2º Parcial

Bloque I. Preguntas cortas LPOO (C++, C#, Java) 5 puntos

Nombre:_____

DNI: _____ **Titulación:** _____

Se debe tener un mínimo del 40% de la puntuación de este bloque para poder aprobar el examen.

1- Indica si el siguiente enunciado es verdadero o falso justificando la respuesta: "En los LPOO estudiados sólo es posible acceder a las características privadas de una clase en el cuerpo de la clase donde se declaran".

2- Indica si el siguiente enunciado es verdadero o falso justificando la respuesta: "La conversión de tipos en los LPOO estudiados siempre provoca una excepción equivalente a la excepción Java `ClassCastException` si los tipos no son compatibles."

3- Dado el siguiente código C++:

```
void met() throw (runtime_error) {
     throw runtime_error("operación  no válida");
}
```

¿Es equivalente declarar la excepción o no hacerlo? Justifica la respuesta.

4- ¿Existe algún problema en la jerarquía C++ siguiente? En ese caso indica la solución.

```
class A{
protected:
   int at;
   virtual void met()=0;
};

class B: public A{};
class C: public A{};

class D: public B, public C{
private:
  void met();
};
```

```
//fichero.cpp

void D::met(){
   ++at;
}
```

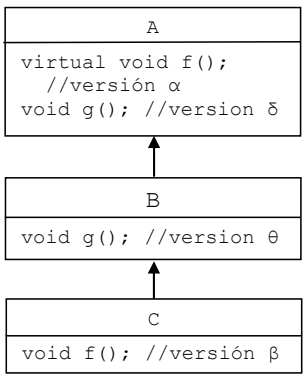
5- Escribe el código de un único constructor C++ equivalente a la siguiente declaración de constructores en Java:

```
public enum Estado {E1, E2, E3}

public class A {
   private Estado estado;
   private int at;

   public A(){ estado = Estado.E1;}
   public A (Estado e, int v){
       estado = e;
       at = v;
   }
}
```

6- ¿Es posible realizar la siguiente asignación en C#? Justifica la respuesta.

```
Persona cliente = "Pepito Pérez";
```

7- Dada la jerarquía de herencia de la izquierda indica qué versión de los métodos `f` y `g` se ejecutará en cada una de las llamadas del código de la derecha, indicando en cada caso si se resuelve por ligadura estática o dinámica.




```
1.   A oa; B ob; C oc;
2.   A *pa; B *pb;
3.   oa = oc;
4.   oa.f();
5.   oa.g();
6.   ob.f();
7.   ob = oc;
8.   ob.f();
9.   oc.g();
10.  pa = new B();
11.  pa->f();
12.  pa->g();
13.  pb = new C();
14.  pb->f();
```

8- La clase C# `Monedero` incluye un único atributo que almacena el dinero disponible en euros. Completa la implementación de la clase `Monedero`, sin añadir ningún atributo más, de manera que el código del programa sea correcto (NOTA: 1 euro son 166 pesetas)

```
class Programa{
  static void Main() {
    Monedero m = new Monedero();
    m.Pesetas = 500;
    Console.WriteLine("Saldo pesetas: "+ m.Pesetas);
  }
}

class Monedero{
  private double euros;
  //Completar









}
```

Previo a las preguntas 9-12: En C#, sea una clase genérica `Bascula` con un tipo genérico `T` que declara un atributo con nombre `elementos` de tipo array de `T`:

9- ¿En qué casos sería correcto el siguiente código? Justifica la respuesta.

```
for (int i = 0; i < 10; i++)
      elementos[i] = new T();
```

10-Sea el método `IndexOf` que nos indica el índice de un elemento en el array. ¿en qué casos sería correcto el siguiente código? Justifica la respuesta.

```
int IndexOf (T elemento) {

      for (int i = 0; i < elementos.Length; i++)
            if (elementos[i] == elemento)
                  return i;
      return -1;
}
```

11-Supongamos que los elementos que almacena la clase genérica tienen un “peso” y queremos añadir un método que calcula el peso de los elementos. ¿Cómo podríamos conseguir implementar esta funcionalidad?

```
int CalcularPeso() {
      int pesoTotal = 0;
      foreach (T elemento in elementos)
                  pesoTotal += elemento.Peso;
      return pesoTotal;
}
```

12-La clase genérica incluye el siguiente método que devuelve un iterador de los elementos que superan un determinado peso. Indica el código que hay que añadir al método:

```
IEnumerable<T> FiltroPeso(int umbral) {

      foreach (T elemento in elementos)
            // Completar




}
```

13-Sean las interfaces `Reseteable` y `Crono` que se definen del siguiente modo:

```
interface Reseteable {
        void Reset();
}
```

```
interface Crono {
     void Start();
     double Stop();
     void Reset();
}
```

Queremos programar la clase `Reloj` que implementa la interfaz `Reseteable`, cuyo método `Reset` deja los valores del reloj en estado inicial y la interfaz `Crono` para reproducir un cronómetro, donde los métodos `Start` y `Stop` inician y detienen el cronómetro y `Reset` lo pone a cero. ¿Existe algún problema en C# para que la clase `Reloj` implemente estas dos interfaces? Justifica la respuesta.

14-Dado el siguiente código en C# donde el método `Transformar` devuelve una matriz de números reales resultado de aplicar una operación sobre los elementos de la matriz receptora de la llamada:

```
Matriz m2 = m1.Transformar( x => x * x );
```

Escribe la declaración del método `Transformar` de la clase `Matriz`. Si en la declaración del método se utiliza algún tipo de datos, escribe la declaración de ese tipo de datos.

15- Escribe el código C++ equivalente al del ejercicio 14, incluyendo la declaración del método `Transformar` y la llamada a dicho método.

16- ¿Son equivalentes las siguientes declaraciones? Justifica la respuesta representando gráficamente el resultado de la creación de cada una de las estructuras de datos.

C++	Java
`Punto* pentagono = new Punto[5];`	`Punto[] pentagono = new Punto[5];`

2º Parcial

Bloque II. Ejercicios — 5 puntos

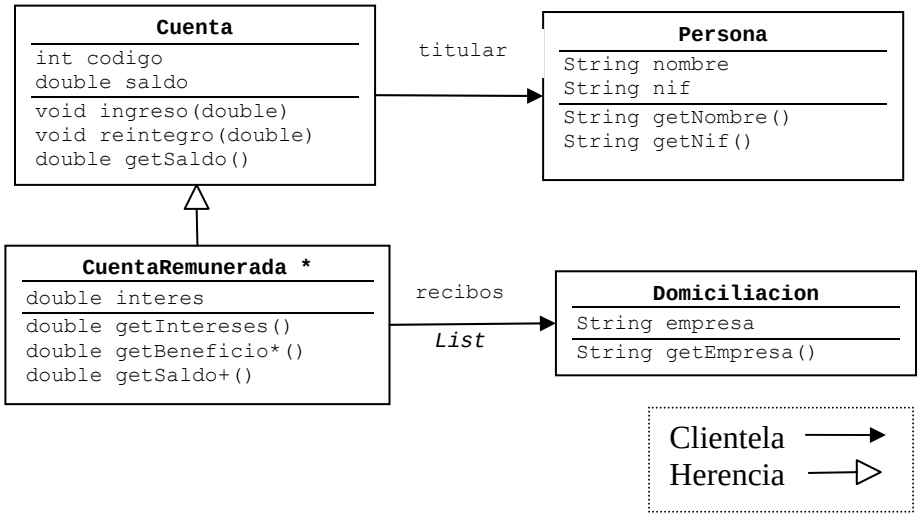
Nombre:_____

DNI: _____ **Titulación:** _____

Se debe tener un mínimo del 40% de la puntuación de este bloque para poder aprobar el examen.

En este apartado se proponen ejercicios de programación que deben ser resueltos utilizando el lenguaje de programación **Java**. Se valorará que los ejercicios cumplan la funcionalidad requerida y que el código cumpla los **criterios de calidad** expresados en el tema 1 de la asignatura: extensibilidad, reutilización, principios de diseño modular, etc.

Para los ejercicios que solicitan expresar gráficamente el código de la aplicación debe utilizarse la siguiente **notación**. Las clases y métodos abstractos se marcarán con *. Un método redefinido se marcará con +.



Ejercicio 1. Queremos implementar el software de los terminales de venta de los comercios. Un comercio pone a la venta **productos** que se caracterizan por su nombre y su precio en euros. Los clientes realizan **compras** de productos. Una compra está formada por **líneas de compra**. Una línea de compra representa la cantidad de elementos de un cierto producto adquiridos en una compra. El total de una compra se calcula como la suma de los precios de cada línea de compra.

Supongamos que un cliente realiza una compra que consiste en 3 cartones de leche (1,2 euros la unidad) y 5 botellas de agua (0,5 euros la unidad). El detalle de la compra sería:

- 3 x Cartón de Leche – 1,2 – 3,6 euros
- 5 x Botella de Agua – 0,5 – 2,5 euros
    - Total Compra: 6,1 euros.

Algunos productos de la tienda pueden estar en promoción. A los productos en promoción se les podrá aplicar descuentos en la compra. Además, en la tienda tenemos productos de marcas blancas (**productos blancos**) que permiten obtener puntos. Cada producto blanco proporciona una cantidad de puntos. Si un producto blanco está en promoción su precio de venta *siempre* será un 10% más barato.

a) (**0'5 ptos**) Representa gráficamente las clases que implementan la funcionalidad de la aplicación siguiendo la notación propuesta.

b) (**0'5 ptos**) Escribe el código del método `getPuntos()` de la clase `Compra`. Los puntos de la compra corresponden a los puntos asociados a los productos blancos.

c) (**0'5 ptos**) Escribe el código de la clase `ProductoBlanco`.

La tienda aplica periódicamente descuentos a las compras. Los descuentos sólo son aplicables a los productos que están en promoción. Actualmente se aplican dos tipos de descuentos: 3x2 y segunda unidad a mitad de precio. El descuento 3x2 consiste en pagar dos unidades cuando se compran 3 de un mismo producto. Por ejemplo, si en una línea de compra se adquieren 4 productos, 1 de ellos es gratuito, y si se adquieren 6, 2 son gratuitos.

A continuación se muestra el resultado de la compra aplicando el descuento 3x2 (los dos productos están en promoción). En la primera línea de compra una unidad sale gratis y en la segunda también:

- 3 x Cartón de Leche – 1,2 – 2,4 euros
- 5 x Botella de Agua – 0,5 – 2 euros
    - Total Compra: 4,4 euros.

d) (**0'75 ptos**) Escribe el código de un método en la clase `Compra` que calcule el total de una compra aplicando un descuento. Nótese que los descuentos son aplicados a las líneas de compra. Además, la solución debe ser válida para cualquier tipo de descuento que se introduzca más adelante en la aplicación. Ilustra la solución implementando un descuento 3x2.

e) (**0'25 ptos**) Supongamos que tenemos un objeto `compra`. Escribe el código que calcula el total de la compra aplicando el descuento 3x2.

Ejercicio 2. Queremos implementar un `Planificador` de tareas de ejecución de código que dé prioridad a aquellas que se ejecutan más rápido y penalice las tareas más lentas. Una `Tarea` se caracteriza por aplicar un *algoritmo* cuando se invoca el método `ejecutar`. Además, este método se encarga de calcular el tiempo en milisegundos consumido en la ejecución del algoritmo. Ejemplos de tareas: poner a un valor determinado los elementos de una lista de enteros, compactar un array de cadenas, multiplicar dos matrices, etc.

El planificador de tareas ofrece métodos para registrar y eliminar tareas de ejecución. Dispone del método `comenzar` encargado de ejecutar todas las tareas. Este método ejecuta una a una cada tarea. Durante la ejecución del método `comenzar` se identifica la tarea que ha consumido más tiempo. Esta tarea será recordada por el planificador para no ejecutarla la próxima vez que se invoque al método `comenzar`. Nótese que la tarea más lenta podrá ser diferente en cada ejecución.

a) (**1'5 ptos**) Escribe el código de la clase `Tarea` que cumpla los requisitos expresados anteriormente y el de una tarea (`TareaReset`) encargada de establecer un valor -1 en todos los elementos de una lista de enteros.
   Nota: la clase `System` ofrece el método de clase `currentTimeMillis` que devuelve un entero (long) que corresponde con los milisegundos del instante actual. Utiliza este método para calcular los milisegundos de ejecución de una tarea.

b) (**1 pto**) Durante la ejecución de una tarea pueden producirse errores. Se identifican dos tipos de problemas de ejecución: temporales y permanentes. Un problema temporal sucede cuando una tarea no puede completar su ejecución por razones transitorias, como por ejemplo, un fallo en la comunicación de red. Un problema permanente significa que la tarea no puede volver a ejecutarse.

   - Explica qué cambios habría que introducir en la clase Tarea del apartado a) para la gestión de los errores de ejecución.
   - El planificador de tareas da un tratamiento diferente a cada tipo de error. Si una tarea produce un error temporal, reintenta su ejecución una segunda vez. Si la tarea sigue fallando, la elimina de la lista de tareas para no volver a ejecutarla. En cambio, si una tarea provoca un error permanente, el planificador la elimina definitivamente de la lista de tareas. Escribe el código del método `comenzar` de la clase `Planificador` que cumpla los requisitos de la clase y tenga en cuenta estas situaciones de error.