Ejercicios Programación Orientada a Objetos (POO)

1. Escribe una clase Cuenta para representar una cuenta bancaria. Los datos de la cuenta son: nombre del cliente (String), número de cuenta (String), tipo de interés (double) y saldo (double).

La clase contendrá los siguientes métodos:

Constructor por defecto

Constructor con todos los parámetros

Constructor copia.

Métodos setters/getters para asignar y obtener los datos de la cuenta.

Métodos *ingreso* y *reintegro*. Un ingreso consiste en aumentar el saldo en la cantidad que se indique. Esa cantidad no puede ser negativa. Un reintegro consiste en disminuir el saldo en una cantidad pero antes se debe comprobar que hay saldo suficiente. La cantidad no puede ser negativa. Los métodos ingreso y reintegro devuelven true si la operación se ha podido realizar o false en caso contrario.

Método *transferencia* que permita pasar dinero de una cuenta a otra siempre que en la cuenta de origen haya dinero suficiente para poder hacerla. Ejemplo de uso del método transferencia:
cuentaOrigen.transferencia(cuentaDestino, importe);
que indica que queremos hacer una transferencia desde cuentaOrigen a cuentaDestino del importe indicado.

Prueba el funcionamiento de la clase Cuenta con este main:

```
public static void main(String[] args) {

Scanner sc = new Scanner(System.in);
String nombre, numero;
double tipo, importe;

//se crea objeto cuenta1 sin parámetros
//se ejecuta el constructor por defecto
Cuenta cuenta1 = new Cuenta();

System.out.print("Nombre : ");
nombre = sc.nextLine();
System.out.print("Número de cuenta : ");
numero = sc.nextLine();
System.out.print("Tipo de interes : ");
tipo = sc.nextDouble();
System.out.print("Saldo: ");
importe = sc.nextDouble();

cuenta1.setNombre(nombre);
cuenta1.setNumeroCuenta(numero);
cuenta1.setTipoInteres(tipo);
cuenta1.setSaldo(importe);

//se crea el objeto cuenta2 con los valores leidos por teclado
//se ejecuta el constructor con parámetros
Cuenta cuenta2 = new Cuenta("Juan Ferrández Rubio", "12345678901234567890", 1.75, 300);

//se crea cuenta3 como copia de cuenta1
//se ejecuta el constructor copia
Cuenta cuenta3 = new Cuenta(cuenta1);
```

```
//mostrar los datos de cuenta1
System.out.println("Datos de la cuenta 1");
System.out.println("Nombre del titular: " + cuenta1.getNombre());
System.out.println("Número de cuenta: " + cuenta1.getNumeroCuenta());
System.out.println("Tipo de interés: " + cuenta1.getTipoInteres());
System.out.println("Saldo: " + cuenta1.getSaldo());
System.out.println();

//se realiza un ingreso en cuenta1
cuenta1.ingreso(4000);

//mostrar el saldo de cuenta1 después del ingreso
System.out.println("Saldo: " + cuenta1.getSaldo());

//mostrar los datos de cuenta2
System.out.println("Datos de la cuenta 2");
System.out.println("Nombre del titular: " + cuenta2.getNombre());
System.out.println("Número de cuenta: " + cuenta2.getNumeroCuenta());
System.out.println("Tipo de interés: " + cuenta2.getTipoInteres());
System.out.println("Saldo: " + cuenta2.getSaldo());
System.out.println();

//mostrar los datos de cuenta3
System.out.println("Datos de la cuenta 3");
System.out.println("Nombre del titular: " + cuenta3.getNombre());
System.out.println("Número de cuenta: " + cuenta3.getNumeroCuenta());
System.out.println("Tipo de interés: " + cuenta3.getTipoInteres());
System.out.println("Saldo: " + cuenta3.getSaldo());
System.out.println();

//realizar una transferencia de 10€ desde cuenta3 a cuenta2
cuenta3.transferencia(cuenta2, 10);

//mostrar el saldo de cuenta2
System.out.println("Saldo de la cuenta 2");
System.out.println("Saldo: " + cuenta2.getSaldo());
System.out.println();

//mostrar el saldo de cuenta3
System.out.println("Saldo de la cuenta 3");
System.out.println("Saldo: " + cuenta3.getSaldo());
System.out.println();
}
```

2. Crea una clase llamada Contador que contenga un único atributo entero llamado *cont*.

La clase tendrá los siguientes constructores:

Constructor por defecto

Constructor con parámetros para inicializar el contador con un valor no negativo. Si el valor inicial que se recibe es negativo el contador tomará el valor cero como valor inicial.

Constructor copia.

Además de los métodos getter y setter, la clase contendrá los métodos:

incrementar: incrementa el contador en una unidad.

decrementar: decrementa el contador en una unidad. El contador nunca podrá tener un valor negativo. Si al decrementar se alcanza un valor negativo el contador toma el valor cero.
Una vez creada la clase, escribe un método main para probar la clase.

Un método main para probar la clase puede ser este:

```
public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);

    //Utilizar el constructor por defecto
    Contador contador1 = new Contador();

    int n;
    System.out.println("Introduce valor para inicializar el contador: ");
    n = sc.nextInt();

    //asignar un valor al contador
    contador1.setCont(n);
    //incrementar el contador
    contador1.incrementar();
    //mostrar el valor actual
    System.out.println(contador1.getCont());

    contador1.incrementar();
    contador1.incrementar();

    //mostrar el valor actual
    System.out.println(contador1.getCont());

    //restar 1 al valor del contador
    contador1.decrementar();

    //mostrar el valor actual
    System.out.println(contador1.getCont());

    //crear un nuevo objeto Contador con valor inicial 10
    Contador contador2 = new Contador(10);

    //incrementar y decrementar el contador2 y mostrar su valor
    contador2.incrementar();
    System.out.println(contador2.getCont());
```

```
    contador2.decrementar();
    System.out.println(contador2.getCont());

    //crear un objeto Contador utilizando el constructor copia
    //se crea el objeto contador3 como copia de contador2
    Contador contador3 = new Contador(contador2);

    //mostrar el valor de contador3
    System.out.println(contador3.getCont());
}
```

3. Crea una clase llamada Libro que guarde la información de cada uno de los libros de una biblioteca. La clase debe guardar el título del libro, autor, número de ejemplares del libro y número de ejemplares prestados. La clase contendrá los siguientes métodos:

Constructor por defecto.

Constructor con parámetros.

Métodos Setters/getters

Método *préstamo* que incremente el atributo correspondiente cada vez que se realice un préstamo del libro. No se podrán prestar libros de los que no queden ejemplares disponibles para prestar. Devuelve true si se ha podido realizar la operación y false en caso contrario.

Método *devolución* que decremente el atributo correspondiente cuando se produzca la devolución de un libro. No se podrán devolver libros que no se hayan prestado. Devuelve true si se ha podido realizar la operación y false en caso contrario.

Método *toString* para mostrar los datos de los libros. Este método se heredada de Object y lo debemos modificar (override) para adaptarlo a la clase Libro.
Escribe un programa para probar el funcionamiento de la clase Libro.

Un programa para probar la clase Libro puede ser este:

```
public static void main(String[] args) {
	Scanner sc = new Scanner(System.in);
	String titulo, autor;
	int ejemplares;

	//se crea el objeto libro1 utilizando el constructor con parámetros
	Libro libro1 = new Libro("El quijote", "Cervantes", 1, 0);
	//se crea el objeto libro2 utilizando el constructor por defecto
	Libro libro2 = new Libro();

	System.out.print("Introduce titulo: ");
	titulo = sc.nextLine();
	System.out.print("Introduce autor: ");
	autor = sc.nextLine();
	System.out.print("Numero de ejemplares: ");
	ejemplares = sc.nextInt();

	//se asigna a libro2 los datos pedidos por teclado.
	//para ello se utilizan los métodos setters
	libro2.setTitulo(titulo);
	libro2.setAutor(autor);
	libro2.setEjemplares(ejemplares);

	//se muestran por pantalla los datos del objeto libro1
	//se utilizan los métodos getters para acceder al valor de los atributos
	System.out.println("Libro 1:");
	System.out.println("Titulo: " + libro1.getTitulo());
	System.out.println("Autor: " + libro1.getAutor());
	System.out.println("Ejemplares: " + libro1.getEjemplares());
	System.out.println("Prestados: " + libro1.getPrestados());
	System.out.println();

	//se realiza un préstamo de libro1. El método devuelve true si se ha podido
	//realizar el préstamo y false en caso contrario
```

```
      if (libro1.prestamo()) {
         System.out.println("Se ha prestado el libro " + libro1.getTitulo());
      } else {
         System.out.println("No quedan ejemplares del libro " + libro1.getTitulo() + " para
prestar");
      }

      //se realiza una devolución de libro1. El método devuelve true si se ha podido
      //realizar la devolución y false en caso contrario
      if (libro1.devolucion()) {
         System.out.println("Se ha devuelto el libro " + libro1.getTitulo());
      } else {
         System.out.println("No hay ejemplares del libro " + libro1.getTitulo() + " prestados");
      }

      //se realiza otro préstamo de libro1
      if (libro1.prestamo()) {
         System.out.println("Se ha prestado el libro " + libro1.getTitulo());
      } else {
         System.out.println("No quedan ejemplares del libro " + libro1.getTitulo() + " para
prestar");
      }

      //se realiza otro préstamo de libro1. En este caso no se podrá realizar ya que
      //solo hay un ejemplar de este libro y ya está prestado. Se mostrará por
      //pantalla el mensaje No quedan ejemplares del libro…
      if (libro1.prestamo()) {
         System.out.println("Se ha prestado el libro " + libro1.getTitulo());
      } else {
         System.out.println("No quedan ejemplares del libro " + libro1.getTitulo() + " para
prestar");
      }
      //mostrar los datos del objeto libro1
      System.out.println("Libro 1:");
      System.out.println("Titulo: " + libro1.getTitulo());
      System.out.println("Autor: " + libro1.getAutor());
      System.out.println("Ejemplares: " + libro1.getEjemplares());
      System.out.println("Prestados: " + libro1.getPrestados());
      System.out.println();

      //mostrar los datos del objeto libro2
      System.out.println("Libro 2:");
      System.out.println("Titulo: " + libro2.getTitulo());
      System.out.println("Autor: " + libro2.getAutor());
      System.out.println("Ejemplares: " + libro2.getEjemplares());
      System.out.println("Prestados: " + libro2.getPrestados());
      System.out.println();
   }
```

4. Escribe una clase para representar fracciones. La clase tendrá dos atributos de tipo int: num (numerador) y den (denominador). La clase debe contener los constructores y métodos adecuados para que este método main funcione de forma correcta:

```
public static void main(String[] args) {
      // Se crean 4 fracciones
      Fraccion f1 = new Fraccion(1, 4); // Fracción 1/4
      Fraccion f2 = new Fraccion(1, 2); // Fracción 1/2
      Fraccion f3 = new Fraccion(); // Fracción 0/1
      Fraccion f4 = new Fraccion(4); // Fracción 4/1
      // operaciones aritméticas con esas fracciones
      Fraccion suma = f1.sumar(f2);
      Fraccion resta = f1.restar(f3);
      Fraccion producto = f1.multiplicar(f4);
      Fraccion cociente = f1.dividir(f2);
      //mostrar resultados
      System.out.println(f1 + " + " + f2 + " = " + suma);
      System.out.println(f1 + " - " + f3 + " = " + resta);
      System.out.println(f1 + " * " + f4 + " = " + producto);
      System.out.println(f1 + " / " + f2 + " = " + cociente);
}
```

La ejecución del método main debe mostar por pantalla lo siguiente:

```
1/4 + 1/2 = 3/4
1/4 - 0/1 = 1/4
1/4 * 4/1 = 1/1
1/4 / 1/2 = ½
```

Las fracciones se deben mostrar siempre simplificadas. Para ello la clase Fraccion debe contener un método privado *simplificar()* que actuará de la siguiente forma:

Para simplificar una fracción primero hay que hallar el máximo común divisor del numerador y del denominador. Una vez hallado se divide el numerador y el denominador por este número. Para calcular el máximo común divisor podemos usar este método mcd que calcula y devuelve el máximo común divisor del numerador y del denominador utilizando el método de Euclides

```
//Cálculo del máximo común divisor
private int mcd(){
    int u=Math.abs(num);
    int v=Math.abs(den);
    if(v==0){
        return u;
    }
    int r;
    while(v!=0){
        r=u%v;
        u=v;
        v=r;
    }
    return u;
}
```

5. Escribe una clase Complejo que modele el comportamiento de los números complejos.
Un número complejo, es una entidad matemática que viene dada por un par de números reales, el primero *a* se denomina la parte real y al segundo *b* la parte imaginaria.
Se representa escribiendo las dos partes del número entre paréntesis (a, b) o también de la forma a + bi.
La i se denomina unidad imaginaria, representa la raíz cuadrada de -1.
La clase Complejo tendrá dos datos privados de tipo double: parte real y parte imaginaria.
La clase Complejo contendrá un constructor por defecto que inicializará a 0 los atributos y un constructor con dos parámetros correspondientes a los valores de la parte real e imaginaria a asignar al nuevo objeto.
Contendrá, además de los setters y getters, los siguientes métodos:

sumar para sumar dos números complejos.
(a, b) + (c, d) = (a + c, b + d);

restar para restar dos números complejos.
(a, b) - (c, d) = (a - c, b – d);

multiplicar para multiplicar dos números complejos:
(a, b) * (c, d) = (a*c – b*d, a*d + b*c)

multiplicar para multiplicar un número complejo por un número double:
(a, b) * n = (a * n, b * n)

dividir para dividir dos números complejos:
(a, b) / (c, d) = ((a*c + b*d) / (c^2 + d^2) , (b*c – a*d) / (c^2 + d^2))

Todos los métodos anteriores devuelven el objeto número complejo resultado de la operación.
La clase contendrá además un método **toString** para mostrar el número complejo de la siguiente forma: (parte real, parte imaginaria) y un método **equals** que compruebe si dos números complejos son iguales o no.
Una vez creada la clase, escribe un programa para probar la clase. Un ejemplo de main podría ser este:

```
public static void main(String[] args) {
      // declaración de números complejos
      Complejo c1 = new Complejo(1.0, 1.0);
      Complejo c2 = new Complejo(2.0, 2.0);
      Complejo c3;
      // operadores aritméticos
      c3 = c1.sumar(c2);
      System.out.println(c1 + " + " + c2 + " = " + c3);
      c3 = c1.restar(c2);
      System.out.println(c1 + " - " + c2 + " = " + c3);
      c3 = c1.dividir(c2);
      System.out.println(c1 + " / " + c2 + " = " + c3);
      c3 = c1.multiplicar(c2);
      System.out.println(c1 + " * " + c2 + " = " + c3);
      c3 = c1.multiplicar(3.5);
      System.out.println(c1 + " * 3.5 = " + c3);
      if (c2.equals(c3)) {
         System.out.println(c2 + " igual que " + c3);
      } else {
         System.out.println(c2 + " distinto que " + c3);
      }
}
```

Si la clase Complejo está bien diseñada, este programa debe mostrar el siguiente resultado por pantalla:

(1.0, 1.0) + (2.0, 2.0) = (3.0, 3.0)
(1.0, 1.0) - (2.0, 2.0) = (-1.0, -1.0)
(1.0, 1.0) / (2.0, 2.0) = (0.5, 0.0)
(1.0, 1.0) * (2.0, 2.0) = (0.0, 4.0)
(1.0, 1.0) * 3.5 = (3.5, 3.5)
(2.0, 2.0) distinto que (3.5, 3.5)

6. Crear una Clase Fecha en Java. La clase tendrá tres atributos privados dia, mes y año de tipo int. La clase contendrá los siguientes métodos:

Constructor por defecto.

Constructor con tres parámetros para crear objetos con valores iniciales.

Métodos set y get para asignar y obtener los valores de los atributos de la clase.

Método fechaCorrecta() que comprueba si la fecha es correcta. Devuelve un valor de tipo boolean indicando si la fecha es correcta o no. Este método a su vez utilizará un método privado de la clase llamado esBisiesto que calcula si el año es o no bisiesto. El método esBisiesto devuelve true si el año es bisiesto y false si no lo es.

Método diaSiguiente() que cambia la fecha actual por la del día siguiente. El objeto de la clase Fecha al que se le aplique este método deberá quedar siempre en un estado consistente, es decir, la nueva fecha deberá ser correcta.

Modificar el método toString() heredado de Object para mostrar las fechas de la forma dd-mm-aaaa. El día y el mes se deben mostrar con dos cifras. Si el dia o el mes tienen solo una cifra se escribirá un cero delante. Por ejemplo si la fecha es dia=1, mes=6, año= 2015 la fecha que se mostrará será: 01-06-2015

Escribe un programa para probar la clase Fecha. El método diaSiguiete() pruébalo dentro de un bucle que imprima la fecha durante cada iteración del bucle.

Ejemplo de ejecución del programa:

Introduce fecha:
dia: 28
mes: 12
año: 2015
Fecha introducida: 28-12-2015
Los 10 días siguientes son:
29-12-2015
30-12-2015
31-12-2015
01-01-2016
02-01-2016
03-01-2016
04-01-2016
05-01-2016
06-01-2016
07-01-2016

7. Crea una clase Empleado que tenga los siguientes atributos privados:

Nif.
Nombre.
Sueldo base.
Horas extra realizadas en el mes.
Tipo de IRPF (%).
Casado o no.
Número de hijos.
Importe de la hora extra. Este será un **atributo static o atributo de clase.**

Los objetos Empleado se podrán crear con un constructor por defecto o con un constructor con un solo parámetro correspondiente al DNI.
Además de los métodos getter/setter la clase Empleado tendrá estos métodos:

Método para el cálculo del complemento correspondiente a las horas extra realizadas.

Método para calcular el sueldo bruto (sueldo base + complemento por horas extras)

Método para calcular las retenciones por IRPF. El porcentaje de IRPF se aplica sobre el sueldo bruto, teniendo en cuenta que el porcentaje que hay que aplicar es el tipo menos 2 puntos si el empleado está casado y menos 1 punto adicional por cada hijo que tenga.

Método toString() para mostrar los datos de los empleados de la siguiente forma:

```
12345678A Lucas Guerrero Arjona
Sueldo Base: 1150.0
Horas Extras: 4
tipo IRPF: 15.0
Casado: S
Número de Hijos: 2
```

Una vez creada la clase Empleado, la utilizaremos en un programa que lea empleados y los guarde en un array estático. El número total de empleados se pide por teclado. El número máximo de empleados es de 20.

Después de leer los datos de los empleados se pedirá que se introduzca el importe correspondiente al pago por hora extra asignándoselo al atributo estático de la clase.

A continuación el programa mostrará:

- El empleado que más cobra y el que menos
- El empleado que cobra más por horas extra y el que menos.
- Todos los empleados ordenados por salario de menor a mayor.

8. Crea una clase NIF que represente el DNI con su correspondiente letra. Los atributos de la clase serán el número de DNI y su letra.

La clase NIF dispondrá de los siguientes métodos:

Un constructor por defecto.

Un constructor que reciba como parámetro el DNI y calcule y asigne la letra que le corresponde.

Un método leer(): que pida por teclado el número de DNI y calcule a partir del DNI introducido la letra que le corresponde.

Método toString() que muestre el NIF de la siguiente forma: ocho dígitos, un guión y la letra en mayúscula. Por ejemplo: 12345678-Z

Método para obtener la letra del NIF:

La letra del NIF se calculará usando un método privado. La forma de obtener la letra del NIF es la siguiente:

Se obtiene el resto de la división entera del número de DNI entre 23 y se usa la siguiente tabla para obtener la letra que corresponde:

0 - T	1 - R	2 - W	3 - A	4 - G	5 - M
6 - Y	7 - F	8 - P	9 - D	10 - X	11 - B
12 - N	13 - J	14 - Z	15 - S	16 - Q	17 - V
18 - H	19 - L	20 - C	21 - K	22 - E	

Una vez creada la clase, escribe un programa para probarla.

La ejecución de este programa produce esta salida por pantalla:
Introduce dni: 22334455
22334455-Y
12345678-Z