Programación Orientada a Objetos

Ejercicios: Clases y Objetos

1. Crea una clase `Complejo` que permita trabajar con números complejos (parte real y parte imaginaria). Incluye los siguientes métodos: constructores (por defecto y parametrizado), accedentes, mutadores, suma, resta, multiplicación, división, acumulación y `print()`.

2. Crea una clase `Racional` que permita trabajar con números racionales (fracciones). Incluye los siguientes métodos: constructores (por defecto y parametrizado), accedentes, `leer()`, suma, resta, multiplicación, división, comparaciones, `copia()` y `print()`.

3. Crea una clase `Rectangulo` que modele rectángulos por medio de cuatro puntos (los vértices). Dispondrá de dos constructores: uno que cree un rectángulo partiendo de sus cuatro vértices y otro que cree un rectángulo partiendo de la base y la altura, de forma que su vértice inferior izquierdo esté en (0,0). La clase también incluirá un método para calcular la superficie y otro que desplace el rectángulo en el plano.

4. Define una clase `Linea` con dos atributos: `_puntoA` y `_puntoB`. Son dos puntos por los que pasa la línea en un espacio de dos dimensiones. La clase dispondrá de los siguientes métodos:
   - ✓ `Linea()`
     Constructor predeterminado que crea una línea con sus dos puntos como (0,0) y (0,0).
   - ✓ `Linea(Punto, Punto)`
     Constructor que recibe como parámetros dos objetos de la clase `Punto`, que son utilizados para inicializar los atributos.
   - ✓ `mueveDerecha(double)`
     Desplaza la línea a la derecha la distancia que se indique.
   - ✓ `mueveIzquierda(double)`
     Desplaza la línea a la izquierda la distancia que se indique.
   - ✓ `mueveArriba(double)`
     Desplaza la línea hacia arriba la distancia que se indique.
   - ✓ `mueveAbajo(double)`
     Desplaza la línea hacia abajo la distancia que se indique.
   - ✓ Accedentes y mutadores.
   - ✓ Método que nos permita mostrar la información de la línea de la siguiente forma: `[`*`puntoA`*`,`*`puntoB`*`]`. Por ejemplo: `[(0.0,0.0),(1.0,1.0)]`.

5. Crea una clase `Cuenta` (bancaria) con atributos para el número de cuenta (un entero largo), el DNI del cliente (otro entero largo), el saldo actual y el interés anual que se aplica a la cuenta (porcentaje). Define en la clase los siguientes métodos:

- ✓ Constructor por defecto y constructor con DNI, saldo e interés
- ✓ Accedentes y mutadores. Para el número de cuenta no habrá mutador.
- ✓ `actualizarSaldo()`: actualizará el saldo de la cuenta aplicándole el interés diario (interés anual dividido entre 365 aplicado al saldo actual).
- ✓ `ingresar(double)`: permitirá ingresar una cantidad en la cuenta.
- ✓ `retirar(double)`: permitirá sacar una cantidad de la cuenta (si hay saldo).
- ✓ Método que nos permita mostrar todos los datos de la cuenta.

El número de cuenta se asignará de forma correlativa a partir de 100001, asignando el siguiente número al último asignado.

6. Desarrolla una clase `Cafetera` con atributos `_capacidadMaxima` (la cantidad máxima de café que puede contener la cafetera) y `_cantidadActual` (la cantidad actual de café que hay en la cafetera). Implementa, al menos, los siguientes métodos:
   - ✓ Constructor predeterminado: establece la capacidad máxima en 1000 (c.c.) y la actual en cero (cafetera vacía).
   - ✓ Constructor con la capacidad máxima de la cafetera; inicializa la cantidad actual de café igual a la capacidad máxima.
   - ✓ Constructor con la capacidad máxima y la cantidad actual. Si la cantidad actual es mayor que la capacidad máxima de la cafetera, la ajustará al máximo.
   - ✓ Accedentes y mutadores.
   - ✓ `llenarCafetera()`: pues eso, hace que la cantidad actual sea igual a la capacidad.
   - ✓ `servirTaza(int)`: simula la acción de servir una taza con la capacidad indicada. Si la cantidad actual de café "no alcanza" para llenar la taza, se sirve lo que quede.
   - ✓ `vaciarCafetera()`: pone la cantidad de café actual en cero.
   - ✓ `agregarCafe(int)`: añade a la cafetera la cantidad de café indicada.

7. Crea una clase NIF que se usará para mantener DNIs con su correspondiente letra. Los atributos serán el número de DNI (entero largo) y la letra que le corresponde. La clase dispondrá de los siguientes métodos:
   - ✓ Constructor predeterminado que inicialice el nº de DNI a 0 y la letra a espacio en blanco (será un NIF no válido).
   - ✓ Constructor que reciba el DNI y establezca la letra que le corresponde.
   - ✓ Accedentes y mutador para el número de DNI (que ajuste automáticamente la letra).
   - ✓ `leer()`: que pida el número de DNI (ajustando automáticamente la letra)
   - ✓ Método que nos permita mostrar el NIF (ocho dígitos, un guión y la letra en mayúscula; por ejemplo: `00395469-F`)

   La letra se calculará con un método auxiliar (privado) de la siguiente forma: 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
```

8. Crea una clase `Fecha` con atributos para el día, el mes y el año de la fecha.
   Incluye, al menos, los siguientes métodos:
   - ✓ Constructor predeterminado con el 1-1-1900 como fecha por defecto.
   - ✓ Constructor parametrizado con día, mes y año.
   - ✓ `leer()`: pedirá al usuario el día (1 a 31), el mes (1 a 12) y el año (1900 a 2050).
   - ✓ `bisiesto()`: indicará si el año de la fecha es bisiesto o no.
   - ✓ `diasMes(int)`: devolverá el número de días del mes que se le indique (para el año de la fecha).
   - ✓ `valida()`: comprobará si la fecha es correcta (entre el 1-1-1900 y el 31-12-2050); si el día no es correcto, lo pondrá a 1; si el mes no es correcto, lo pondrá a 1; y si el año no es correcto, lo pondrá a 1900. Será un método auxiliar (privado). Este método se llamará en el constructor parametrizado y en `leer()`.
   - ✓ Accedentes y mutadores.
   - ✓ `corta()`: mostrará la fecha en formato corto (`02-09-2003`).
   - ✓ `diasTranscurridos()`: devolverá el número de días transcurridos desde el 1-1-1900 hasta la fecha.
   - ✓ `diaSemana()`: devolverá el día de la semana de la fecha (0 para domingo, ..., 6 para sábado). El 1-1-1900 fue domingo.
   - ✓ `larga()`: mostrará la fecha en formato largo, empezando por el día de la semana (`martes 2 de septiembre de 2003`).
   - ✓ `fechaTras(long)`: hará que la fecha sea la correspondiente a haber transcurrido los días que se indiquen desde el 1-1-1900.
   - ✓ `diasEntre(Fecha)`: devolverá el número de días entre la fecha y la proporcionada.
   - ✓ `siguiente()`: pasará al día siguiente.
   - ✓ `anterior()`: pasará al día anterior.
   - ✓ `copia()`: devolverá un clon de la fecha.
   - ✓ `igualQue(Fecha)`: indica si la fecha es la misma que la proporcionada.
   - ✓ `menorQue(Fecha)`: indica si la fecha es anterior a la proporcionada.
   - ✓ `mayorQue(Fecha)`: indica si la fecha es posterior a la proporcionada.

9. Crea las siguientes clases (cada una en su archivo):
   - ✓ `Motor`: con métodos para arrancar el motor y apagarlo.
   - ✓ `Rueda`: con métodos para inflar la rueda y desinflarla.
   - ✓ `Ventana`: con métodos para abrirla y cerrarla.
   - ✓ `Puerta`: con una ventana y métodos para abrir la puerta y cerrar la puerta.

- ✓ `Coche`: con un motor, cuatro ruedas y dos puertas;
  con los métodos que te parezcan adecuados

10. Crea una clase `Hora` con atributos para las horas, los minutos y los segundos de la hora. Incluye, al menos, los siguientes métodos:
    - ✓ Constructor predeterminado con el 00:00:00 como hora por defecto.
    - ✓ Constructor parametrizado con horas, minutos y segundos.
    - ✓ `leer()`: pedirá al usuario las horas, los minutos y los segundos.
    - ✓ `valida()`: comprobará si la hora es correcta; si no lo es la ajustará. Será un método auxiliar (privado) que se llamará en el constructor parametrizado y en `leer()`.
    - ✓ Accedentes y mutadores.
    - ✓ `print()`: mostrará la hora (`07:03:21`).
    - ✓ `aSegundos()`: devolverá el número de segundos transcurridos desde la medianoche.
    - ✓ `deSegundos(int)`: hará que la hora sea la correspondiente a haber transcurrido desde la medianoche los segundos que se indiquen.
    - ✓ `segundosDesde(Hora)`: devolverá el número de segundos entre la hora y la proporcionada.
    - ✓ `siguiente()`: pasará al segundo siguiente.
    - ✓ `anterior()`: pasará al segundo anterior.
    - ✓ `copia()`: devolverá un clon de la hora.
    - ✓ `igualQue(Hora)`: indica si la hora es la misma que la proporcionada.
    - ✓ `menorQue(Hora)`: indica si la hora es anterior a la proporcionada.
    - ✓ `mayorQue(Hora)`: indica si la hora es posterior a la proporcionada.

11. Crear una clase `Empleado` que modele la información que una empresa mantiene sobre cada empleado: NIF, sueldo base, pago por hora extra, horas extra realizadas en el mes, tipo (porcentaje) de IRPF, casado o no y número de hijos.

    La clase debe contemplar accedentes y mutadores para todos los atributos. Al crear los objetos se podrá proporcionar, si se quiere, el número de DNI. Los demás servicios que deberán proporcionar los objetos de la clase serán los siguientes:
    - ✓ Cálculo y devolución del complemento correspondiente a las horas extra realizadas.
    - ✓ Cálculo y devolución del sueldo bruto.
    - ✓ Cálculo y devolución de las retenciones (IRPF) a partir del tipo, teniendo en cuenta que el porcentaje de retención que hay que aplicar es el tipo menos 2 puntos si el empleado está casado y menos 1 punto por cada hijo que tenga; el porcentaje se aplica sobre todo el sueldo bruto.
    - ✓ `println()`: visualización de la información básica del empleado.
    - ✓ `printAll()`: visualización de toda la información del empleado. La básica más el sueldo base, el complemento por horas extra, el sueldo bruto, la retención de IRPF y el sueldo neto.

- ✓ `copia()`: clonación de objetos.

12. Desarrolla una clase `Cancion` con los siguientes atributos:
    - ✓ `titulo`: una variable `String` que guarda el título de la canción.
    - ✓ `autor`: una variable `String` que guarda el autor de la canción.

    y los siguientes métodos:
    - ✓ `Cancion(String, String)`: constructor que recibe como parámetros el título y el autor de la canción (por este orden).
    - ✓ `Cancion()`: constructor predeterminado que inicializa el título y el autor a cadenas vacías.
    - ✓ `dameTitulo()`: devuelve el título de la canción.
    - ✓ `dameAutor()`: devuelve el autor de la canción.
    - ✓ `ponTitulo(String)`: establece el título de la canción.
    - ✓ `ponAutor(String)`: establece el autor de la canción.

13. Crea una clase `Libro` que modele la información que se mantiene en una biblioteca sobre cada libro: título, autor (usa la clase `Persona`), ISBN, páginas, edición, editorial , lugar (ciudad y país) y fecha de edición (usa la clase `Fecha`). La clase debe proporcionar los siguientes servicios: accedentes y mutadores, método para leer la información y método para mostrar la información. Este último método mostrará la información del libro con este formato:

```
Título: Introduction to Java Programming
3a. edición
Autor: Liang, Y. Daniel
ISBN: 0-13-031997-X
Prentice-Hall, New Jersey (USA), viernes 16 de noviembre de
2001
784 páginas
```

14. Escribe un programa que pida diez números enteros y los muestre en orden inverso (del último leído hasta el primero).

15. Escribe un programa que rellene un array de 30 `double`s con números aleatorios y luego calcule la media y la desviación estándar.

$$media = \frac{\sum_{i=1}^{n} x_i}{n} \qquad desviación = \sqrt{\frac{\sum_{i=1}^{n} (x_i - media)^2}{n-1}}$$

    La función `Math.random()` devuelve un número real aleatorio entre 0 y 1.

16. Desarrolla una clase `Array234` que maneje un array de dimensiones 2 x 3 x 4. La clase tendrá estas características:

- ✓ El array de 2 x 3 x 4 será un atributo de la clase.
- ✓ El constructor de la clase inicializará el array aleatoriamente, utilizando la función `Math.random()`.
- ✓ Un método `max_min()` mostrará en la pantalla los valores máximos y mínimos del array, así como los índices de los componentes que los almacenan.

17. Desarrolla una clase `CD` con los siguientes atributos:
    - ✓ `canciones`: un array de objetos de la clase `Cancion`.
    - ✓ `contador`: la siguiente posición libre del array `canciones`.

    y los siguientes métodos:
    - ✓ `CD()`: constructor predeterminado (creará el array `canciones`).
    - ✓ `numeroCanciones()`: devuelve el valor del `contador` de canciones.
    - ✓ `dameCancion(int)`: devuelve la `Cancion` que se encuentra en la posición indicada.
    - ✓ `grabaCancion(int, Cancion)`: cambia la Cancion de la posición indicada por la nueva `Cancion` proporcionada.
    - ✓ `agrega(Cancion)`: agrega al final del array la `Cancion` proporcionada.
    - ✓ `elimina(int)`: elimina la `Cancion` que se encuentra en la posición indicada.

18. Desarrollar una lista de `Libro`s ordenada por título. La funcionalidad de la lista será la habitual: conocer el número de libros que hay en la lista, insertar un nuevo libro (en la posición que le corresponda), eliminar el libro de una determinada posición y obtener el libro de una determinada posición. También incluirá un método para buscar un libro a partir de una parte de su título (sin distinguir entre mayúsculas y minúsculas); el método devolverá la posición en la que se encuentra el libro (–1 si no se encuentra).