización) {
 sentencias
}
```

Señalar cuál es la afirmación falsa:

a) La inicialización se realiza cada vez antes de entrar al bucle.

b) La condición se comprueba cada vez antes de entrar al bucle.

c) La actualización se realiza siempre al terminar de ejecutar la iteración.

d) La inicialización, condición y actualización son elementos "opcionales".

9. ¿Cuál de estas características de las sentencias "break" y "continue" es "falsa"?

a) La sentencia break permite salirnos de un bucle que está en ejecución, sin embargo, continue sigue ejecutando las siguientes iteraciones del bucle.

b) Las sentecias break y continue pueden usarse en tanto en sentencias de repetición (bucles) como de bifurcación (switch).

c) Después de ejecutarse la sentencia continue en un bucle, se vuelve a evaluar la condición del bucle.

d) La sentencia break termina la ejecución de una sentencia de bifurcación (switch).

10. ¿ Cuántas veces se ejecuta el cuerpo del siguiente bucle ?

```
int i = 0;
while (true) {
   i ++;
   if (i <10)
      continue;
   i ++;
   if (i== 10)
      break;
}
```

a) 10.

b) El bucle se ejecuta permanentemente (no sale).

c) 0.

d) 11.

FICHA DIDÁCTICA DE LA UNIDAD 03

Bloque	1	**Metodología de la programación**		
UNIDAD	3	**Estructuras de datos estáticas: arrays, Strings**	**Nº SESIONES:**	30

Objetivos didácticos específicos de la unidad

21. Conocer el concepto de vector o array.
22. Estudiar el concepto de array multidimensional o matriz.
23. Utilizar arrays en la resolución de problemas sencillos.
24. Utilizar las cadenas de caracteres.
25. Conocer el uso de expresiones regulares en la búsqueda de patrones de texto.
26. Conocer e investigar el funcionamiento de algoritmos de ordenación de vectores.

Objetivos generales: j), q), w)	*Capacidades PPS:* a), i) j), t) , w)

Resultado de aprendizaje

RA 6: escribe programas que manipulen información seleccionando y utilizando tipos avanzados de datos. RAT1, RAT2, RAT3

Contenidos didácticos de la unidad

- Creación de arrays.
- Inicialización de arrays.
- Utilización de arrays.
- Arrays multidimensionales.
- Cadenas de caracteres.
- Búsqueda de datos dentro de un array.
- Expresiones regulares en la búsqueda de patrones de texto.
- Hashtables.
- Algoritmos de ordenación.

Criterios de evaluación
• **44.** Se han escrito programas que utilicen arrays. • **45.** Se han reconocido las librerías de clases relacionadas con tipos de datos avanzados. • **46.** Se han utilizado listas para almacenar y procesar información. • **47.** Se han utilizado iteradores para recorrer los elementos de las listas. • **48.** Se han reconocido las características y ventajas de cada una de la colecciones de datos disponibles. • **50.** Se han utilizado expresiones regulares en la búsqueda de patrones en cadenas de texto. • RT1, RT2, RT3, RT4
Actividades generales
• Realización al inicio de la Unidad de un Brainstorming sobre "¿Qué es un vector?" • Exposición oral de los contenidos conceptuales de la unidad. • Dibujar gráficamente todas las estructuras estáticas de datos vistas en la unidad, para comprender mejor su significado, y sus operaciones. • Elección de las estructuras correctas para la resolución de los problemas planteados. • Diseño de programas en Java que resuelvan problemas mediante la utilización de arrays. • Diseño de programas que resuelvan problemas mediante la utilización de matrices. • Discusión y puesta en común de las diferentes soluciones aportadas a los ejercicios.

GUÍA ACTIVIDADES

Actividad	Tiempo	¿Cómo? / ¿Qué?	CE
IM: Introducción/motivación	5 min	Tormenta de ideas: ¿qué es un vector?	-
T1. Concepto array. Instancia. Almacenamiento	15 min	Exposición concepto array, cómo se declaran. Cómo se almacenan en memoria.	44, 45
TP2. Declaración, instanciación y reccorrido de un array	10 min	Realización guiada y explicación ejemplo declaración, instanciación y recorrido.	44, 45, 47
TP3. Calcular media de un vector de enteros	10 min	Propuesta y resolución de ejemplo que calcula media de los números almacenados en un vector.	44, 45, 47
T4. Copia de vectores recorriendo array y **TP5**. Uso de System.arrayCopy	15 min	Explicación de copia de vectores recorriendo el array. Uso de System. arrayCopy para copiar un vector en otro.	44, 45, 47, 48
TP6. Ejemplo métodos clone e equals	10 min	Estudio de clone e equals; comprobar la diferencia entre =, equals.	44, 45, 47, 48
T7. Arrays multidimensionales. Concepto. Declaración	5 min	Explicación del concepto y declaración de una matríz.	44, 45
TP8. Ejemplo matriz 3x3	10 min	Realización de ejemplo guiado de declaración e inicialización a la vez de matriz 3x3. Recorrido.	44, 45, 47, 48
TP9. Matriz con filas de tamaño diferente	10 min	Realización, análisis y explicación del ejemplo.	44, 45, 47

T10. La clase Arrays. Métodos asociados: fill, equals, sort, binarySearch	20 min	Exposición de la clase Arrays y alguno de sus métodos asociados a través de mini ejemplos.	44, 45, 47, 48
T11. La clase String	5 min	Exposición del concepto de cadena. La clase String.	44, 45
T12. Métodos de la clase String. Expresiones regulares	40 min	Explicación de algunos métodos a través de mini ejemplos:String, length, concat, toString, compareTo, equals, toLowerCase, toUpperCase, trim, replace, substring, startsWith, endsWith, charAt, indexOf, toCharArray, valueOf, formt, matches. Expresiones regulares.	44, 45, 46, 47, 50
T13. La clase StringTokenizer	20 min	Realización de ejemplo para dividir una cadena en elementos independientes.	44, 45, 47, 48
T14. La clase HashMap	5 min	Exposición teórica de la colección HashMap.	44, 45, 46, 47, 48
TP15. Ejemplo de uso detallado HashMap. Diccionario biligue	20 min	Realización y explicación de un ejemplo guiado. Declaración. Detallar el uso de values, keySet, get,contains , getOrDefault, remove.	44, 45, 46, 47, 48
T16. Búsqueda dicotómica en vector ordenado	5 min	Exposición del clásico método de búsqueda binaria o dicotómica.	44, 45, 47
TP17. Ejemplo búsqueda dicotómica	15 min	Propuesta y realización del ejemplo de búsqueda binaria.	44, 45, 47
T18. Ordenación de vectores	5 min	Indicación de los métodos clásicos que existen para ordenar vectores.	44, 45, 47
TP19. Método de la burbuja	20 min	Explicación, resolución gráfica mediante traza y codificación guiada.	44, 45, 47
PRÁCTICA EJERCICIOS RESUELTOS (Se dejará unos minutos para que el alumnado realice los ejercicios y se realizará la resolución y explicación en clase conforme al feedback recibido**).**			
R1. Calcular la media de temperaturas	25 min	Practicar lectura de datos de un vector y recorrido.	44, 45, 47
R2. Pedir datos y mostrar array	5 min	Practicar la lectura y recorrido de un array de 10 enteros.	44, 45, 47
R3. Utilizar expresiones regulares	25 min	Práctica guiada de utilización de expresiones regulares filtrando una fecha.	44, 45, 46, 47, 50
R4. Utilizar expresiones regulares	20 min	Aplicar expresiones regulares para introducir correctamente un dni.	44, 45, 46, 47, 50
R5. Dado vector de 5 enteros contar cuantos son positivos, negativos o cero	15 min	Practicar recorridos de vector y realizar operaciones con los datos obtenidos.	44, 45, 47

R6. Mezclar dos arrays	25 min	Practicar el uso y recorrido de tablas o matrices.	44, 45, 47
R7. Vector de Strings	15 min	Practicar el uso y recorrido de vectores utilizando datos tipo String.	44, 45, 47
EJERCICIOS PROPUESTOS			
PRÁCTICA (quedan unas 19 sesiones para la realizar los 25 ejercicios). Se indica un tiempo estimado, pero el alumnado irá a su propio ritmo, con supervisión de la docente.			
BÁSICOS			
P1. Usando arraycopy para copiar un vector en otro	10 min	Realizar de nuevo el ejemplo de T4: copiando un array en otro utilizando el método arraycopy; pero utiliza un bucle for para mostrar la información por pantalla.	44, 45, 47, 48
P2. Pedir diez números reales almacenar en un array, y recorrerlo. Averiguar el máximo y mínimo para mostrarlos por pantalla	20 min	Crear vector de reales y recorrerlo realizando cálculos y mostrando el resultado por pantalla.	44, 45, 47, 48
P3. Leer 5 números y mostrarlos en el mismo orden introducido	10 min	Lectura y recorrido de vector de enteros.	44, 45, 47, 48
P4. Leer veinte números enteros por teclado, almacenar en un array y luego mostrar por separado la suma de todos los valores positivos y negativos	20 min	Crear vector de reales y recorrerlo realizando cálculos y mostrando el resultado por pantalla.	44, 45, 47, 48
P5. Leer 10 números enteros. Debemos mostrarlos en el siguiente orden: el primero, el último, el segundo, el penúltimo, el tercero, etc.	30 min	Manejar arrays unidimensionales.	44, 45, 47, 48
AVANZADOS			
P6. Leer los datos de dos tablas de 12 elementos numéricos, y mezclarlos en una tercera de la forma: 3 de la tabla A, 3 de la B, otros 3 de A, otros 3 de la B, etc	20 min	Leer los datos de dos tablas de 12 elementos numéricos, y mezclarlos en una tercera según criterios.	44, 45, 47, 48
P7. Leer por teclado una serie de 10 números enteros. La aplicación debe indicarnos si los números están ordenados de forma creciente, decreciente, o si están desordenados	20 min	Dada una serie de números indicar si los números están ordenados de forma creciente, decreciente, o si están desordenados.	44, 45, 47, 48
P8. Crea un programa que cree un array de enteros de tamaño 100 y lo rellene con valores enteros aleatorios entre 1 y 10 (utiliza 1 + Math.random()*10). Luego pedirá un valor N y mostrará en qué posiciones del array aparece N	20 min	Generar array con números aleatorios y mostrar en qué posiciones aparece un valor determinado.	44, 45, 47, 48

P9. Aplicación que declare una tabla de 10 elementos enteros. Leer mediante el teclado 8 números. Después se debe pedir un número y una posición, insertarlo en la posición indicada, desplazando los que estén detrás	35 min	Inserción de elementos en un vector desplazando el resto.	44, 45, 47, 48
P10. String por teclado. Por la salida queremos que nos diga cuántas letras tiene (contando espacios en blanco si es que los tiene)	10 min	Manejo de Strings, contar elementos incluyendo los espacios en blanco. Uso de length.	44, 45, 47
P11. String por teclado. Por la salida queremos que nos diga cuántas letras tiene contando espacios en blanco	10 min	Manejo y recorrido de Strings, contar elementos incluyendo los espacios en blanco. Uso de charAt.	44, 45, 47
P12. String por teclado. Por la salida queremos que nos diga cuántas letras 'a' tiene	10 min	Manejo y recorrido de Strings, contar elementos incluyendo los espacios en blanco. Uso de charAt.	44, 45, 47
P13. Aceptar varios números por teclado hasta llegar a un 0, cuando llegue a un 0 no pedirá más y nos mostrará por pantalla el mayor de los números introducidos (como máximo 10, contando el 0)	15 min	Manejo y recorrido de Strings, contar elementos incluyendo los espacios en blanco. Uso de charAt.	44, 45, 47
P14. Hacer un programa que acepte inicialice un vector de Strings y declare cuatro marcas de coches. Recorrer el vector utilizando el bucle for-each para indicar las marcas de coches almacenadas	10 min	Inialización de vector de Strings y recorrido utilizando iterador for-each.	44, 45, 47
P15. Hacer un programa que simule un torneo de baloncesto	30 min	Aplicación de uso de arrays.	44, 45, 47,
P16. Array de enteros de tamaño 100 y lo rellene con valores enteros aleatorios entre 1 y 10 (utiliza 1 + Math. random()*10). Luego pedirá un valor N y mostrará en qué posiciones del array aparece N	20 min	Aplicación uso de arrays.	44, 45, 47
P17. Palíndromo	20 min	Uso y recorridos de Strings. CharAt, toUpper.	44, 45, 47
P18. Crear una tabla bidimensional 5x5 y rellenarla de la siguiente forma: la posición T[n,m] debe contener n+m. Después se debe mostrar su contenido	20 min	Utilización, recorrido y operaciones con datos de matrices.	44, 45, 47

P19. Introducir NxM valores. Recorrer la matriz y al final mostrar por pantalla cuántos valores son mayores que cero, cuántos son menores que cero y cuántos son igual a cero	20 min	Utilización, recorrido y operaciones con datos de matrices.	44, 45, 47
P20. Tabla de tamaño 4x4. Decir si es simétrica o no, es decir si se obtiene la misma tabla al cambiar las filas por columnas	30 min	Utilización, recorrido y operaciones con matrices.	44, 45, 47
P21. Crear y cargar dos matrices de tamaño 3x3, sumarlas y mostrar su suma	20 min	Uso y recorrido. Operaciones con matrices.	44, 45, 47
P22. Crear una matriz "marco" de tamaño 8x6: todos sus elementos deben ser 0 salvo los de los bordes que deben ser 1. Mostrarla	20 min	Uso y recorrido. Operaciones con matrices.	44, 45, 47
P23. Crear una tabla de tamaño 7x7 y rellenarla de forma que sea una matriz identidad	15 min	Operaciones con matrices.	44, 45, 47
P24. Crear y cargar una tabla de tamaño 10x10, mostrar la suma de cada fila y de cada columna	20 min	Utilización, recorrido y operaciones con matrices.	44, 45, 47
P25. Utilizando dos tablas de tamaño 5x9 y 9x5, cargar la primera y trasponerla en la segunda	20 min	Operaciones con matrices. Uso y recorrido.	44, 45, 47
ACTIVIDADES DE AMPLIACIÓN (opcionales)			
ISA DESAFÍA: ¡TE RETO!	-	Actividades de motivación y diversión. Toma de contacto con el juez del concurso de programación *Programame.*	-
TEST/CUESTIONARIO	45 min	Test/ Cuestionario de la unidad.	44, 45, 46, 47, 48

Test Unidad 03

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

   int[] arreglo={32,52,37,12,32,5};

   System.out.printf ( "%s %8s\n", "Indice", "Valor" );

   for ( int contador = 0; contador < arreglo.length; contador++ )
   {
   System.out.printf ("%5d %8d\n", contador, arreglo[ contador ] );
   }
 }
```

1. **Del anterior código de java, podemos afirmar que:**
   - ○ a. El código lee los valores del arreglo.
   - ○ b. Imprime en pantalla los valores del arreglo.
   - ○ c. Imprime en pantalla del valor del arreglo con su respectivo índice.
   - ○ d. Imprime en pantalla el tamaño del arreglo.

```
public class Ejercicios {

        public static void main( String[] args )
        {
        int total=0;

        int[] num={32,52,37,12,32,5};


        for ( int i = 0; i < num.length; i++ ){

        total += num[i]        ;

        }
        System.out.printf ("la suma de los elementos del arreglo es: "+ i )       ;

        }
}
```

2. **El código anterior realiza la función de sumar los valores que tiene el arreglo, imprimiendo el resultado de la suma de dichos valores, el código no funciona debido a que tiene un error en:**

```
public class Ejercicios {

	public static void main( String[] args )
	{
	int total=0;

	int[] num={32,52,37,12,32,5};


	for ( int i = 0; i < num.length; i++ ){

	total += num[i]	;

	}
	System.out.printf ("la suma de los elementos del arreglo es: "+ i )	;

	}
}
```

El código anterior realiza la función de sumar los valores que tiene el array, imprimiendo el resultado de la suma de dichos valores, el código no funciona debido a que tiene un error en:

- a. El índice del for debe sumar +2.
- b. El código funciona correctamente, no tiene errores.
- c. Al imprimir en pantalla debe concatenar la variable total y no el índice.
- d. La función de la variable total está mal declarada.

```
public class Ejercicios {

	public static void main( String[] args )
	{
		int[][] matriz = new int[3][3] ;

		matriz[0][0]=2;
		matriz[0][1]=3;
		matriz[0][2]=5;
		matriz[1][0]=4;
		matriz[1][1]=6;
		matriz[1][2]=6;
		matriz[2][0]=3;
		matriz[2][1]=3;
		matriz[2][2]=3;

		for(int i=0;i < 3;i++){
			System.out.println();

			for(int j=0;j <=i;i++){
				System.out.print(matriz[i] [j]+" ");
			}
		}
	}
}
```

3. Del código anterior afirmamos que:

- ○ a. Imprime la matriz de 3x3 haciendo un recorrido total.
- ○ b. Imprime la matriz 3x3 haciendo un recorrido parcial.
- ○ c. Imprime la diagonal de la matriz haciendo un recorrido parcial.
- ○ d. Imprime solo los valores que están debajo de la diagonal de la matriz incluyendo la diagonal.

```
public class Ejercicios {

        public static void main( String[] args )
        {









        }
}
```

4. Del código anterior se requiere crear un arreglo que tenga 10 elementos de números decimales, la línea de código correspondiente para satisfacer el código anterior es:

- ○ a. double num [] =new double[10];
- ○ b. int num []=new double [10];
- ○ c. int num []={3.3, 4, 5, 5.7, 3.4, 5, 6, 7, 4, 2};
- ○ d. double num[] ={3.2, 4.2, 5.3, 4.2, 5.2}

```
public class Examen
{
        public static void main(String[] args)
        {
                int[][] arreglo1={{1, 2, 3}, {4, 5, 6}};

                System.out.println( "Los valores en arreglo1 por filas son" );
                imprimirArreglo( arreglo1 );
        }
        public static void imprimirArreglo( int[][] arreglo )
        {

        for ( int fila = 0; fila < arreglo.length; fila++ )
        {

        for ( int columna = 0; columna < arreglo[ fila ] .length; columna++ ){
        System.out.printf (" %d ", arreglo[ fila ]  [ columna ] );
        }
        System.out.println( );


 }
 }
}
```

5. El resultado que muestra en pantalla el código anterior es:

a) 1 2 3
4 5 6

b) 1 2 3 4 5 6

c) No imprime nada.

d) 1 2
3 4
5 6

```
public class Ejercicios {

    public static void main( String[] args )
    {
        int[][] matriz = {{3,1,3},{4,5,6},{5,4,2}};

        for(int i=0;i < 3;i++){
            System.out.println();

            for(int j=0;j <3;j++){
            if(i==j){
                matriz[0][1]=0;
                matriz[0][2]=0;
                matriz[1][0]=0;
                matriz[1][2]=0;
                matriz[2][1]=0;
                matriz[2][0]=0;
            }
            System.out.print(matriz[i][j]+" ");
            }
        }
    }
}
```

6. Del código anterior, la función que realiza el if es para:

- ○ a. Convertir la matriz cuadrada en una matriz de 3x2.
- ○ b. Convertir todos los elementos de la matriz en 0.
- ○ c. Convertir la matriz en una matriz diagonal.
- ○ d. Obtener la inversa de la matriz.

7. Teniendo un array de números enteros de calificaciones de 1 a 10, el tamaño del array es de 20. Si se desea conocer si alguno de los 20 estudiantes obtuvo una nota de 10 se requiere:

- ○ a. Un patrón de recorrido total.
- ○ b. Un patrón de recorrido parcial.
- ○ c. Un patrón de doble recorrido.
- ○ d. Un for anidado.

```
public class Ejercicios {

        public static void main( String[] args )
        {
                int[][] matriz = new int[3][3] ;

                matriz[0][0]=2;
                matriz[0][1]=3;
                matriz[0][2]=5;
                matriz[1][0]=4;
                matriz[1][1]=6;
                matriz[1][2]=6;
                matriz[2][0]=3;
                matriz[2][1]=3;
                matriz[2][2]=3;

                for(int i=0;i < 3;i++){
                        System.out.println();

                        for(int j=0;j <3;j++){
                                System.out.print(matriz[i] [j]+" ");
                        }
                }
        }
}
```

8. Del código anterior podemos deducir que:

- ○ a. Imprime una matriz cuadrada de 2x2
- ○ b. Imprime los valores de los arreglos.
- ○ c. Imprime la matriz de 3x3 con sus respectivos valores.

```
public class Ejercicios {

        public static void main( String[] args )
        {
                int [] matriz=new int[5];
                matriz[0]=5;
                matriz[1]=4;
                matriz[2]=3;
                matriz[3]=2;
                matriz[4]=1;

                for(int i =0;i<5;i++){
                        System.out.println("valor del indice "+i+" = "+matriz[i]);
                }


        }


}
```

9. Una forma para crear el array del código anterior en una sola línea de código es la siguiente:

- a. int [] matriz={5,4,3,2,1};
- b. String [] matriz={1,2,3,4,5};
- c. int () matriz={5,4,3,2,1};
- d. int [] matriz={1,2,3,4,5};

```
public class Examen {
        public final  static int TAMAÑO=5;

        public static void main(String[] args) {


          int [ ] contenedor= new int[TAMAÑO];

          contenedor [0]=1;
          contenedor [0]=98;
          contenedor [1]=44;
          contenedor [2]=12;
          contenedor [3]=13;

          for(int i=0; i<contenedor.length;i++){

                }

              System.out.println[contenedor[0]);
        }

}
```

10. Del código anterior podemos afirmar que:

- a. El código tiene un error, ya que se repite el contenedor[0].
- b. El programa imprime 5 valores numéricos.
- c. El valor 1 del contenedor [0] no se imprime, solo imprime el valor 98
- d. El código imprime los números 98, 44 ,12 ,13

FICHA DIDÁCTICA DE LA UNIDAD 04

Bloque	1	**Metodología de la programación**		
UNIDAD	4	**Métodos y funciones. Recursividad**	**Nº SESIONES:**	22

Objetivos didácticos específicos de la unidad

27. Crear métodos o funciones en Java.
28. Pasar parámetros a métodos.
29. Diferenciar entre parámetro por valor y por referencia.
30. Conocer y utilizar variables globales o locales.
31. Crear bibliotecas a través de paquetes.
32. Conocer el concepto de recursividad.
33. Depurar programas recursivos y analizarlos.

Objetivos generales: j),q), w)	*Capacidades PPS:* a), j), t , w)

Resultado de aprendizaje

RA 2: escribe y prueba programas sencillos, reconociendo y aplicando los fundamentos de la programación orientada a objetos.
RA 3: escribe y depura código, analizando y utilizando las estructuras de control del lenguaje.
RA 6: escribe programas que manipulen información seleccionando y utilizando tipos avanzados de datos.
RAT1, RAT2, RAT3

Contenidos didácticos de la unidad

- Utilización y creación de métodos.
- Creación de bibliotecas de rutinas.
- Codificación, edición, compilación y depuración de programas recursivos.

Criterios de evaluación

- **14.** Se han escrito llamadas a métodos estáticos.
- **15.** Se han utilizado parámetros en la llamada a métodos.
- **18.** Se ha utilizado el entorno integrado de desarrollo en la creación y compilación de programas recursivos.
- **19.** Se han resuelto problemas usando recursividad.
- **20.** Se han creado bibliotecas a través de paquetes.
- RT1, RT2, RT3, RT4

Actividades generales

- Introducción/motivación. Debate de por qué es necesario crear funciones o librerías de funciones.
- Exposición oral de los contenidos conceptuales de la unidad.
- Aprender la sintaxis de los métodos y las funciones.
- Crear métodos y funciones sencillos.
- Crear funciones con paso de parámetros por valor.
- Crear funciones con paso de parámetros por referencia.
- Utilizar funciones pasando parámetros tanto por valor como por referencia.
- Realizar programas recursivos sencillos.
- Depurar y analizar programas recursivos utilizando el IDE.
- Diseño de programas que resuelvan problemas mediante la utilización de funciones.
- Discusión y puesta en común de las diferentes soluciones aportadas a los ejercicios.
- Cuestionario de la unidad.

GUÍA ACTIVIDADES			
Actividad	**Tiempo**	**¿Cómo? / ¿Qué?**	**CE**
IM: Introducción/motivación	5 min	Debate de por qué es necesario crear funciones o librerías de funciones.	-
T1. Modularidad. Funciones y procedimientos	10 min	Exposición teórica de qué es la modularidad. Tipos de módulos: funciones y procedimientos.	-
TP2. Estudio ejemplo definición y creación de métodos	10 min	Ejemplo teórico/práctico sencillo de suma de números, exponiendo los elementos que componen una función.	14, 15 RT 1,4,5,6
TP3. Implementación función esPrimo	10 min	Codificación y explicación de la implementación de la función.	14, 15 RT 1,4,5,6
T4. Paso de parámetros	5 min	Exposición de la necesidad y uso de los parámetros.	14, 15
TP5. Ejemplo paso de parámetros por valor	5 min	Ejemplo guiado de paso de parámetros por valor.	14, 15 RT 1,4,5,6
T6. Ejemplo paso de parámetros por referencia	5 min	Ejemplo guidado de paso de parámetros por referencia.	14, 15 RT 1,4,5,6
T7. Variables locales y globales	5 min	Exposición de las variables locales y globales.	15
TP8. Ejemplo práctico paso variables locales	10 min	Ejemplo guiado variables locales. Elevar un número a otro.	14, 15 RT 1,4,5,6
TP9. Creación de bibliotecas de funciones	35 min	Ejemplo guiado creación de bibliotecas de rutinas matemáticas.	14, 18, 20 RT 1,4,5,6
T10. Concepto de recursividad	10 min	Exposición del concepto y tipos de recursividad.	19
TP11. Factorial recursivo vs factorial iterativo	20 min	Implementación, depuración y comparación mediante el IDE.	14, 15, 18, 19 RT 1,4,5,6
PRÁCTICA EJERCICIOS RESUELTOS (Se dejará unos minutos para que el alumnado realice los ejercicios y se realizará la resolución y explicación en clase conforme al feedback recibido**).**			
R1. Programa que genere números aleatorio utilizando función	20 min	Implementar programa que genere una cierta cantidad de números aleatorios que le indicamos en el principal. Llamar a función que lo genere.	14, 15 RT 1, 2 3,4,5
R2. Programa que convierta de decimal a binario, usando función	30 min	Programa que convierta de decimal a binario, usando función (mediante divisiones sucesivas).	14, 15 RT 1, 2 3,4,5
R3. Contar las cifras de un entero pasado como parámetro	20 min	Implementar método que cuente las cifras de un entero. (Bucles, strings).	14, 15 RT 1, 2 3,4,5

R4. Diversas funciones array bidimensional	45 min	Implementar diversas funciones donde se pasa por referencia array bidimensional.	14, 15 RT 1, 2 3,4,5
R5. Ejemplo recursivo que produce desbordamiento pila	15 min	Implementar programa recursivo erróneo que produzca un Stack overflow. Utilizar depurador para reflexión.	14, 15, 18, 19 RT 1, 2 3,4,5
R6. Recursivo 5 4 3 2 1	25 min	Implementar programa que imprima 5 4 3 2 1 de forma recursiva. Utilizar depurador para analizar el programa.	14, 15, 18, 19 RT 1, 2 3,4,5
R7. Recursivo 1 2 3 4 5	35 min	Modificar ejemplo anterior para que imprima 1 2 3 4 5. Explicación gráfica de la pila de llamadas.	14, 15, 18, 19 RT 1, 2 3,4,5
PRÁCTICA (quedan x sesiones para la realizar los 9 ejercicios). Se indica un tiempo estimado, pero el alumnado irá a su propio ritmo, con supervisión de la docente.			
P1. Llamar a función que acepta un número como parámetro e imprima por pantalla las tablas de multiplicar (0 a 10) de dicho número	20 min	Implementar y utilizar una función que recibe parámetro e imprime sus tablas multiplicar. Creación y uso funciones.	14, 15 RT 1, 2 3,4,5
P2. Programa que indique si un número es par o impar llamando a una función booleana que lo indique	10 min	Implementar un programa que utilice una función que devuelve un valor booleano.	14, 15 RT 1, 2 3,4,5
P3. Programa que ejecuta de forma constante. Leer cadena e imprimir “cadena (IA) llamando a una función; hasta que se introduzca “salir”	10 min	Implementar un programa que se ejecute constantemente hasta que se introduzca la cadena “salir”. Debe usar función que imprima cadena (IA).	14, 15 RT 1, 2 3,4,5
P4. Menú que permita administrar alumnos de la ESO. Implementar funciones	25 min	Menú que implemente funciones. Añadir, eliminar, mostrar número alumnos y salir.	14, 15 RT 1, 2 3,4,5
P5. Biblioteca de funciones	2 sesiones	Implementar una biblioteca con 14 funciones matemáticas.	14, 15, 20 RT 1, 2 3,4,5
P6. esPrimo recursivo	40 min	Implementar recursivamente la función esPrimo.	14, 15, 18, 19 RT 1, 2 3,4,5
P7. Encontrar cuántas veces aparece la cadena b en la cadena a de forma recursiva	40 min	Implementar buscar una cadena en otra de forma recursiva.	14, 15, 18, 19 RT 1, 2 3,4,5
P8. Invertir una cadena de forma recursiva. Indicar pila de llamadas para “hola”	40 min	Programa recursivo que invierte cadena. Realizar.	14, 15, 18, 19 RT 1, 2 3,4,5
P9. Ordenar los elementos de un vector pasado como parámetro, de forma recursiva	1 sesión	Paso de parámetros por referencia. Ordenamiento vector de forma recursiva.	14, 15, 18, 19 RT 1, 2 3,4,5

ACTIVIDADES DE AMPLIACIÓN (opcionales)			
ISA DESAFÍA: ¡TE RETO!	-	Actividades de motivación y diversión. Toma de contacto con el juez del concurso de programación *Programame.*	-
TEST/CUESTIONARIO	1 sesión	Test/ Cuestionario de la unidad.	14, 15, 18, 19, 20

Cuestionario Unidad 04

1. **¿Qué significa void?**
   - ☐ Significa que es obligatorio que tenga parámetros.
   - ☐ Significa que no devuelve nada.
   - ☐ Void no existe.
2. **¿Qué límite de parámetros tenemos?**
   - ☐ 10
   - ☐ 5
   - ☐ No hay límites.
   - ☐ 3
3. **¿Puedo usar la sentencia return en un método (void)?**
   - ☐ Si.
   - ☐ No.
4. **¿Es correcto el código?**

```
public static boolean metodo1(){

    int numero = 5;

    if(numero > 10){
        return true;
    }

}
```

   - ☐ Si, con que haya un return es suficiente.
   - ☐ No, debes añadir un return en caso de que no entre en el if.