Relación de ejercicios – Clase

1. Diseñar la clase Hora, que representa un instante de tiempo compuesto por la hora (de 0 a 23) y los minutos. Dispone de los métodos:
    - Hora(hora, minuto), que construye un objeto con los datos pasados como parámetros.
    - public void inc(), que incrementa la hora en un minuto.
    - boolean setMinutos(valor), que asigna un valor, si es válido, a los minutos. Devuelve true o false según haya sido posible modificar los minutos o no.
    - boolean setHora(valor), que asigna un valor, si está comprendido entre 0 y 23, a la hora. Devuelve true o false según haya sido posible cambiar la hora o no.
    - String toString(), que devuelve un String con la representación de la hora.
2. A partir de la clase Hora implementar la clase HoraExacta, que incluye en la hora los segundos. Además de los métodos heredados de Hora, dispondrá de:
    - HoraExacta(hora, minuto, segundo), que construye un objeto con los
    - datos pasados como parámetros.
    - setSegundo(valor), que asigna, si está comprendido entre 0 y 59, el valor indicado a los segundos.
    - inc(), que incrementa la hora en un segundo.
3. Añadir a la clase HoraExacta un método que compare si dos horas (la invocante y otra pasada como parámetro de entrada al método) son iguales o distintas.
4. Crear la clase abstracta Instrumento, que almacena en una tabla las notas musicales de una melodía (dentro de una misma octava). El método add() añade nuevas notas musicales. La clase también dispone del método abstracto interpretar() que, en cada subclase que herede de Instrumento, mostrará por consola las notas musicales según las interprete. Utilizar enumerados para definir las notas musicales.
5. Crear la clase Piano heredando de la clase abstracta Instrumento. Incluir atributos de cantidad de teclas y cantidad de pedales.
6. Las empresas de transporte, para evitar daños en los paquetes, embalan todas sus mercancías en cajas con el tamaño adecuado. Una caja se crea expresamente con un ancho, un alto y un fondo y, una vez creada, se mantiene inmutable. Cada caja lleva pegada una etiqueta, de un máximo de 30 caracteres, con información útil como el nombre del destinatario, dirección, etc. Implementa la clase caja con los siguientes métodos:

- Caja (int ancho, int alto, int fondo, Unidad unidad): que construye una caja con las dimensiones especificadas, que pueden encontrarse en «cm» (centímetros) o «m» (metros).
- double getVolumen(): que devuelve el volumen de la caja en metros cúbicos.
- void setEtiqueta(String etiqueta): que modifica el valor de la etiqueta de la caja.
- string toString() : que devuelve una cadena con la representación de la caja.

7. La empresa de mensajería BiciExpress, que reparte en bicicleta, para disminuir el peso que transportan sus empleados solo utiliza cajas de cartón. El volumen de estas se calcula como el 80% del volumen real, con el fin de evitar que se deformen y se rompan. Otra característica de las cajas de cartón es que sus medidas siempre están en centímetros. Por último, la empresa, para controlar costes, necesita saber cuál es la superficie total de cartón utilizado para construir todas las cajas.

   Escribe la clase CajaCarton heredando de la clase caja.

8. Implementa la clase abstracta Poligono, con los atributos base y altura, de tipo double y el método abstracto double area(). Heredando de Poligono, implementa las clases no abstractas Triangulo y Rectangulo.

9. Define la clase Punto,que tiene como atributos las coordenadas x e y, de tipo entero, que lo sitúan en el plano. Además del constructor, implementa el método double distancia(Punto otroPunto), que devuelve la distancia a otro punto que se le pasa como parámetro. A partir de Punto, por herencia, implementa la clase Punto3D, que representa un punto en tres dimensiones y necesita una coordenada adicional z. Reimplementa el método distancia() para puntos 3D. Implementa el método equals() para las clases Punto y Punto3D, teniendo en cuenta que dos puntos son iguales solo si tienen todas sus coordenadas iguales.

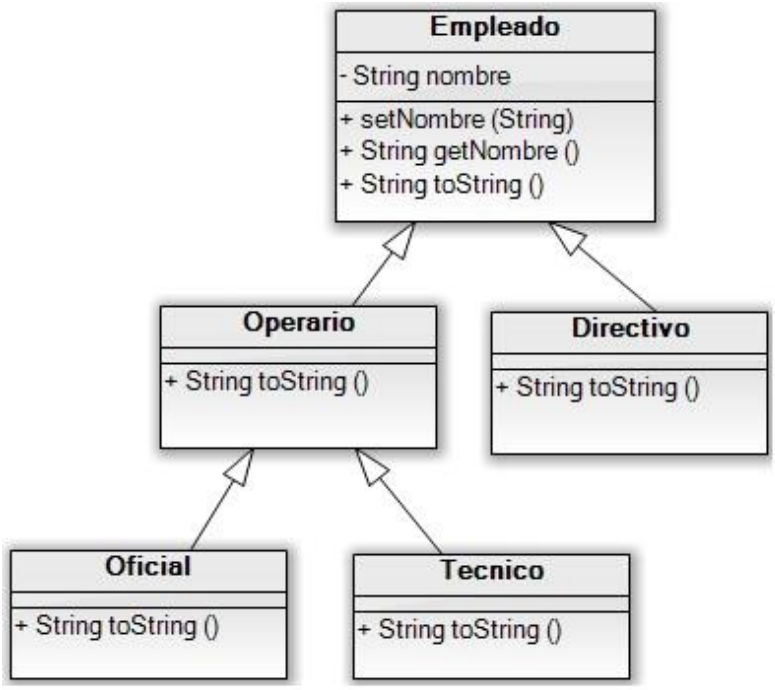
10. En primer lugar, diseña la clase Yogur sabiendo que un yogur siempre tiene 120.5 calorías. Se requiere implementar un método consultor para obtener sus calorías. A continuación, diseña la clase YogurDesnatado sabiendo que siempre tiene la mitad de calorías que un Yogur normal. Finalmente, construye una clase Main con un método principal que cree un objeto Yogur y un YogurDesnatado y muestre sus calorías.

11. Escribe una clase Multimedia para almacenar información de los objetos de tipo multimedia (películas, discos, mp3,mp4…). Esta clase contiene título, autor, formato, y duración como atributos. El formato puede ser uno de los siguientes: wav, mp3, midi, avi, mov, mpg, cdAudio y dvd. El valor de todos los atributos se pasa por parámetro en el momento de crear el objeto. Esta clase tiene además, un método para devolver cada

uno de los atributos y un método toString() que devuelve la información del objeto. Por último, un método equals() que recibe un objeto de tipo Multimedia y devuelve true en caso de que el título y el autor sean iguales y false en caso contrario.

12. Escribe una clase Película que herede de la clase Multimedia anterior. La clase Película tiene, además de los atributos heredados, un actor principal y una actriz principal. Se permite que uno de los dos sea nulo, pero no los dos. La clase debe tener dos métodos para obtener los nuevos atributos y debe sobreescribir el método toString() para reflejar la nueva información.

13. Codifica la siguiente jerarquía de clases java representada por este diagrama UML:



La clase base es la clase Empleado. Esta clase contiene:

- Un atributo privado nombre de tipo String que heredan el resto de clases.
- Un constructor por defecto.
- Un constructor con parámetros que inicializa el nombre con el String que recibe.
- Método set y get para el atributo nombre.
- Un método toString() que devuelve el String: "Empleado " + nombre.

El resto de clases solo deben sobrescribir el método toString() en cada una de ellas y declarar el constructor adecuado de forma que cuando la ejecución de las siguientes instrucciones:

```
Empleado E1 = new Empleado("Rafa");
Directivo D1 = new Directivo("Mario");
Operario OP1 = new Operario("Alfonso");
Oficial OF1 = new Oficial("Luis");
Tecnico T1 = new Tecnico("Pablo");
System.out.println(E1);
System.out.println(D1);
System.out.println(OP1);
System.out.println(OF1);
System.out.println(T1);
```

Den como resultado:

```
Empleado Rafa
Empleado Mario -> Directivo
Empleado Alfonso -> Operario
Empleado Luis -> Operario -> Oficial
Empleado Pablo -> Operario -> Tecnico
```