Módulo: PROGRAMACIÓN EN LENGUAJES ESTRUCTURADOS

I.E.S.: AGUADULCE **C.F.G.S. : DESARROLLO DE APLICACIONES INFORMÁTICAS**

Fecha: Martes 20 de Febrero de 2007 **Hora de Comienzo:** 17:30 h

Examen: FEBRERO – PARTE TEÓRICA **Duración:** 1 hora 30 minutos

Nombre: **D.N.I.:**

INSTRUCCIONES:

Estructura del examen PLE Febrero 2007.

- **Parte teórica (sobre 10 puntos) Tiempo disponible: 1h 30 minutos.**
  - **20 preguntas tipo test**: **5 puntos en total.**
    (+0,25 cada unasi correcta, -0,125 si incorrecta, ± 0 puntos si no contestada)
  - **5 preguntas abiertas de respuesta corta: 5 puntos en total**
    (+1punto cada una si correcta)
- **Parte práctica (sobre 10 puntos) Tiempo disponible: 1h 30 minutos.**
  - **1 diagrama de flujo: 2 puntos.**
  - **2 pseudocódigos: 4 puntos.** (2 puntos cada uno)
  - **2 programas en código java (para hacer a papel): 4 puntos.** (2 puntos cada uno)
- **La nota del examen se calculará como la media de la parte teórica y la parte práctica, siempre y cuando la nota de cada parte sea mayor o igual que 3. En caso contrario, si alguna de las partes tiene nota menor que 3), la nota final de todo el examen se calculará como el menor valor de entre 4 y la media obtenida.**
- **Para el examen práctico se podrá hacer uso del material bibliográfico que se estime oportuno, así como de apuntes. No obstante, se advierte del peligro de pérdida de tiempo que conlleva ponerse a consultarlo durante el examen, pudiendo consumirse el tiempo disponible en la consulta, y quedándose sin tiempo para las respuestas.**

EXAMEN:

PREGUNTAS TIPO TEST (Parte Teórica): Usa los espacios habilitados tras el enunciado de cada pregunta para las respuestas, escribiendo en ellos la letra o los números correspondientes a la solución correcta.

1ª.-Mediante punteros: **Respuesta:____d___**

a) Podemos almacenar en una variable todo tipo de datos.
b) Se puede reservar toda la memoria que queramos en tiempo de compilación.
c) Podemos realizar una gestión estática de la memoria más eficiente.
d) No necesitamos conocer el tamaño que va a necesitar una estructura de datos hasta que se está ejecutando el programa.

2ª.-Se produce un error de overflow: **Respuesta:____c___**

a) Cuando al operar con dos números enteros obtenemos un resultado menor que el menor número representable.
b) Cuando al operar con dos números reales obtenemos un resultado más próximo a cero que el número más cercano a cero que se puede representar.
c) Cuando al operar con dos números reales obtenemos un resultado mayor que el mayor número representable.
d) Cuando al número más grande que se puede representar para un tipo entero le sumo 1.

3ª.-Indica cuál de las siguientes afirmaciones es verdadera: **Respuesta:____b___**

a) Los literales se pueden usar como identificadores.
b) Los operadores son tokens que también actúan como separadores.
c) Las palabras reservadas se pueden utilizar como identificadores.
d) Los operadores permiten unir literales, identificadores y palabras reservadas.

4ª.-Dados A= 5 y B= -6, indica cuál de las respuestas resulta verdadera: **Respuesta:____b___**

a) (B > -3 O A < -5) Y (B > -5)
b) NO (A < 0) Y (B > -8 Y B < -1)
c) (B > -3) Y (A > 2 Y A >-1)
d) (B > -3 O A < -5) Y (B >-5)

5ª.-Indica cuál de las siguientes afirmaciones es correcta: **Respuesta:____c___**

a) En la estructura tipo mientras-hacer la condición del bucle se evalúa una vez que se ha entrado en él.

b) En la estructura tipo repetir-hasta se repiten las sentencias que incluye hasta que la condición no sea verdadera.
c) Si en una estructura repetir la condición es falsa se vuelven a ejecutar las acciones del bucle.
d) En la estructura tipo mientras-hacer aunque la condición sea falsa la primera vez se ejecutan las sentencias que hay en él.

6ª.-Dado el siguiente algoritmo en pseudocódigo, señala el resultado correcto: **Respuesta:____c___**

```
n ← 7
c ← n
Repetir
 c ← c - 1
 r ← c * c
Hasta r<=n
```

a) 36	b) 7	c) 2	d) 3

7ª.-El pseudocódigo: **Respuesta:____b___**

a) Es un lenguaje de especificación de algoritmos que como es tan parecido al lenguaje de programación el compilador se encarga de traducirlo a código máquina.
b) Se centra más en la lógica del problema que en los detalles de sintaxis.
c) Es un lenguaje tan ambiguo como el lenguaje natural y por eso facilita la creación de algoritmos.
d) Es una forma intermedia entre el lenguaje natural y el lenguaje de programación para la descripción de la solución de un problema, que evita las ambigüedades del lenguaje natural y que sigue la sintaxis de un lenguaje de programación concreto.

8ª.- Indica cuáles de las siguientes afirmaciones se considera una directriz de buen estilo de programación:

Respuesta:____a___

a) En las construcciones de nuestro programa podemos anidar tantas como sean necesarias, sin que esto conlleve la pérdida de legibilidad de nuestro programa.
b) Podemos utilizar sentencias GOTO siempre que sea necesario, aunque algunas pudieran evitarse.
c) Nuestros subprogramas deben ser lo más extensos posibles, ya que así disminuye la complejidad de nuestro programa principal.
d) Para salir de un bucle no podremos usar la sentencia GOTO que es admisible en otros muchos casos.

9ª.- Cuando hablamos de JDK nos estamos refiriendo a: **Respuesta:____c___**

a) El Kit Dinámico de Java que permite cargar objetos en memoria y que serán liberados automáticamente por el "recolector de basura" cuando éstos no se usen.
b) Una herramienta libre que nos permite compilar programas Java, aunque para ejecutarlos debemos disponer del JRE que no viene incluido en la misma.
c) Una herramienta bastante potente que nos permite compilar y ejecutar programas Java aunque no dispone de un entorno integrado de desarrollo.
d) Una herramienta comercial básica que nos permite ejecutar programas Java para plataformas como Windows, aunque todavía no ha salido la versión para Linux.

10ª.- Indica cuál de los siguientes NO es un tipo básico en Java: **Respuesta:____c___**

a) int.	b) boolean.	c) Date.	d) char.

11ª.- Relaciona los datos representados en la columna de la izquierda con el tipo de dato básico más adecuado para representarlo en Java, de la columna derecha. **RESPUESTA:**

a)	b)	c)	d)	e)
3	5	4	2	1

a) **cancelado**: Queremos almacenar si un espectáculo se ha cancelado o no.	1.- byte.
b) **letraPiso**: Representa la letra del piso de la dirección de una persona.	2.- String.
c) **diaAno**: Almacenará el día del año entre 1 y 366.	3.- boolean.
d) **codigoBarras**: Identificará el código de barras de un producto cuya precisión es de 13 cifras.	4.- short.
e) **orientaciónSexual**: Indicará la orientación sexual de una persona, cuyos valores pueden ser heterosexual (1), homosexual (2), bisexual (3) y transexual (4).	5.- char.

12ª.- Los literales de tipo boolean se pueden representar en Java: **Respuesta:____d___**

a) Utilizando las palabras true y false, o True y False.
b) Usando las palabras True y False.
c) Indistintamente usando True y False o 0 y 1.
d) Usando las palabras true y false solamente.

13ª.- Los literales enteros en Java los podemos expresar usando: **Respuesta:____a___**

a) Notación decimal, octal y hexadecimal.
b) Notación binaria, octal y hexadecimal.
c) Notación decimal, binaria y hexadecimal.
d) Notación decimal, binaria y octal.

14ª.- Un identificador en Java debe cumplir: **Respuesta:____c___**

a) Su tamaño está limitado a 256 caracteres de longitud para que pueda ser representado en un byte.
b) Puede comenzar por una letra, el símbolo "_", el símbolo "$" o por un número.
c) Se recomienda que empiece por una letra minúscula sólo si es una variable.
d) No debe contener espacios en blanco, aunque si los contiene lo debemos acotar por medio de comillas.

15ª.- Relaciona cada grupo de sentencias de la primera columna con el tipo de sentencias java al que pertenece de la segunda columna. **RESPUESTA:**

a)	b)	c)	d)	e)	f)	g)	h)	i)	j)
3	2	1	3	2	1	3	1	2	3

a) Sentencias de asignación	1. De Control de Flujo
b) Declaración e inicialización de variables **del mismo tipo**.	
c) Secuencial	
d) Creación de objetos	2. De Declaración
e) Declaración e inicialización de variables	
f) Condicional o Selectiva	
g) Llamadas a métodos, procedimientos y funciones	3. De Expresión
h) Cíclica, repetitiva o iterativa	
i) Declaración de tipo de variables	
j) Sentencias de incremento y decremento	

16ª.- Suponiendo que hemos declarado una clase llamada Alumno, con un constructor al que se le pasan como argumentos un String que se usa como Apellidos y Nombre, otro String que se usa como Grupo, y un int que se usa como Edad, y suponiendo que tenemos la siguiente sentencia:

```
Alumno alumno = new Alumno("García López, Juan", "1ºDAI A", 24);
```

indica la afirmación que NO es correcta de entre las que se proponen: **Respuesta:____a___**

a) El método new se encarga de crear un nuevo objeto de tipo Alumno, buscando memoria libre donde alojarlo y haciendo que la referencia alumno apunte a ese lugar de la memoria para poder acceder al objeto.
b) El constructor Alumno(…) es invocado por el operador new para crear un nuevo objeto de tipo Alumno que tendrá por nombre García López, Juan, por grupo 1º DAI A, y por edad 24. Ese nuevo objeto será accesible mediante la referencia alumno.
c) El operador new invoca al método constructor de la clase Alumno para crear un nuevo objeto, al que podremos acceder mediante la referencia alumno para poder usarlo.
d) En esa sentencia sólo hay un método, que es el constructor de la clase Alumno, siendo new el operador que lo invoca, busca espacio para el nuevo objeto que se obtiene como resultado de ejecutar el constructor, y hace que la referencia alumno apunte a ese espacio para poder acceder y usar el nuevo objeto creado.

17ª.- Relaciona el esquema de cada sentencia java de la primera columna, con lo que conseguimos al usarla. Suponemos que *condicion* es una expresión lógica (de tipo boolean) que se tomará el valor *true* o *false* y *expresionEntera*, como su nombre indica, una expresión que devuelve un valor de tipo int.

RESPUESTA:

a)	b)	c)	d)	e)	f)	g)	h)
2	5	3	1	7	6	4	8

a) `if(condicion){sentencia1;} else{sentencia2;}`	1. Comprobar si es necesario ejecutar un grupo de sentencias, y si es necesario, seguir ejecutándolas mientras siga siendo necesario
b) `switch(expresionEntera){ case 1: sentencia1;break; case 2: sentencia2;break; case 3: sentencia3;break; default:sentenciadefault;}`	2. Ejecutar una sentencia u otra, dependiendo del valor de una condición
c) `do{sentencia1;sentencia2; }while(condicion);`	3. Ejecutar un grupo de sentencias, y si es necesario, seguir ejecutándolas mientras siga siéndolo.
d) `while condicion) {sentencia1;sentencia2;}`	4. Terminar la ejecución de un método declarado como void.
e) `if (condicion) sentencia1;`	5. Ejecutar una sentencia u otra, dependiendo del valor de una variable
f) `for (a==1;a<=10;a++){ sentencia1;sentencia2}`	6. Ejecutar un grupo de sentencias un número conocido de veces
g) `return;`	7. Ejecutar o no una sentencia, dependiendo de una condición.
h) `return variable;`	8. Terminar un método definido como de un tipo determinado, y devolviendo el valor que produce.

18ª.-Por método recursivo entendemos: **Respuesta:____c___**

a) El que utiliza sentencias cíclicas para ejecutar repetidamente el mismo código sin tener que escribirlo nuevamente.
b) El que devuelve un valor que puede ser usado por otros métodos que lo invoquen.
c) El que se invoca a sí mismo, reduciendo la complejidad de la llamada en cada nueva invocación, de forma que llegará un momento en el que se llegará a un caso base para el que existe solución sin necesidad de hacer una nueva llamada.
d) El que se invoca a sí mismo, entrando en un bucle infinito de llamadas que el usuario deberá interrumpir introduciendo algún valor base.

19ª.- Indica las afirmaciones que son incorrectas: **Respuesta:__a_d___**

a) La solución recursiva de un problema siempre será más eficiente que la solución iterativa.
b) Las implementaciones recursivas a un problema se suelen usar por aportar soluciones elegantes y simples, fáciles de entender, para problemas complejos, cuya solución iterativa es mucho menos clara o incluso muy complicada y difícil de implementar.
c) Siempre existe una solución iterativa que se puede usar como alternativa a la solución recursiva, que además será más eficiente.
d) Para que un algoritmo recursivo esté bien definido, sólo es necesario que haga al menos una invocación a sí mismo, y que esa nueva ejecución se corresponda con una simplificación del problema.

20ª.- Para los siguientes ejemplos típicos de programas que pueden resolverse de forma recursiva, indica cuales se corresponden con una utilización adecuada de la recursividad, y cuales sería más apropiado resolver con soluciones iterativas. RESPUESTA:

a)	b)	c)	d)	e)	f)	g)	h)
1	2	2	1	2	2	2	1

a) Problema de las Torres de Hanoi.	1. Uso adecuado de recursividad para resolverlo.
b) Cálculo del término n-ésimo de la sucesión de Fibonacci.	
c) Cálculo del factorial de un número entero positivo	
d) Ordenación rápida (Quick Sort).	
e) Algoritmo de búsqueda binaria en una lista ordenada (comparando con el elementos central, para ver si el buscado es mayor o menor, y seguir buscando en la mitad adecuada, hasta que lo encontremos).	2. Uso más adecuado de una solución iterativa para resolverlo.
Euclides.	
g) Paso a binario de un número en base 10.	
h) Algoritmos de vuelta atrás (backtraking), como encontrar la salida de un laberinto, o resolver el problema del salto del caballo.	

PREGUNTAS ABIERTAS (Parte Teórica)

1ª.- Iteración frente a Recursividad:

- Indica cuándo es adecuado el uso de la recursividad para resolver un problema, y cuando es adecuado el uso de una solución iterativa.
- Enumera las ventajas e inconvenientes que aporta el uso de la recursividad y el uso de la iteración a la hora de resolver un determinado problema, haciendo mención de cómo afecta a la eficiencia del algoritmo el tipo de solución adoptada.
- Indica qué debe cumplir la definición de un algoritmo recursivo para ser correcta.

2ª.- Respecto a los lenguajes de programación, describe breve, pero correctamente:

- La diferencia entre código máquina y lenguaje de alto nivel.
- La diferencia entre compilador e intérprete.
- Ventajas e inconvenientes de un programa compilado y uno interpretado.
- El proceso de compilación y ejecución de un programa Java.
- Variables de entorno que necesitamos modificar para compilar y ejecutar un programa Java mediante el JDK.

3ª.- Describe breve, pero correctamente los conceptos de la siguiente tabla. Limítate para ello a usar el espacio disponible en la propia tabla.

• Expresión.	• Operador.
• Token.	• Separador.
• Palabra reservada.	• Estructura de control.
• Identificador.	• Máquina Virtual Java.
• Literal.	• Programa.

4ª.- ¿Qué entiendes por programación estructurada? Describe brevemente, pero con corrección, las reglas de la programación estructurada. Detalla las estructuras de control de flujo que NO se corresponden con la programación estructurada.

5ª.- Haz de compilador, y encuentra los 10 errores que hemos introducido en el siguiente código java, que obtiene la tabla de multiplicar clásica para un número entre 1 y 10 introducido por teclado. Asumimos que la clase ES es la que hemos venido usando en el curso para la lectura de datos desde teclado, y que el código está guardado en un fichero de nombre Tabla.java

Código con errores:

```
/**
 * Programado por: TutorPLE
 * Curso 2006/2007
 *
public class CódigoTablaMultiplicar {
       public static void main(String[] args){
       //inicialización de las variables. */
       int numero; contador=0;
       string continuar="";
       do{
            contador=0;
            numero=ES.leeN°("Introduce un n° entre 1 y 10 para escribir su tabla:
,1,10);
            System.out.println("TABLA DE MULTIPLICAR DEL "+numero+":");
            System.out.println("====================================")
            while (contador<=10){
                System.out.println(numero+"*"+contador+"="+numero*contador);
                contador+;
            }
            //Para continuar se puede pulsar  cualquier cosa que empiece por S o  s,
            //Cualquier otra entrada, hará que el programa termine.
            continuar=ES.leeDeTeclado("¿Deseas escribir la tabla de otro n°? [S]->Si
[N]-> No: ");
       while(continuar.charAt(0)='S' || continuar.charAt(0)='s');
   }
}
```

```
/**
 * Programado por: TutorPLE
 * Curso 2006/2007
 *
public class CódigoTablaMultiplicar {

   public static void main(String[] args){
        //inicialización de las variables. */
        int numero; contador=0;
        string continuar="";
        do{
            contador=0;
            numero=ES.leeNº("Introduce un nº entre 1 y 10 para escribir su tabla:
,1,10);
            System.out.println("TABLA DE MULTIPLICAR DEL "+numero+":");
            System.out.println("====================================")
            while (contador<=10){
                System.out.println(numero+"*"+contador+"="+numero*contador);
                contador+;
            }
            //Para continuar se puede pulsar  cualquier cosa que empiece por S o  s,
            //Cualquier otra entrada, hará que el programa termine.
            continuar=ES.leeDeTeclado("¿Deseas escribir la tabla de otro nº? [S]->Si
[N]-> No: ");
        while(continuar.charAt(0)='S' || continuar.charAt(0)='s');
   }

}
```

¡Buena Suerte!