



Universidad
de Huelva

Departamento de Ingeniería Electrónica,
Sistemas Informáticos y Automática.
Área de Ciencias de la Computación e Inteligencia Artificial

Examen de Metodología de la Programación Curso 2005-2006, Convocatoria de Febrero. Plan 2005

Apellidos: _____ Nombre: _____ DNI: _____

Sección Informativa:

- El tiempo para el test es de 25 minutos.
- Después del test se realizará un descanso de 5 minutos.
- Durante la realización de la parte teórica no se podrá salir del aula (aproveche el descanso de 5 minutos).
- Los móviles deberán permanecer desconectados durante la realización del examen.

TEST (1,5 Punto): (una pregunta mal resta media correcta).

- 1) Un objeto Cloneable
 - a. implementa el interfaz Cloneable;
 - b. implementa o hereda el método clone();
 - c. por defecto, no hace copia de los objetos referenciados;
 - d. todas son ciertas
- 2) Un interfaz
 - a. obliga siempre a implementar algún método a las clases que lo implementan
 - b. puede contener constantes estáticas
 - c. a y b
 - d. ninguna de las anteriores

- 3) En la siguiente clase en *Java*:

```
public class UnaClase {  
  
    private int variable;  
  
    public void incrementa( UnaClase otro ){  
        otro.variable++;  
    }  
  
}
```

- a. El código es incorrecto porque estamos accediendo a una variable privada de otro objeto
 - b. El código es correcto
 - c. El método incrementa sólo se puede utilizar si el objeto al que se le invoca **NO** fue declarado como final
 - d. El método incrementa sólo se puede utilizar si el objeto al que se le invoca fue declarado como final
- 4) En uno de los módulos de una librería en *C++*, una de las clases declaradas contiene un atributo de clase (*static*). ¿Dónde debemos realizar la inicialización de este atributo?
 - a. Por legibilidad, lo mejor es inicializar el atributo directamente en la misma declaración.
 - b. En el archivo de cabecera *.hpp* correspondiente ya que se trata de una declaración y, por tanto, no se genera código.
 - c. Es indiferente: podemos hacerlo tanto en el *.hpp* como en el *.cpp* sin que nos dé problemas.
 - d. En el archivo de código *.cpp* correspondiente ya que en la inicialización se genera código y esto debe hacerse en los archivos *.cpp*.



Universidad
de Huelva

Departamento de Ingeniería Electrónica,
Sistemas Informáticos y Automática.
Área de Ciencias de la Computación e Inteligencia Artificial

Examen de Metodología de la Programación Curso 2005-2006, Convocatoria de Febrero. Plan 2005

- 5) el acceso a una superclase en Java
 - a. se realiza con nombre de la clase mas ::
 - b. sólo es posible el acceso a la clase padre
 - c. a y b
 - d. ninguna de las anteriores


- 6) el acceso a una superclase en C++
 - a. se realiza con nombre de la clase mas ::
 - b. sólo es posible acceso a la clase padre
 - c. a y b
 - d. ninguna de las anteriores

- 7) En relación con la sobrescritura de métodos:
 - a. los métodos en *Java* se comportan como si los métodos equivalentes de C++ los hubiéramos declarado virtuales (*virtual*).
 - b. los métodos en *Java* se comportan como si los métodos equivalentes de C++ los hubiéramos declarado NO virtuales (*virtual*).
 - c. no se puede hacer que los métodos de C++ tengan un comportamiento equivalente a los de *Java*.
 - d. el comportamiento de los métodos es siempre equivalente con independencia que se declaren como virtuales o no.

- 8) Los métodos estáticos
 - a. sólo pueden ser llamados por otros métodos estáticos.
 - b. realizan computación que no depende directamente de las instancias que existen de una clase sino de la propia clase en sí.
 - c. pueden utilizar atributos de instancia para su computación sin ninguna restricción.
 - d. sólo pueden invocarse cuando se ha instanciado al menos un objeto de la clase que los define.

- 9) En *Java*, cuando una clase implementa un interfaz que contiene métodos con los mismos prototipos que otros métodos que hereda de su superclase,
 - a. tendremos un error de compilación.
 - b. tendremos que volver a definir los métodos para evitar la ambigüedad.
 - c. no será obligatorio volver a implementar los métodos declarados en el interfaz en la clase en cuestión.
 - d. deberemos invocar al constructor de la subclase adecuado utilizando la palabra reservada *super*.

- 10) Suponiendo que *f* es un *ostream* abierto correctamente, la llamada `f.write(&a, strlen(a)+1)` (el prototipo es `ostream& write(const char* str, streamsize n)`) guarda correctamente el contenido de:
 - a. `int a = 10;`
 - b. `int *a = new int(10);`
 - c. `char *a = "Hola"`
 - d. `char a[10] = "Hola"`

 <p>Universidad de Huelva Departamento de Ingeniería Electrónica, Sistemas Informáticos y Automática. Área de Ciencias de la Computación e Inteligencia Artificial</p>	<p align="center">Examen de Metodología de la Programación Curso 2005-2006, Convocatoria de Febrero. Plan 2005</p>
--	---

Sección Informativa:

- La parte teórica tiene una duración de 3 Horas.
- La parte teórica se evalúa sobre 7 (aprobado a partir de 3,5 puntos).
- Para hacer media con la práctica es necesario aprobar la parte teórica y tener apta la parte práctica.
- Entregue cada ejercicio en hojas separadas.
- Escriba su nombre y apellidos **EN TODAS LAS HOJAS** que entregue.
- Escriba su especialidad al menos en la primera hoja de respuestas.
- Las hojas entregadas deben estar escritas con bolígrafo azul o negro.
- Transcurridos los primeros 20 minutos de examen, se contará la convocatoria.
- Las personas que salgan del aula de teoría durante el tiempo de examen consideran terminados y entregados sus ejercicios.

1- (1,5 Puntos) Dibuje el diagrama de clases de Punto, Círculo y Dibujable. Escriba en código Java los siguiente requerimientos.

- a) Punto tiene como atributos protegidos x, y inicializados por parámetro en su constructor. Círculo hereda de Punto y añade el atributo protegido radio. El constructor de Círculo inicializa todos los atributos con los valores pasados por parámetro. (0,50)
- b) Escribe el código de Dibujable y modifica Círculo y/o Punto **sin cambiar su herencia** para que el siguiente código muestre por pantalla la salida que se muestra. (0,50)
- c) Modifique Punto para que admita coordenadas de tipos genéricos. (0,50)

Código:

```
public static void main(String[] args){
    Dibujable []d = new Dibujable[2];
    d[0]= new Circulo(1,2,2.45);
    d[1] = new Punto(1,1);
    for (int i =0;i<d.length;i++)
        d[i].dibuja();
}
```

Salida

```
Circulo>>(1,2,2.45)
Punto >>(1,1)
```

2- (1 Punto) Dada la siguiente descripción: un hombre es mamífero, un pájaro es animal, un árbol es vegetal, un mamífero es animal. Se considera que los animales respiran y los vegetales no. Animales y vegetales son seres vivos. En cuanto a reproducción son Vivíparos (mamíferos), ovíparos (pájaros), semilla (árbol).

```
Hombre h1 ("Francisco", "34 años", " c/Murcia 23 Huelva");
Pajaro p1 ("Colibri", "2 meses");
Arbol a1 ("Roble","30 años", "Latitud: 37° 12' 'N" ," Longitud 6° 54' ' W");
Mamífero *m;
SerVivo **s = new * SerVivo[3];
Vegetal *v;

m= &h1;
m->edad();
v = &a1;
v->tieneClorofila();
p1.vuela();

s[1]=m;
s[2]=&p1;
s[3]=v;
for (int i= 1;i<4;i++){
    s[i]->edad();
    s[i]->respira();
    s[i]->reproducción();
}
```

En vista de la descripción anterior y el código presentado:

- a) Cree un diagrama de clases utilizando clases abstractas cuando sea posible para que funcione este hipotético pseudocódigo y almacene además las características enumeradas en la clase más adecuada para ello. Los métodos muestran por pantalla la característica nombrada.
- b) Cree los constructores de cada clase en estilo C++.



3- (1 Punto) Mezcla de dos vectores en C++:

Dispone de las clases

Ordenable: Implementa una clase cuyas instancias contienen una clave entera en base a la cual se pueden ordenar. La idea de esta clase es servir como base para la herencia y permitir la programación de ordenables que contengan atributos de otros tipos/clases.

Vector: Clase que implementa un vector de punteros a Ordenables.

Ordenable
+ int clave
+ Ordenable(int nclave)

Vector
+ int almacenados
+ Vector ()
+ Ordenable *elementoEn(int pos)
+ void almacena(Ordenable *o)
+ void eliminaEn(int pos)

Ordenable:

clave: clave entera en base a la cual se ordenan las instancias.

Ordenable(int nclave): constructor al que le pasamos la clave de ordenación para la nueva instancia.

Vector:

Sólo se muestran los atributos y métodos públicos del vector. El resto se supone correctamente programado (respetando los detalles que se facilitan a continuación).

almacenados: atributo público que guarda el número de elementos actualmente almacenados en el vector correspondiente.

Vector(): constructor. No recibe argumentos.

Ordenable *elementoEn(int pos): método que devuelve el puntero a *Ordenable* en la posición *pos*.

void almacena(Ordenable *o): método que almacena el puntero a *Ordenable* o al final del vector.

void eliminaEn(int pos): quita el elemento en la posición *pos* del vector.

Realice una función que mezcle dos vectores. La mezcla de vectores consiste en tomar dos vectores de tamaños cualesquiera (no tienen por qué ser iguales) y construir un nuevo vector con todos los elementos de los primeros vectores de tal manera que vamos insertando en el nuevo vector la menor de las cabezas los dos vectores iniciales:



Ejemplo:

<p>V1: 2 4 6 V2: 3 5 9 1 Resultado:</p>	<p>2 4 6 < 3 5 9 1 Resultado: 2</p>
<p>4 6 > 3 5 9 1 Resultado: 2 3</p>	<p>4 6 < 5 9 1 Resultado: 2 3 4</p>
<p>6 > 5 9 1 Resultado: 2 3 4 5</p>	<p>6 < 9 1 Resultado: 2 3 4 5 6</p>
<p>V1: V2: 9 1 Resultado: 2 3 4 5 6 9 1</p>	<p>V1: V2: Resultado: 2 3 4 5 6 9 1</p>

La función tendrá el prototipo: *Vector * mezclar(Vector *v1, Vector *v2)* y los vectores v1 y v2 no deben quedar modificados después de la ejecución de la función.

4- (2 Puntos) Algorítmica:

En su trabajo como jefe de grupo de desarrollo dos de sus programadores le han preparado versiones diferentes de un algoritmo que necesitaban para el proyecto en el que trabajan actualmente.

López le presenta una versión recursiva del algoritmo implementada de la siguiente manera:

```

1) Alg( A[0..n-1] ) {
2)
3)     If (n<=1) return;
4)     For (i=0; i<n; i++)
5)         A[i]++;
6)     Alg( A[0 .. (n-1)/2] );
7)     Alg( A[ ((n-1)/2)+1 .. n-1] );
8)     Alg( A[ (n-1)/4 .. (3*(n-1))/4] );
9) }
```

Y Martínez le presenta otra implementación con un análisis que demuestra que su expresión de operaciones elementales es:

$$9 + 5n \frac{1}{\sqrt{3}} + 8n \frac{2\sqrt{2}}{\sqrt{2}} + 9n \log n$$

a) ¿Cuál es la complejidad de cada algoritmo?

(Expresión de operaciones de Alg, **0,5** puntos)

(Resolución de la recurrencia: **0,5** puntos)

(Cálculo de las complejidades: **0,5** puntos)

b) En base a la información anterior, ¿qué implementación decide incorporar a su proyecto en base a la eficiencia en tiempo de ejecución de los algoritmos? ¿Por qué? (**0,5** puntos)

NOTAS:

- La complejidad del algoritmo deberá expresarse en base a n.
- En la línea 1), A[0..n-1] denota un vector cuyo índice va de 0 a n-1.
- Considere que en las líneas 6,7,8, A[ini..fin] denota la creación de un vector resultado de “cortar” el vector A y considerar en el nuevo vector las componentes que en A van desde el índice ini hasta el índice fin. Esta computación requerirá una operación elemental (además de los accesos y/u operaciones que se realicen para calcular los nuevos índices).

RECORDATORIO:

$$n = k^i \rightarrow i = \log_k n$$

$$a^{\log_x(b)} = b^{\log_x(a)}$$

$$(a_0x^k + a_1x^{k-1} + \dots + a_k)(x-b_1)^{d_1+1}(x-b_2)^{d_2+1} \dots$$

$$a_0T(n) + a_1T(n-1) + \dots + a_kT(n-k) = b_1^n p_1(n) + b_2^n p_2(n) + \dots$$

$$T(n) = \sum_{i=1}^l \sum_{j=0}^{m_i-1} c_{ij} n^j r_i^n$$