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

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- (4.5 Puntos). C++. Dadas las siguientes clases, indica qué métodos hay que implementar teniendo en cuenta lo siguiente:

- Clase1 debe tener un único constructor con un parámetro obligatorio y queremos llevar la cuenta de los objetos de tipo Clase1 que existen en todo momento en memoria.
- Si al crear objetos de Clase2 no se indica el tamaño máximo de la tabla, debe crearse una tabla de tamaño 10.
- A la hora de crear objetos de Clase3 ~~no~~ hay que pasar ~~como~~ **por** parámetro **z** ~~ningún parámetro~~ (debe pedir ~~todos los~~ **el resto de datos** que sean necesarios por teclado dentro del constructor).

Rellena y completa el fichero cabecera examen.h indicando los prototipos de todos los métodos que sean necesarios para que las clases funcionen para cualquier main( ) posible.

Implementa todos los constructores y destructores que hayas indicado (el resto de métodos no hay que implementarlos, solo indicar su prototipo).

examen.h

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

class Clase1 {
  static int n; //para saber cuantos objetos de tipo Clase3 se crean
  int i;
public:
  Clase1(int x); //el parámetro x es obligatorio
//A RELLENAR POR EL ALUMNO
};

class Clase2: public Clase1 {
  int tamanioMax;
  int tamanio;
  int *tabla; //el constructor crea la tabla con un tamaño indicado por parámetro o 10 si no se indica
public:
  //A RELLENAR POR EL ALUMNO
};

class Clase3: public Clase2 {
  const int z;
  Clase1 y;
public:
  //A RELLENAR POR EL ALUMNO
};
```

SOLUCION: La puntuación de cada método viene indicada en color azul

Para que funcionen para cualquier main( ) posible, se debe contemplar la posibilidad de que haya **objetos constantes y punteros** en dicho main().

Si hay objetos constantes → los métodos que sean posibles deben ser const (+0,30 puntos)
Si hay punteros → los destructores deben ser virtuales (+0,30 puntos)

Clase1: aunque no tiene memoria dinámica **hay que implementar el constructor, el constructor de copia (y destructor) para que incremente (decremente) el contador de objetos**

Clase2: al tener memoria dinámica hay que implementar el constructor, el constructor de copia, el destructor y el operador de asignación

Clase3: al no tener memoria dinámica en principio no hace falta constructor de copia, ni destructor, ni operador de asignación… pero **al tener un atributo constante (z) hace falta implementar el operador de asignación para que copia todo excepto el atributo constante (z)**

Los métodos geti(), seti() y setMax() son necesarios para implementar el constructor de Clase2 y el de Clase3.
En el examen no había que implementarlos, solo indicar su prototipo (los implemento para que veáis como se haría)

examen.h

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

class Clase1 {
  static int n; //para saber cuantos objetos de tipo Clase3 se crean
  int i;
public:
  Clase1(int x) { i=x; n++; } //el parámetro x es obligatorio                    //0.20 puntos
  Clase1(const Clase1 &c) { i=c.i; n++; }                                        //0.20 puntos
  virtual ~Clase1() { n--; }                                                     //0.50 puntos
  int geti() const { return i; } //necesario en metodo setMax de Clase2          //0.40 puntos
  void seti(int i) { this->i = i; }                                              //0.10 puntos
};

class Clase2: public Clase1 {
  int tamanioMax;
  int tamanio;
  int *tabla; //el constructor crea la tabla con un tamaño indicado por parámetro o 10 si no se indica
public:
  Clase2(int i, int tam=10);
  Clase2(const Clase2 &c);
  ~Clase2();
  Clase2& operator=(const Clase2 &c);                                            //0.10 puntos
  void setMax(int tamMax) { Clase2 aux(geti(),tamMax);  (*this)=aux;  }          //0.10 puntos
};

class Clase3: public Clase2 {
  const int z;
  Clase1 y;
public:
  Clase3(int z);
  Clase3& operator=(const Clase3 &c);                                            //0.30 puntos
};
```

examen.cpp

```
#include "examen.h"

int Clase1::n=0;                                                                 //0.30 puntos

Clase2::Clase2(int i, int tam):Clase1(i) {                                       //0.40 puntos
  tamanioMax=tam;
  tabla=new int[tam];
  tamanio=0;
}

Clase2::Clase2(const Clase2 &c):Clase1(c) {                                      //0.80 puntos
  tamanioMax=c.tamanioMax;
  tabla=new int[c.tamanioMax];
  tamanio=c.tamanio;
  for(int i=0; i<tamanio; i++)
    tabla[i]=c.tabla[i];
}
```

```
Clase2::~Clase2() { delete [] tabla; /*cout << "destruyendo..." << tamanioMax << endl;*/ }      //0.20 puntos

Clase2& Clase2::operator=(const Clase2 &c) {                          //en el examen solo había que
  if (this != &c) {                                                   //implementar los constructores
    Clase1::operator=(c); //lo 1º es llamar al = del padre            //y los destructores
    delete [] tabla;                                                  //
    tamanioMax=c.tamanioMax;                                          //Aquí he implementado el operator=
    tabla=new int[c.tamanioMax];                                      //para que sepias como se hace
    tamanio=c.tamanio;                                                //aunque para el examen no se pedia
    for(int i=0; i<tamanio; i++)
      tabla[i]=c.tabla[i];
  }
  return *this;
}

Clase3::Clase3(int vz):Clase2(0), z(vz), y(0) {                                   //0.90 puntos
  int i, tam;
  cout << "valor del atributo y.i: ";
  cin >> i;
  y.seti(i);    //seti() en Clase1 es necesario para poder modificar el atributo privado y.i
  cout << "tamaño de la tabla: ";
  cin >> tam;
  cout << "valor del atributo i heredado de Clase1: ";
  cin >> i;
  seti(i);      //seti() en Clase1 para modificar el atributo privado i que Clase3 hereda de Clase1
  setMax(tam);  //setMax() en Clase2 para modificar los atributos privados tabla y tamanioMax
}               //que Clase3 hereda de Clase2

Clase3& Clase3::operator=(const Clase3 &c) {                          //en el examen solo había que
  if (this != &c) {                                                   //implementar los constructores
    Clase2::operator=(c); //lo 1º es llamar al operator= del padre    //y los destructores
    y=c.y;                                                            //
  }                                                                   //Aquí he implementado el operator=
  return *this;                                                       //para que sepias como se hace
}                                                                     //aunque para el examen no se pedia
```

Para probar su correcto funcionamiento, crea un proyecto con los ficheros examen.h, examen.cpp y main.cpp (en examen.h introduce las siguientes funciones amigas en cada clase) y ejecuta el siguiente main( ), donde hay objetos constantes (solo pueden ejecutar métodos **const**, aparte por supuesto de los constructores y destructoress) y un puntero de Clase1 (es la clase base de la que heredan el resto de clases) que apunta a un objeto de Clase3 y lo destruye y luego, a un objeto de Clase2 que posteriormente también destruye. **Descomenta el cout que hay en el destructor de Clase2** para que comprobéis que efectivamente, se ejecuta el destructor correcto (cuando p apunta a un objeto de Clase3 y éste se destruye se ejecuta el destructor de Clase3, el cual llama al de Clase2, el cual llama al de Clase1; cuando p apunta a un objeto de Clase2 se ejecuta el destructor de Clase2, el cual llama al de Clase1). **Si el destructor de Clase1 no fuera virtual, al destruirse el objeto de Clase3 y el objeto de Clase2 apuntado por p solo se ejecutaría el destructor de Clase1** (quitad la palabra **virtual** y comprobareis que lo mensajes que lanza el destructor de Clase2 no aparece, demostrando lo dicho).

examen.h

```
class Clase1 {
  ...
  friend ostream& operator<<(ostream &s, const Clase1 &c) {
    s << "n: " << c.n << " i: " << c.i;
    return s;
  }
}

class Clase2: public Clase1 {
  ...
  friend ostream& operator<<(ostream &s, const Clase2 &c) {
    s << (Clase1 &)c << " tamanioMax: " << c.tamanioMax
    s << " tamanio: " << c.tamanio << " { ";
    for(int i=0; i<c.tamanio; i++)
      s << c.tabla[i] << " ";
    s << " }";
    return s;
  }
}

class Clase3: public Clase2 {
  ...
  friend ostream& operator<<(ostream &s, const Clase3 &c) {
    s << (Clase2 &)c << " z: " << c.z << " y(" << c.y << ")";
    return s;
  }
}
```

main.cpp

```
#include <iostream>
#include <sstream>
#include "examen.h"
using namespace std;

int main() {
  const Clase1 x(1);
  Clase1 xx(11);
  const Clase2 y(2);
  Clase2 yy(2);
  const Clase3 z(3);
  Clase3 zz(33), zzz(zz);
  cout << "x: " << x << endl;
  cout << "y: " << y << endl;
  cout << "z: " << z << endl;
  xx=xx;
  xx=x;
  yy=y;
  zz=zz;
  zz=z;
  cout << "xx: " << xx << endl;
  cout << "yy: " << yy << endl;
  cout << "zz: " << zz << endl;
  cout << "zzz: " << zzz << endl;
  cout <<"\n--------------\n";
  Clase1 *p=new Clase3(333);
  delete p;
  cout <<"\n--------------\n";
  p=new Clase2(222);
  delete p;
  cout <<"\n--------------\n";
}
```

2- **(4 Puntos). Java.** Dado el siguiente código indica cual sería la definición de las clases e implementa todos los métodos para que el main( ) produzca la salida indicada:

Nota: A la hora de pedir datos por teclado en vez de usar System.out.pritnln usar sysout y para pedir datos por teclado usad sysin.

```
sysout("UAM");    //muestra por pantalla UAM   (es como si pusiéramos System.out.pritnln("UAM");)
sysin(this.edad);  //pide por teclado la edad
```

```
package libClases;

public class Persona {
  private int [] telefonos;
  private int edad;
  private String nombre;

  //A RELLENAR POR EL ALUMNO (2.4 puntos)
}
```

```
package libClases;

public class Alumno extends Persona {
  String universidad;

  //A RELLENAR POR EL ALUMNO (1.6 puntos)
}
```

```
package libPruebas;

import libClases.*;

public class Prueba1 {
  public static void main(String[] args) {
    Persona a=new Persona("juan", 23, 1), b=new Persona(), c=new Persona(b);
    Persona x=new Alumno("UHU", "pepe", 23, 2);
    Alumno y=new Alumno ("UCO"), z, j=new Alumno("UHU", a);
    c=a;
    z=y;
    ((Alumno) x).cambiarUniversidad("UAM");    //el alumno x se cambia a la universidad UAM
    if (Persona.mayor(b,a))                    //true si b tiene mas años que a
      System.out.println("b:" + b);
    System.out.println(z);
  }

}
```

Salida:

```
Indica telefono1 de juan: 657084578                                    //a

Nombre: luis                                                           //b
Edad: 46
Cuantos teléfonos tiene: 0

Indica telefono1 de pepe: 666684500                                    //x
Indica telefono2 de pepe: 655584517


Nombre: ana                                                            //y
Edad: 19
Cuantos teléfonos tiene: 2
Indica telefono1 de ana: 600084500
Indica telefono2 de ana: 600084517


pepe se ha cambiado de la universidad UHU a la universidad UAM

b:luis tiene 46 años y tiene 0 telefonos
ana tiene 19 años y tiene 2 telefonos (600084500 600084517). Estudia en la universidad UCO
```

SOLUCION: La puntuación de cada método viene indicada en color azul

Persona.java **(2.4 puntos)**

```
import java.util.Scanner;

public class Persona {
	private int [] telefonos;
	private int edad;
	private String nombre;

	public Persona(String nom, int edad, int ntel) {                              //0.50 puntos
		Scanner sc = new Scanner(System.in);
		nombre=nom;
		this.edad=edad;
		telefonos=new int[ntel];
		for(int i=0; i<ntel; i++) {
			System.out.print("Indica telefono" + (i+1) + " de "+ nombre+": ");
			telefonos[i]=sc.nextInt();
		}
	}

	public Persona() {                                                             //0.50 puntos
		Scanner sc = new Scanner(System.in);
		System.out.print("Nombre: ");
		nombre = sc.next();
		System.out.print("Edad: ");
		edad=sc.nextInt();
		System.out.print("Cuantos telefonos tiene: ");
		int ntel=sc.nextInt();
		telefonos=new int[ntel];
		for(int i=0; i<ntel; i++) {
			System.out.print("Indica telefono " + (i+1) + " de "+ nombre+": ");
			telefonos[i]=sc.nextInt();
		}
	}

	public Persona(Persona p) {                                                    //0.50 puntos
		nombre=p.nombre;
		edad=p.edad;
		telefonos=new int[p.telefonos.length];
		for(int i=0; i<telefonos.length; i++) {
			telefonos[i]=p.telefonos[i];
		}
	}

	public String getNombre() { return nombre; }                                   //0.10 puntos

	public static boolean mayor(Persona a, Persona b) {                            //0.30 puntos
		return (a.edad>b.edad);
	}

	public String toString() {                                                     //0.50 puntos
		String s=nombre+" tiene " + edad + " años y tiene " + telefonos.length +
				" telefonos";
		if (telefonos.length>0) {
			s=s+" ( ";
			for(int i=0; i<telefonos.length; i++)
				s=s+telefonos[i]+ " ";
			s=s+")";
		}
		return s;
	}
}
```

Alumno.java **(1.6 puntos)**

```
package libClases;

public class Alumno extends Persona {
	private String universidad;

	public Alumno(String uni, String nom, int edad, int ntel) {                  //0.30 puntos
		super(nom, edad, ntel);
		universidad=uni;
	}

	public Alumno(String uni) {                                                  //0.30 puntos
		//super();    //no es necesario, si no lo ponemos el compilador lo hace
		universidad=uni;
	}

	public Alumno(String uni, Persona p) {                                       //0.30 puntos
		super(p);
		universidad=uni;
	}

	public void cambiarUniversidad(String uni) {                                 //0.30 puntos
		System.out.println(getNombre()+ " se ha cambiado de la universidad " +
				universidad + " a la universidad " + uni);
		universidad=uni;
	}

	public String toString() {                                                   //0.40 puntos
		return super.toString()+". Estudia en la universidad " + universidad;
	}
}
```