Apellidos:

Nombre: D.N.I.:

INTRODUCCIÓN A LA PROGRAMACIÓN - E. P. DE INGENIERÍA DE GIJÓN
19 DE ENERO DE 2018

---

[1] (0.5 puntos) Declara una constante no estática para la Constante de Legendre (valor 1.08366)

```
final  double  LEGENDRE = 1.08366;
```

[2] (1,5 p.) Dada la siguiente definición de variables y sus valores iniciales,

```
int x=9, y=2; double f=3.5; boolean b=true;
```

indica para las siguientes expresiones, si son o no correctas (SI/NO), en caso de resultar incorrectas JUSTIFICA por qué lo son, y en el caso de ser correctas indica el TIPO y el VALOR que producen:

`++f`	**X** Sí ☐No Tipo-Valor/Motivo: **double, 4.5**
`b < x`	☐Sí **X** No T-V/M: **Incorrecto el operador < no se puede aplicar a booleanos**
`f = x/y`	**X** Sí ☐No T-V/M: **double 4.0**
`x>y && b`	**X** Sí ☐No T-V/M: **boolean, true**
`x <> y`	☐Sí **X** No T-V/M: **Incorrecto, el operador <> no existe, existe != (distinto)**
`x+b`	☐Sí **X** No T-V/M: **Incorrecto el operador + no se puede aplicar a booleanos**

[3] (2 p.) Supuesto que **N > 0** y **par**, diseña los algoritmos iterativos que se indican a continuación.

Bucle **for** que imprima una secuencia de números en [1, N] donde cada número sea el doble del anterior. Ej N=18: 1 2 4 8 16	Bucle **do/while** para imprimir la secuencia N N/2 (N/2)/2 … hasta obtener un impar (no incluido) Ej: N=40: 40 20 10	Bucle **while** para imprimir la secuencia de números de N/2 a 1, ambos incluidos Ej: N=8: 4 3 2 1

`for(int i=1;i<=N;i*=2)`
`  System.out.println(i);`
	`int i=N;`
	`do {`
	`  System.out.println(i);`
	`  i/=2;`
	`}`
	`while ( i % 2 == 0 )`
		`int i=N/2;`
		`while ( i >= 1 ){`
		`  System.out.println(i);`
		`  i--;`
		`}`

[4] (3 p.) Escribir el método **rotarColumnaArriba**, privado y estático, que recibe una matriz de caracteres no vacía y el índice de una de sus columnas y modifica la matriz rotando las componentes de esa columna una posición hacia arriba. Se supone que el índice de la columna es correcto.

Ejemplo:

rotarColumnaArriba(M,0)

M

'O'	'O'	'O'	'X'
'X'	'X'	'X'	'O'
'O'	'O'	'X'	'O'
'X'	'X'	'O'	'X'

M

'X'	'O'	'O'	'X'
'O'	'X'	'X'	'O'
'X'	'O'	'X'	'O'
'O'	'X'	'O'	'X'

```
private static void rotarColumnaArriba(char[][] M, int col) {
//Si la columna pudiera ser incorrecta añadiríamos un if aquí
    //Guardamos la componente de la 1ª fila
    char temp = M[0][col];
    //Tratamiento secuencial: movemos cada componente a la
    //posición de arriba, salvo la componente de la 1ª fila
    //Secuencia: i:1..M.length-1
    //1er elemento: i=1, movemos m[1][col] a m[0][col]
    //Sgte. elto.: i++, movemos m[i][col] a m[i-1][col]
    //Fin: i==M.length
    for (int i=1; i < M.length; i++)
        M[i-1][col]=M[i][col];
    //Ponemos la 1a componente de la columna en la última fila
    M[M.length-1][col]=temp;
}
```

[5] (1.5 p.) Sustituye las sentencias siguientes por otro trozo de código equivalente (que produzca los mismos resultados) pero que sea más corto, más fácil de entender y/o más eficiente.

`if (x<18)  d--;`
`else if (x>65)  d--;`
`else if (x<=65) d++;`
	`if (y<-10)`
	`  if (y>10) d=d/10;`
	`  else  d=d*10;`
	`else  d=d*10;`
		`if (z>=5)  {`
		`  x++; y*=2;}`
		`else  {`
		`  x++; y/=2;}`

`if (x<18)||(x>65)` **`  d--;`** **`else`** **`  d++;`**	**`d=d*10;`**	**`x++;`** **`if (z>=5)`** **`  y*=2;`** **`else  y/=2;`**

[6] (3.5 p.) El Club de Tenis de Gijón va a organizar un torneo en el que los precios de la inscripción son los que se muestran en la tabla adjunta. Se pide hacer un **programa de consola completo** (importando las clases necesarias) que pida por teclado el tipo de competición y de jugador/a y nos indique el precio de la inscripción. Se valorará especialmente el diseño del esquema condicional, y la conveniencia y el correcto uso de las variables y tipos elegidos para representar los datos. El usuario introducirá los datos correctamente, no se debe incluir código para comprobar si son correctos.

	Alevín	**Infantil**	**Junior**	**Senior**
Individuales	*20*	*20*	*25*	*25*
Dobles	*10*	*10*	*20*	*20*
Dobles Mixtos	*10*	*10*	*15*	*15*

```
import java.util.Scanner;

public class Inscripción {
    public static void main(String[] args) {
        // Scanner
        Scanner teclado = new Scanner(System.in);
        // Variables
        System.out.print("(a)levín/(i)nfantil/(j)unior/(s)enior");
        char participante = teclado.next().charAt(0);
        System.out.print("Prueba: indi(v)iduales/(d)obles/(m)ixtos");
        char prueba = teclado.next().charAt(0);

        // Cálculo del precio
        if (participante == 'a' || participante == 'i')
            if (prueba == 'v')
                precio = 20;
            else
                precio = 10;
        else
            switch(prueba) { // también se puede usar if/else
                case 'v': precio = 25; break;
                case 'd': precio = 20; break;
                default: precio = 15; break;
            }

        System.out.printf("El precio es %d\n", precio);
    }
}
```

[7] (2 p.) Escribir el método estático **calculaPicos** que reciba como parámetro un vector de enteros y que devuelva el número de picos que contiene. Un pico es un elemento que es estrictamente mayor que las componentes anterior y siguiente.

Ejemplo:

1	3	0	5	7	9	4	1	5

Retorna: 2. Hay 2 picos: 3, y 9

```
public static int calculaPicos(int[] v) {
   //Recorrido en secuencia de todas los tríos de componentes
   //El índice (i) representa el elemento central del trío
   //1er elemento: i=1(elemento central) trío: v[0] v[1] v[2]
   //Sgte. elto.: i++, trío: v[i-1] v[i] v[i+1]
   //Fin: i<=v.length-1,
   //    trío: v[v.length-3], v[v.length-2] y v[v.length-1]
   int picos=0;     //para contar el nº de picos
   for (int i=1; i<=v.length-2; i++)
      if ( v[i]>v[i-1] && v[i]>v[i+1] )
         picos++;
   //Se retorna el valor de la variable picos
   return picos;
}
```

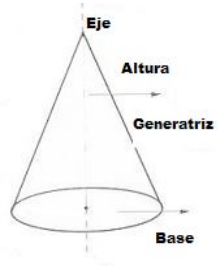
[8] (3 p.) Escribir el método público y estático **dobleLetra** que reciba como parámetro una palabra (representada con una cadena de caracteres) y que devuelva cierto si en la palabra al menos una misma letra aparece en dos posiciones consecutivas. Debe devolver falso en caso contrario.

Ejemplos: "lluvia" "acción" "arrollar"

Restricción: **no se pueden emplear** las sentencias break y return dentro de un bucle.

```
public static boolean dobleLetra(String s) {
   //Buscar un carácter que esté repetido
   //Comparamos el carácter i, con el anterior i-1
   //1er elemento: i=1, comparamos s.charAt(0) y s.charAt(1)
   //Sgte. elto.: i++, comparamos s.charAt(i-1) y s.charAt(i)
   //Últ. elto.: i=s.length()-1,  comparamos
   //          s.charAt(s.length()-2) y s.charAt(s.length()-1)
   //Buscamos: s.charAt(i) == s.charAt(i-1)
   int i=1;
   while ( i < s.length() && s.charAt(i-1) != s.charAt(i) )
      i++;
   if ( i < s.length() )  return true;
   else                   return false;
   //también: return (i<s.length());
}
```

[9] (4 p.) Escribir la clase **Cono** para representar conos. Un Cono se identifica por su radio y su altura, que son valores de tipo real. Deben implementarse los siguientes métodos:



- Tres constructores: 1) sin parámetros (ambos valores valdrán 1.0), con dos parámetros reales (uno para inicializar cada valor), y el constructor copia.
- Métodos set() y get() para los atributos, con la restricción de que ambos deben tener siempre una cantidad mayor que 0.
- Métodos calculaGeneratriz(), calculaÁrea() y calculaVolumen() (ver fórmulas).

$$g = \sqrt{a^2 + r^2} \quad area = \pi r^2 + \pi r g \quad volumen = \frac{\pi r^2 a}{3}$$

- Método esMayor(), que recibe como parámetro otro objeto Cono y devuelve cierto si el objeto que invoca el método tiene un volumen mayor que el del objeto pasado como argumento.

Notas: Se tendrá en cuenta la corrección de la implementación y que la clase se escriba de forma que se faciliten futuras modificaciones. No es necesario poner comentarios Javadoc.

```
public class Cono {
   //Atributos
   private final static double PI=3.1416;
   private double radio;
   private double altura;
   //Constructores. Se escriben usando this, de forma que
   //todas las versiones acaban realizándose a través del
   //constructor con dos parámetros double
   public Cono() {
      this(1.0,1.0);  //valores por defecto 1.0
   }
   //se obtienen los valores de los radios usando los métodos
   //get()
   public Cono(Cono c) {
      this(c.getRadio(),c.getAltura());
   }
   //se inicializan los atributos usando los método set()
   public Cono(double radio, double altura) {
      this.setRadio(radio);
      this.setAltura(altura);
   }
   //Métodos set() y get(). Los métodos set() comprueban que
   //los valores de los atributos sean positivos, y los get()
   //se limitan a devolver el valor de los atributos.
   public void setRadio(double radio) {
      if ( radio > 0 ) this.radio=radio;
   }
   public double getRadio(){
      return this.radio;
   }
   public void setAltura(double altura) {
      if (altura > 0) this.altura=altura;
   }
   public double getAltura(){
      return this.altura;
   }
   //Métodos para calcular la generatriz, el area y el volumen
   public double calculaGeneratriz() {
      return Math.sqrt(this.getRadio()*this.getRadio()+
                       this.getAltura()*this.getAltura());
   }
   public double calculaÁrea() {
      return PI*(this.getRadio()*this.getRadio()+
                 this.getRadio()*this.calculaGeneratriz());
   }
   public double calculaVolumen() {
      return PI*this.getRadio()*this.getRadio()
               *this.getAltura()/3.0;
   }
   //Un cono es mayor que otro cuando su volumen es mayor. Se
   //retorna la comparación directa entre las llamadas al
   //método calculaVolumen() con el objeto que hace la llamada
   //(this) y el que se pasa como argumento (c)
   public boolean esMayor(Cono c) {
       return this.calculaVolumen() > c.calculaVolumen();
   }
}
```

[10] (3 p.) Escribir la clase **Tarjeta** que representa los datos de una tarjeta de crédito y las operaciones que se pueden realizar con ella. Los datos que tiene son:

- El **nombre** del titular: cadena de caracteres.
- El **número** de la tarjeta: 16 dígitos.
- La **fecha** de validez: mes y año. Deben usarse obligatoriamente un objeto de la clase Fecha cuya especificación UML es la mostrada.
- El **límite**: cantidad en euros máxima que se puede cargar, y
- El **saldo**: cantidad gastada (valor positivo), o abonada (valor negativo).

Fecha
- mes: int - año: int
+ Fecha(int,int) + getMes(): int + setMes(int) + getAño(): int + setAño(int) + toString(): String

Los métodos a implementar deben hacer funcionar el código siguiente:

```
Tarjeta t1=new Tarjeta("Juan",6023567921123217L,7, 2016 , 2500.0);
                      //nombre,      n°        ,mes, año ,límite, saldo=0
t1.cargarCompra(359.99);     //saldo = 359.99
t1.abonarDevolución(39.95); //saldo = 320.04
t1.cargarCompra(2250.0);     //supera el límite, no se hace!!! saldo=320.04
System.out.print(t1);        //Nombre: Juan
                             //Número: 6023567921123217
                             //Valida: 07/2016
                             //Límite: 2500.00 €    Saldo: 320.04 €
```

Notas: No hay que implementar los métodos get() y set(), **pero debes usarlos** como si lo estuviesen para acceder a los atributos en el resto de métodos. En el constructor, el campo Fecha **NO** hay que inicializarlo con su método set; si lo inicializas de ese modo debes programar dicho método. En cargarCompra() y abonarDevolución() tiene que comprobarse que la cantidad es siempre positiva.

```
public class Tarjeta {
   private String nombre;   //Atributos
   private long número;
   private Fecha fecha;
   private double saldo, límite;
   //Constructor
   public Tarjeta(String nombre, long número,
                  int mes, int año, double límite) {
       this.setNombre(nombre);
       this.setNúmero(número);
       this.fecha=new Fecha(mes,año);
       this.setSaldo(0);
       this.setLímite(límite);
   }
   //Se comprueba que la cantidad sea positiva y que no se
   //excede el límite al hacer la operación
   public void cargarCompra(double cantidad) {
       if ( cantidad + this.getSaldo() <= this.getLímite() &&
            cantidad > 0 )
           this.setSaldo( this.getSaldo() + cantidad );
   }
   //Se comprueba que la cantidad es positiva. No pasa nada si
   //tras la operación el saldo es negativo, eso es válido.
   public void abonarDevolución(double cantidad) {
       if(cantidad>0) this.setSaldo(this.getSaldo()-cantidad);
   }
   //imprime todos los datos
   @Override
   public String toString(){
      String s=String.format("Nombre: %s",this.getNombre());
      s+=String.format("\nNúmero: %d",this.getnúmero);
      s+=String.format("\nValida %s",this.getFecha());
      s+=String.format("\nLímite %f,Saldo %f",
                       this.getLímite(),this.getSaldo());
      return s;
   }
}
```