Examen de Metodología de la Programación
Curso 2017-2018, Convocatoria Febrero

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

1- (5 Puntos). C++. Dadas las clases Par y ParEtiquetado, implementa **el mínimo número** de métodos y funciones para que el siguiente main( ) funcione correctamente y genere la salida indicada:

Notas:

- **No está permitido el uso de funciones amigas**
- **En la salida, fíjate bien en las claves de cada objeto para deducir cómo y cuándo son asignadas**
- Si hay algún método que no hace falta implementar **justifica la razón** por la que no es necesario.
- Las clases **no pueden tener ningún método que no se invoquen** directa o indirectamente en el main.
- Indica en el main que métodos son los que se ejecutan en las líneas 24, 26, 27 y 28.

```
 1  #include <iostream>
 2  #include <sstream>
 3  #include <cstring>
 4  #include <cstdlib>
 5  using namespace std;
 7
 8  class Par {
 9    const int clave;
10    float valor;
11  public:
12    //A RELLENAR POR EL ALUMNO    // 2.70 puntos
13  };
14
15  class ParEtiquetado: public Par {
16    char *etiqueta;
17  public:
18   //A RELLENAR POR EL ALUMNO     // 1.90 puntos
19  };
20
21  int main() {                    // 0.40 puntos indicar lo que ocurre en líneas 24, 26, 27 y 28
22    Par a(4.5), b(5.0), c(a);
23    const Par x=a.copia();
24    float n=x;
25    cout << "a:" << a << " b:" << b << " c:" << c << endl;
26    a=4;
27    b=1+x+b;
28    Par y;
29    cout << "a:" << a << " b:" << b << " c:" << c << endl;
30    cout << "x:" << x.texto() <<  " y:" << y.texto() << endl;
31    cout << "n:" << n << " clave de x: " << x.getClave() << endl;
32
33    const ParEtiquetado pe1("xxx",3.9);
34    ParEtiquetado pe2=pe1;
35    Par *ppar=new ParEtiquetado("yyy", 4.7);
36    cout << "pe1:" << pe1 << " pe2:" << pe2 << " ppar:" << *((ParEtiquetado *)ppar) << endl;
37    cout << pe1.getEtiqueta() << endl;
38    cout << pe2.cambiarEtiqueta("zzz") << endl;
39    cout << "pe1:" << pe1.texto() <<  " pe2:" << pe2.texto() << " ppar:" << ppar->texto() << endl;
40    delete ppar;
41    system("PAUSE");
42    return EXIT_SUCCESS;
43  }
```

Salida:

```
a:1-4.5 b:2-5 c:1-4.5
a:1-4 b:2-10.5 c:1-4.5
x:3(4.5) y:6(0)
n:4.5 clave de x: 3
pe1:xxx,7-3.9 pe2:xxx,7-3.9 ppar: yyy,8-4.7
xxx
zzz
pe1:xxx,7(3.9) pe2:zzz,7(3.9) ppar:yyy,8(4.7)
Presione una tecla para continuar . . .
```

SOLUCION: (en rojo viene indicado la puntuación de cada apartado)

En celeste marcamos soluciones alternativas

En gris marcamos el por qué hay que usar const (hay que mirar el main()…)

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

class Par {
  static int n;                                          //0.20
  const int clave;
  float valor;

public:
  Par(float v=0): clave(n) {              //obligatorio usar inicializadores al ser clave constante
    valor=v; n++;                                        //0.30
  }

//Par(const Par &p): clave(p.clave) {   //nos sirve el constructor copia de oficio ya que n no cambia
    valor=p.valor;                                       //0.30
  }

  int getClave() const { return clave; }                 //0.10

  operator float() const {                //necesario para que float n=x; no de error
       return valor;                      //y pueda convertir el Par x en un float
  }                                                      //0.30

  Par& operator=(const Par &p) {          //obligatorio implementarlo ya que el de oficio falla
       valor=p.valor;                     //porque clave es constante y no se puede cambiar
       return *this;                                     //0.40
  }


  virtual string texto() const {          //virtual para que: Par *ppar=new ParEtiquetado("y", b);
    stringstream s;                       //  ppar->texto() ejecute texto() de ParEtiquetado
    s << clave << "(" << valor << ")";    //  si no se pone virtual entonces se ejecuta texto() de Par
    return s.str();                                      //0.30
  }

  Par copia() const { return Par(valor); }               //0.30
};

ostream& operator<<(ostream &s, const Par &p) {          //0.30
    s << p.getClave() << "-" << (float)p;
    return s;
}

class ParEtiquetado: public Par {
  char *etiqueta;

public:
  ParEtiquetado(const char *cad, float f): Par(f) {      //0.30
    etiqueta=new char[strlen(cad)+1];
    strcpy(etiqueta, cad);
  }

  ParEtiquetado(const ParEtiquetado &p): Par(p) {        //0.30
    etiqueta=new char[strlen(p.etiqueta)+1];
    strcpy(etiqueta, p.etiqueta);
  }

  ~ParEtiquetado() { delete [] etiqueta; }               //0.20

  const char *getEtiqueta() const { return etiqueta; }   //0.10

  const char *cambiarEtiqueta(const char *cad) {         //0.40
    delete [] etiqueta;
    etiqueta=new char[strlen(cad)+1];
    strcpy(etiqueta, cad);
    return etiqueta;
  }

  string texto() const {                                 //0.30
    stringstream s;
    s << etiqueta << "," << Par::texto();
    return s.str();
  }
};
```

```
ostream& operator<<(ostream &s, const ParEtiquetado &p) {  //0.30
    s << p.getEtiqueta() << "," << (Par &)p;
    return s;
}

int Par::n=1;                                              //0.20

int main() {
  Par a(4.5), b(5.0), c(a);    //a,b constructor (incrementa clave) //Par(float v=0);
                               //c constructor copia de oficio (no incrementa clave) //Par(const Par &p):
                              //no memoria dinámica -> vale constructor copia de oficio del compilador
                              //porque en la copia no incrementamos clave
  const Par x=a.copia();      //x constructor copia (de oficio) //Par copia() const
  float n=x;  //0.10                          //float n=(float)x;  // operator float() const;
  cout << "a:" << a << " b:" << b << " c:" << c << endl;  // ostream& operator<<(ostream &s, const Par &p)
  a=4;        //0.10                          //a.operator=(Par(4)); //Par& operator=(const Par &p) ...
  b=1+x+b;    //0.10                          //b.operator=(Par(1+(float)x+(float)b));
  Par y;      //0.10                          //Par(float v=0);
  cout << "a:" << a << " b:" << b << " c:" << c << endl;  // ostream& operator<<(ostream &s, const Par &p)
  cout << "x:" << x.texto() <<  " y:" << y.texto() << endl;     //string texto() const
  cout << "n:" << n << " clave de x: " << x.getClave() << endl; //int getClave() const

  const ParEtiquetado pe1("xxx",3.9);      //ParEtiquetado(const char *cad, float f);
  ParEtiquetado pe2=pe1;                     //ParEtiquetado(const ParEtiquetado &p);
  Par *ppar=new ParEtiquetado("yyy", 4.7);   //ParEtiquetado(const char *cad, const Par &p):
  cout << "pe1:" << pe1 << " pe2:" << pe2 << " ppar:" << *((ParEtiquetado *)ppar) << endl;
  cout << pe1.getEtiqueta() << endl;         //const char *getEtiqueta() const;
  cout << pe2.cambiarEtiqueta("zzz") << endl;//const char *cambiarEtiqueta(const char *cad)
  cout << "pe1:" << pe1.texto() <<  " pe2:" << pe2.texto() << " ppar:" << ppar->texto() << endl;
  delete ppar;                               //~ParEtiquetado();
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

2- (5 Puntos). Java. Dado el siguiente código:

```
public class Fecha {
        private int dia, anio;
        private String mes;
        public Fecha(int d, String m, int a) { dia=d; mes=m; anio=a; }

        public int getDia() { return dia; }
        public String getMes() { return mes; }
        public int getAnio() { return anio;  }

        public void setFecha(int d, String m, int a) {      dia=d; mes=m; anio=a; }
        public String toString() { return dia+"/"+mes+"/"+anio; }
}

public class Persona {
        private String nombre;
        private Fecha fecha;
        public Persona(String nom, Fecha f) { nombre=nom; fecha=f; }

        public String getNombre() { return nombre; }
        public Fecha getFecha() { return fecha; }
        public String toString() { return nombre+" "+fecha; }
}

public class Prueba {
        public static void main(String[] args) {
                Persona p=new Persona("pepe", new Fecha(2, "feb",2000));
                System.out.println(p.getNombre()+" nacio el "+p.getFecha());

                System.out.println(p);
        }
}
```

1) Sin modificar nada en las clases **Fecha** y **Persona**, ¿es posible en el **main()** cambiar la fecha a 1/1/2001?. Si crees que si añade en el **main()** el código necesario para poder hacerlo. Si crees que no indica el motivo por el que no es posible. **(0.50 puntos)**

2) ¿Y si fuera final el atributo **fecha** de **Persona**?, es decir, ¿y si fuera **final private Fecha fecha;** en lugar de **private Fecha fecha;**? **(0.50 puntos)**

3) Si la clase Fecha fuera así y no se pudiera cambiar absolutamente nada…

```
public class Fecha {
        final private int dia, anio;
        final private String mes;

        public Fecha(int d, String m, int a) { dia=d; mes=m; anio=a; }
        public int getDia() { return dia; }
        public String getMes() { return mes; }
        public int getAnio() { return anio;  }

        public String toString() { return dia+"/"+mes+"/"+anio; }
}
```

¿Es posible añadir a la clase Persona un método de un sólo parámetro que permita cambiar en el main() la fecha de pepe a 3/3/2003? Si crees que sí hazlo y si no indica el por qué. **(1.00 puntos)**

4) ¿Y si fuera **final** el atributo **fecha** de **Persona (final private Fecha** fecha;) y **no fuera final** los atributos de **Fecha (private int** dia, anio; y **private String** mes;)?. **(1.00 puntos)**

5) Volviendo al código original de inicio de página y sin cambiar absolutamente nada en la clase Fecha ¿Crees que es posible modificar la clase Persona para que en el main(), ponga lo que se ponga, la fecha sólo se puede cambiar un máximo de 2 veces? Si crees que es posible hazlo, y si no indica la razón por la cual no se puede hacer. **(2.00 puntos)**

SOLUCION:

(5 Puntos). Java. Dado el siguiente código:

```
public class Fecha {
	private int dia, anio;
	private String mes;
	public Fecha(int d, String m, int a) { dia=d; mes=m; anio=a; }

	public int getDia() { return dia; }
	public String getMes() { return mes; }
	public int getAnio() { return anio;  }

	public void setFecha(int d, String m, int a) {	dia=d; mes=m; anio=a; }
	public String toString() { return dia+"/"+mes+"/"+anio; }
}

public class Persona {
	private String nombre;
	private Fecha fecha;
	public Persona(String nom, Fecha f) { nombre=nom; fecha=f; }

	public String getNombre() { return nombre; }
	public Fecha getFecha() { return fecha; }
	public String toString() { return nombre+" "+fecha; }
}

public class Prueba {
	public static void main(String[] args) {
		Persona p=new Persona("pepe", new Fecha(2, "feb",2000));
		System.out.println(p.getNombre()+" nacio el "+p.getFecha());

		System.out.println(p);
	}
}
```

1) Sin modificar nada en las clases **Fecha** y **Persona**, ¿es posible en el **main()** cambiar la fecha a 1/1/2001?. Si crees que si añade en el **main()** el código necesario para poder hacerlo. Si crees que no indica el motivo por el que no es posible. **(0.50 puntos)**

```
	public static void main(String[] args) {
		Persona p=new Persona("pepe", new Fecha(2, "feb",2000));
		System.out.println(p.getNombre()+" nacio el "+p.getFecha());

		p.getFecha().setFecha(1, "ene", 2001);
		System.out.println(p);
	}
```

2) ¿Y si fuera final el atributo **fecha** de **Persona**?, es decir, ¿y si fuera **final private Fecha fecha;** en lugar de **private Fecha fecha;**? **(0.50 puntos)**

```
	public static void main(String[] args) {
		Persona p=new Persona("pepe", new Fecha(2, "feb",2000));
		System.out.println(p.getNombre()+" nacio el "+p.getFecha());

		p.getFecha().setFecha(1, "ene", 2001);
		System.out.println(p);
	}
```

3) Si la clase Fecha fuera así y no se pudiera cambiar absolutamente nada…

```
public class Fecha {
	final private int dia, anio;
	final private String mes;

	public Fecha(int d, String m, int a) { dia=d; mes=m; anio=a; }
	public int getDia() { return dia; }
	public String getMes() { return mes; }
	public int getAnio() { return anio;  }

	public String toString() { return dia+"/"+mes+"/"+anio; }
}
```

¿Es posible añadir a la clase Persona un método de un sólo parámetro que permita cambiar en el main() la fecha de pepe a 3/3/2003? Si crees que sí hazlo y si no indica el por qué. **(1.00 puntos)**

```
public class Persona {
	private String nombre;
	private Fecha fecha;
	public Persona(String nom, Fecha f) { nombre=nom; fecha=f; }
	public String getNombre() { return nombre; }
	public Fecha2 getFecha() { return fecha; }
	public String toString() { return nombre+" "+fecha; }
	public void setFecha(Fecha f) { fecha=new Fecha(f.getDia(), f.getMes(), f.getAnio()); }
}                                    //fecha=f; //también valdría ya que la clase fecha es inmutable

public class Prueba {
	public static void main(String[] args) {
		Persona p=new Persona("pepe", new Fecha(2, "feb",2000));
		System.out.println(p.getNombre()+" nacio el "+p.getFecha());

		p.setFecha(new Fecha(3,"mar",2003));
		System.out.println(p);
	}
}
```

4) ¿Y si fuera **final** el atributo **fecha** de **Persona (final private Fecha** fecha;) y **no fuera final** los atributos de **Fecha (private int** dia, anio; y **private String** mes;)?. **(1.00 puntos)**

No es posible hacerlo ya que la clase Fecha es inmutable (no tiene ningún método set que permita modificar los atributos de la fecha) y en la clase Persona no se puede hacer que el atributo fecha apunte a un objeto fecha distinto al ser final

5) Volviendo al código original de inicio de página y sin cambiar absolutamente nada en la clase Fecha ¿Crees que es posible modificar la clase Persona para que en el main(), ponga lo que se ponga, la fecha sólo se puede cambiar un máximo de 2 veces? Si crees que es posible hazlo, y si no indica la razón por la cual no se puede hacer. **(2.00 puntos)**

```
public class Fecha {
	private int dia, anio;
	private String mes;
	public Fecha(int d, String m, int a) { dia=d; mes=m; anio=a; }
	public int getDia() { return dia; }
	public String getMes() { return mes; }
	public int getAnio() { return anio;  }
	public void setFecha(int d, String m, int a) {      dia=d; mes=m; anio=a; }
	public String toString() { return dia+"/"+mes+"/"+anio; }
}

public class Persona {
	private int n; //aunque se inicializa a 0 por defecto lo hago explícitamente en el constructor
	private String nombre;
	private Fecha fecha;
	public Persona(String nom, Fecha f) { n=0; nombre=nom; fecha=f; }
	public String getNombre() { return nombre; }
	public Fecha getFecha() { return fecha; } //debo devolver una copia en vez del original
	public Fecha getFecha() { return new Fecha(fecha.getDia(), fecha.getMes(), fecha.getAnio()); }
	public String toString() { return nombre+" "+fecha; }
	public void setFecha(int d, String m, int a) { //si la nueva fecha es distinta a la anterior...
		if (d!=fecha.getDia() || !m.equals(fecha.getMes()) || a!=fecha.getAnio())
			if (n<2) { //... y no se ha cambiado 2 veces
				fecha.setFecha(d, m, a); //fecha=new Fecha(d,m,a);
				n++;
			}
	}
}

public class Prueba {
	public static void main(String[] args) {
		Persona p=new Persona("pepe", new Fecha(2, "feb",2000));
		System.out.println(p.getNombre()+" nacio el "+p.getFecha());
		System.out.println(p); p.setFecha(3,"mar",2003); //cambia la fecha 1ª vez
		System.out.println(p); p.setFecha(3,"mar",2003); //no cuenta porque la fecha es la misma
		System.out.println(p); p.setFecha(4,"abr",2004); //cambia la fecha 2° vez
		System.out.println(p); p.setFecha(5,"may",2005); //no cambia: ya se ha cambiado 2 veces
		System.out.println(p); p.getFecha().setFecha(7, "jul", 2007); //cambia la copia, no fecha
		System.out.println(p);
	}
}
```