Universidad de Huelva

Examen de Metodología de la Programación
Curso 2014-2015, Convocatoria Septiembre

Apellidos:_____Nombre:_____DNI:_____

Sección Informativa: **Duración del examen: 2 horas**

Test (0.50 Puntos)

- **El tiempo para el test es de 10 minutos.**
- **Cuando se avise del final del test deberá colocarse boca abajo en la mesa.**
- **Una pregunta mal resta 1/3 correcta, las no contestadas no puntúan.**

1) Los métodos estáticos:
   a. No pueden acceder a los atributos no estáticos de su clase ni tampoco pueden invocar los métodos no estáticos de la clase a la que pertenecen.
   b. Pueden invocarse aunque no se haya creado ningún objeto de su clase.
   c. Pueden invocar cualquier método de su clase, pero sólo pueden acceder a los atributos estáticos de su clase.
   d. Tanto la a) como la b) son correctas.

2) Cual de la siguientes cabeceras es correcta en JAVA
   a. Class A extends B, C implements D
   b. Class A implements B,C,D
   c. Class A extends B implements C, D
   d. Tanto la b) como la c) son correctas

3) En Java cuando un atributo es final:
   a. No puede ser modificado una vez inicializado
   b. Sólo puede ser accedido por los métodos final de la clase.
   c. Debe ser inicializado obligatoriamente en el momento de declararse
   d. Tanto la a) como la c) son correctas

4) En Java una referencia, pueden apuntar:
   a. Únicamente a objetos de la clase con que se declaró
   b. A objetos de la clase con que se declaró o a alguna superclase
   c. A objetos de la clase con que se declaró, a subclases o a superclases
   d. Ninguna de la anteriores

5) Dado el siguiente código

```
class Madre { ... }
class Hija extends Madre { ... }
class Nieta extends Hija { ... }
…
Madre m= new Nieta(...);
…
```

   a. m.getClass( ) == Madre.class devuelve false ya que m apunta a un objeto Nieta
   b. m instanceof Madre devuelve true ya que m apunta a un objeto Nieta
   c. Hay un error en el código: m no puede apuntar a un objeto que no es de su tipo
   d. Tanto la a) como la b) son correctas

Examen de Metodología de la Programación
Curso 2014-2015, Convocatoria Septiembre

Sección Informativa: **Duración del examen: 2 horas**

- ~~**Hay que tener las prácticas aprobadas para poder hacer el examen de teoría**~~**.**
- **La nota de teoría no se guarda. El resto de notas son válidas hasta Septiembre.**

Fórmula: Si (Teoria*50% + Sesiones Problemas*10%) >=3
Nota Acta = Sesiones Problemas *10% + Teoría*50% + Práctica*40%

1- (3.00 Puntos). C++. Dada la clase Numeros, indica qué errores se producen al ejecutarse el fichero prueba.cpp y por qué motivo. Añade e implementa los métodos necesarios en numero.cpp para que el fichero prueba.cpp produzca la salida correcta y no de ningún tipo de error al ejecutarse:

numero.cpp

```
#include <iostream>
using namespace std;

class Numeros {
  int *array;
  int n, nmax;
public:
  Numeros(int capacidad, int ini, int incr) {
    n=nmax=capacidad;
    array = new int[n];
    for(int i=0; i<n; i++)
      array[i] = ini+incr*i;
  }

  ~Numeros() {
    delete [] array;
  }

  void ver() {
    for(int i=0;i<n;i++)
      cout << array[i] << " ";
    cout << endl;
  }

  void eliminarPrimero() {
    for(int i=1; i<n; i++)
      array[i-1]=array[i];
    if (n>0) n--;
  }

  int getN() { return n; }
};
```

prueba.cpp

```
#include <iostream>
#include "numero.cpp"
using namespace std;

bool mayor(Numeros a, Numeros b) {
  if (a.getN() > b.getN())
    return true;
  return false;
}

int main() {
  Numeros a(6, 0, 2), b(3, 1, 4), c=a;
  a.ver();
  b.ver();
  c.ver();
  a.eliminarPrimero();
  if (mayor(a,b))
    cout << "a es mayor que b \n";
  b=a;
  a=a;
  a.eliminarPrimero();
  a.ver();
  b.ver();
  c.ver();
}
```

Salida correcta por pantalla:

```
0 2 4 6 8 10
1 5 9
0 2 4 6 8 10
a es mayor que b
4 6 8 10
2 4 6 8 10
0 2 4 6 8 10
```

SOLUCION: Se van a producir 2 errores: `//1.00 puntos la explicacion`

1) El constructor de copia no está implementado, con lo que se crea uno de oficio que hace una copia binaria de los datos que hace que el puntero array de la copia apunte a la misma zona de memoria a la que apunta el puntero array del original:
   El constructor de copia se ejecuta en la linea Numeros c=a; que es equivalente a Numeros c(a); lo que provoca que c y a compartan el mismo array (los cambios que se hagan en a o en c afectan a ambos ya que comparten el mismo array)
   El constructor de copia también se ejecuta en la línea if (mayor(a,b)). Al ejecutarse la función no miembro mayor se crea una copia de ambos que apuntan al mismo array. Al terminar la función mayor se destruyen las copias y al ejecutarse el destructor de dichas copias libera la zona de memoria a la que apuntan sus arrays, que al ser las mismas a las que apuntan sus originales, hace que el array de los originales quede libre para el sistema.
2) El operador de asignación no está implementado por lo que el compilador genera uno de oficio que hace una copia binaria de los datos, lo que provoca que al ejecutarse la sentencia b=a ambos objetos compartan el mismo array y la zona de memoria a la que apuntaba antes b nunca se libere

Hay que añadir los siguientes métodos al fichero numero.cpp

```
class Numeros {

  Numeros(const Numeros &c) {  //1.00 puntos
    nmax=c.nmax;
    n=c.n;
    array = new int[nmax];
    for(int i=0; i<n; i++) {
      array[i] = c.array[i];
    }
  }
```

```
  Numeros& operator=(const Numeros &c) { //1.00 puntos
    if (this != &c) {
      delete [] array;
      nmax=c.nmax;
      n=c.n;
      array = new int[nmax];
      for(int i=0; i<n; i++)
        array[i] = c.array[i];
    }
    return *this;
  }
 }
```

2- (2.00 Puntos). C++. Responde a las siguientes cuestiones: `//1.00 puntos cada una`

- ¿Cuando un destructor debe declararse virtual?. Pon un ejemplo en el que se ilustre su utilidad.

SOLUCION:

Un destructor debe declararse virtual siempre que la clase a la que pertenece sea superclase de alguna otra y necesitemos asegurarnos que al destruirse el objeto de una clase hija se ejecute el destructor del hijo (bien porque la clase hija tiene memoria dinámica y queremos garantizar que ésta se libera o porque nos interesa que al destruirse el objeto hijo su destructor haga ciertas tareas).

Cuando un método se declara virtual informa al compilador que cuando dicho método sea invocado por un puntero o por una referencia el método que se ejecute sea determinado por el tipo al que apunta el puntero o referencia y no por el tipo del puntero o referencia.

Un puntero de una clase puede apuntar a objetos de dicha clase o de clases hijas. Si el puntero de una clase se usa para apuntar a un objeto de una clase hija creado dinámicamente (new), cuando ejecutemos el operador delete para liberar dicho objeto, el destructor que se ejecuta será el de la clase del puntero (si el destructor no es virtual) o el de la clase a la que apunta el puntero (si el destructor es virtual): Si el objeto tiene atributos propios (no heredados) creados dinámicamente, en el código de su destructor habrá sentencias que se encarguen de liberar la memoria dinámica, por lo que es de vital importancia asegurarnos que dicho destructor se ejecute (en ese caso debemos declarar el destructor de su clase base como virtual).

```
class A {
public:
  A() { cout << "crea a\n"; }
  virtual ~A() { cout << "mata a\n"; }
};

class B: public A {
  int *t;
public:
  B() {
    t = new int [10];
    cout << "crea b\n";
  }
  ~B() {       //es virtual porque ~A lo es
    delete [] t;
    cout << "mata b\n";
  }
};

class C: public B {
public:
  C() { cout << "crea c\n"; }
  ~C() { cout << "mata c\n"; }//es virtual
};                            //porque ~B es

int main() {
  A *p, *pc;
  p = new B();   //crea a y luego crea b
  pc = new C();  //crea a, luego b y luego c
  delete p;
  delete pc;
}
```

Notas:

- Cuando una clase hereda un método virtual, si la clase derivada la sobreescribe también es virtual aunque no se indique explícitamente: por tanto el destructor de la clase B también es virtual aunque no se indique explícitamente.
- Si el destructor de la clase A no se hubiera declarado virtual, cuando se ejecuta la sentencia
  delete p;
  el destructor que se hubiera ejecutado sería el de la clase A (en lugar del de la clase B), ya que se ejecutará el del tipo del puntero (A) en vez el del tipo al que apunta el puntero (B), con lo que la zona de memoria reservada para el array dinámico de 10 elementos creado en el constructor de la clase B nunca se eliminaría.
- Al declarar el destructor de la clase A virtual, el destructor que se ejecuta en la sentencia
  delete p;
  es el de B que tras ejecutarse (y liberar la memoria del array) invoca automáticamente al del A

Pantalla:

Con virtual en ~A()	Sin virtual en ~A()
crea a crea b	crea a crea b
crea a crea b crea c	crea a crea b crea c
mata b mata a	mata a
mata c mata b mata a	mata a

- ¿Un método estático puede acceder o modificar cualquier atributo de su clase? Razone la respuesta.

SOLUCION:

No, ya que un método estático puede invocarse aunque no exista ningún objeto de su clase.

Un método estático sólo puede acceder y llamar a los atributos y métodos static de su clase (aunque puede llamar a cualquier miembro de los objetos pasados por parámetro).

Un método estático no puede acceder a los atributos no estáticos de su clase ni puede invocar los métodos no estáticos de su clase ya que no tienen el argumento implícito this que tienen los métodos normales, debido a que existe aunque no exista ningún objeto (es un método de clase y no un método de los objetos).

Tanto los atributos como los métodos y atributos estáticos son compartidos por todos los objetos de una clase y existen aunque no se haya creado ningún objeto de dicha clase: por ello no tiene sentido que un método estático acceda a un atributo no estático ya que éstos son de un objeto concreto y no de la clase en sí (cada objeto de una clase tiene su propia copia de cada una de las variables miembros de esa clase. Dichos miembros no existen hasta que no creamos un objeto).

Por igual motivo, un método estático no puede invocar un método no estático ya que los métodos no estáticos acceden e invocan atributos y método del objeto concreto que lo invoca (si el método estático es común a todos los objetos e invoca un método no estático que accede a un atributo no estático ¿a que objeto concreto pertenece dicho atributo?).

```
class A {
  static int n;
  int y;
public:
  A() { y = 0; n++; }
  A(int y) { this->y = y; n++; }
  static int getN() { return n; }
  int getY() { return y; }
  ...
};

int A::n = 0;

int main() {
  A::getN();
  A a, b(5);
  a.getY();
  A::getN();
}
```

Notas:

- n lo utilizamos para contar cuantos objetos de la clase A creamos.
- n es declarado estático para que dicho atributo sea compartido por todos los objetos que se creen de la clase A
- al ser estático existe antes de crearse cualquier objeto de la clase (por ello no se inicializa en el constructor de la clase, sino fuera de ella).
- Como podemos observar en el main el método estático es invocado antes de crearse objetos de la clase A: si dentro del método estático hubiera alguna línea de código que accediera al atributo y o al método getY no se podría saber a qué objeto y pertenece dicho atributo ya que en ese momento no existe ningún objeto y si existiera varios (como ocurre en al invocarse en la ultima línea del main, el compilador no podría determinar si se refiere al del objeto a o al delobjeto b

3- **(4.50 Puntos). C++ y Java.** En una biblioteca se desea crear un programa para controlar los libros y revistas que tiene en depósito. Teniendo en cuenta las siguientes restricciones:

- Los libros y revistas tienen en común los siguientes campos: titulo, autor y ISBN (código numérico que lo identifica unívocamente).
- En la biblioteca pueden existir varios ejemplares de un mismo libro, mientras que de cada revista solo hay un único ejemplar (ej. la biblioteca está suscrita a las revistas "NATIONAL GEOGRAPHICS" y "MUY INTERESANTE" de la que sólo llega un ejemplar cada mes).
- De cada revista llega un número nuevo cada mes y nos interesa saber qué números de dichas revistas tenemos en depósito.
- El número páginas es un dato que nos interesa sólo de los libros (no de las revistas)

```
public static void main(String[] args) {
    System.out.println( "BIBLIOTECA\n");
    Biblioteca uhu = new Biblioteca();
    Libro lib1=new Libro("LA COLMENA", "Cela", 1001, 1, 276);
    Libro lib2=new Libro("PLATERO Y YO", "Jimenez", 1002, 2, 330);
    Libro lib3=new Libro("EL HOBBIT", "Tolkien", 1003, 2, 330);
    Revista rev2=new Revista("MUY INTERESANTE", "Godoy", 2001, 45);
    Revista rev3=new Revista("NATIONAL GEOGRAPHICS", "Gil", 2002, 60);

    uhu.alta(rev2);
    uhu.alta(rev3);
    uhu.alta(lib2);
    uhu.alta(lib3);
    uhu.alta(lib1);
    uhu.alta(rev3); //No debe darlo de alta porque ya existe un volumen con ese ISBN
    uhu.alta(lib3); //No debe darlo de alta porque ya existe un volumen con ese ISBN

    System.out.print("\nListado de Libros y Revistas\n\n");
    uhu.listar();

    int nej=uhu.buscar(2002); //busca el libro o revista con ISBN 2002
    if (nej != -1) {          //devuelve -1 si no esta en la biblioteca
        Revista r=(Revista)uhu.getVolumen(nej); //devuelve el libro o revista que esta en la pos nej
        r.agregarNumero(61);  //agrega el numero 61 a la revista
        r.agregarNumero();    //agrega el ultimo numero a la revista
        r.agregarNumero(70);  //agrega el numero 70 a la revista
    }

    ((Libro)uhu.getVolumen(uhu.buscar(1001))).agregarEjemplares(5); //agregar 5 ejemplares al libro
                                                                    //con ISBN 1001
    uhu.agregarNumero(); //agrega a todas las revistas de la biblioteca su ultimo numero

    System.out.print("\nListado de Revistas\n\n");
    uhu.listarRevistas();
    System.out.print("\nListado de Libros\n\n");
    uhu.listarLibros();
}
```

PANTALLA:

```
BIBLIOTECA
Error: Ya existe un volumen con ese ISBN
Error: Ya existe un volumen con ese ISBN

Listado de Libros y Revistas

1 MUY INTERESANTE (Godoy) 2001 (1 numeros: 45)
2 NATIONAL GEOGRAPHICS (Gil) 2002 (1 numeros: 60)
3 PLATERO Y YO (Jimenez) 1002 2 ej. (330 pag.)
4 EL HOBBIT (Tolkien) 1003 2 ej. (330 pag.)
5 LA COLMENA (Cela) 1001 1 ej. (276 pag.)

Listado de Revistas

1 MUY INTERESANTE (Godoy) 2001 (2 numeros: 45,46)
2 NATIONAL GEOGRAPHICS (Gil) 2002 (5 numeros: 60,61,62,70,71)

Listado de Libros

1 PLATERO Y YO (Jimenez) 1002 2 ej. (330 pag.)
2 EL HOBBIT (Tolkien) 1003 2 ej. (330 pag.)
3 LA COLMENA (Cela) 1001 6 ej. (276 pag.)
```

a) **C++.** Indica cómo sería la definición de las clases de dicho programa. Diseña bien las clases que necesita el programa, de forma que se tenga que codificar lo menos posible y que sólo se puedan crear objetos que sean instancias de libros y revistas. Indica los métodos que deben tener las clases para que el main( ) anterior (OJO está escrito en Java) pueda ejecutarse.
Solo hay que indicar el contenido de los .h (no implementar los métodos. **(2.00 Puntos).**

SOLUCION:

```
class Volumen {                          //la clase Volumen debe ser abstracta para que no se puedan
  char *titulo;                          //crear objetos Volumen (si se podrán crear objetos de tipo
  char *autor;                           //Libro y Revista, los cuales derivan de Volumen)
  int ISBN;
public:
  Volumen(char *titulo, char *autor, int ISBN);
  virtual ~Volumen();

  Volumen(const Volumen &v);             //aunque la clase sea abstracta y no pueda recibir como
  Volumen& operator=(const Volumen &v);  //parametro un objeto Volumen, se implementa para que
                                         //las clases hijas las llamen al implementar sus
  virtual void ver() const = 0;          //constructor de copia y operador de asignacion
  int getISBN() { return ISBN; }
protected:
  void verDatos() const;
};

class Libro: public Volumen {
  int nejemplares;
  int npaginas;
public:
  Libro(char *titulo, char *autor, int ISBN, int nej, int npag);
  virtual ~Libro();

  //Libro(const Libro &lib);             //no hace falta ni el constructor de copia ni el operator=
  //Libro& operator=(const Libro &lib); //porque la clase no tiene atributos dinamicos por lo que los
                                         //que genera de oficio el compilador funcionan perfectamente
  void ver() const;
  void agregarEjemplares(int n);
};

class Revista: public Volumen {
  int *numero; //array dinamico con una capacidad inicial de 12 elementos
  int n; //numero de elementos en el array numero
  int nmax; //capacidad del array
public:
  Revista(char *titulo, char *autor, int ISBN, int numero);
  virtual ~Revista();

  Revista(const Revista &r);             //en los inicializadores llama al constructor copia de Volumen
  Revista& operator=(const Revista &r); //lo primero que hace es llamar al operator= de Volumen

  void ver() const;
  void agregarNumero(int num);
  void agregarNumero();
};

class Biblioteca {
  Volumen **deposito; //array dinamico con una capacidad inicial de 100 elementos
  int n; //numero de elementos en el array deposito
  int nmax; //capacidad del array
public:
  Biblioteca();
  virtual ~Biblioteca();

  Biblioteca (const Biblioteca &b);          //todas las clases con atributos dinámicos (punteros)
  Biblioteca& operator=(const Biblioteca &b); //deben implementar el constructor de copia y operator=

  bool alta();                               //si no los implementan, el main() funciona bien
  bool alta(Volumen *v);                     //siempre y cuando NUNCA usen el operador de asignacion
  int buscar(int ISBN);                      //y nunca usen el constructor de copia (explícitamente)
  void listarLibros();                       //y no haya ningún método o función que pase un objeto
  void listarRevistas();                     //de dicha clase por valor (si se pasa por valor se
  void listar();                             //ejecuta el constructor de copia) o devuelva un objeto
  Volumen *getVolumen(int n);                //de dicha clase (al devolverse un objeto se ejecuta
  void agregarNumero();                      //el constructor de copia, a menos que se devuelve una
};                                           //referencia)
```

Nota:

La clase Libro no tiene atributos dinámicos, por lo que no hace falta el constructor de copia ni el operador de asignación ya que el de que genera de oficio el compilador funciona bien (llama al del padre y hace copia binaria de los atributos exclusivos del hijo)

En el main() anterior en ningún momento se invoca constructor copia y operator=
Los ponemos por si “en un futuro” se invocaran en el main(), pero para este examen concreto no se exigía ponerlo

b) **Java.** Implementa en Java lo siguiente:

- La clase Revista completa (sus atributos y todos sus métodos): Se deben implementar todos los métodos que aparecen explícitamente en el main(), incluido por supuesto el constructor de la clase **(1.90 puntos)**

SOLUCION:

```
public class Revista extends Volumen {
    private int [] numeros;
    private int n;

    public Revista(String titulo, String autor, int ISBN, int numero) {
        super(titulo, autor, ISBN);
        numeros=new int[12];// initialise instance variables
        numeros[0] = numero;
        n = 1;
    }

    void agregarNumero() {
        agregarNumero(numeros[n-1]+1);
    }

    void agregarNumero(int num) {
        if (n==numeros.length) {
            int [] aux = numeros;
            numeros=new int[numeros.length+12];
            for(int i=0; i<n;i++)
                numeros[i]=aux[i];
        }
        numeros[n]=num;
        n++;
    }
}
```

- La definición de los atributos de la clase Biblioteca y la implementación del método agregarNumero( ) de dicha clase **(0.60 puntos)**

SOLUCION:

```
public class Biblioteca {
    private Volumen [] deposito;
    private int n;

    void agregarNumero() {
        for (int i=0; i<n; i++) {
            if(deposito[i].getClass() == Revista.class) {
               ((Revista)deposito[i]).agregarNumero();
            }
        }
    }
}
```