1 **Personas:**

Construye en Java la clase `Persona` que se define a continuación:

```
public class Persona
{
 private String nombre;
 private int edad;
 private float altura;
 private String ocupacion;
 String getNombre(){
      return nombre;
 }
 void setNombre(String nombre){
      this.nombre=nombre;
 }
}
```

1.1 Realizar un main que solicite un valor al usuario y lo introduzca en el atributo `nombre`, para posteriormente mostrar por pantalla el nuevo valor del atributo.

1.2 Añade a la clase `Persona` los métodos que faltan para poder consultar y modificar el valor de todos los atributos. Después completa el programa para comprobar el funcionamiento de los nuevos métodos.

1.3 Crea un método constructor para la clase `Persona` que asigne los siguientes valores a sus atributos:
nombre="Sin nombre"
edad=0
altura=0.0f;
ocupacion="Sin ocupación"
Crea un main que declare un objeto de tipo `Persona` utilizando el constructor, para posteriormente mostrar el contenido de sus atributos por pantalla.

1.4 Crea un constructor con parámetros para la clase `Persona` que inicialice los atributos del objeto con los valores indicados en los parámetros. Utiliza el operador *`this.atributo`*. A continuación crea un main que declare un objeto de tipo `Persona` utilizando el constructor, para posteriormente mostrar el contenido de los atributos por pantalla.

1.5 Modifica el constructor del apartado 1.3 para que utilice el this(parámetros).

1.6 Añadir un atributo decimal a la clase Persona que almacene el sueldo. Crearle a dicho atributo los métodos get y set. Crear un método sumar_sueldo que reciba una persona por parámetro. Dicho método tiene que incrementar el sueldo con el sueldo de la persona recibida como parámetro. Crear otro método, doblar_sueldo, que doble el sueldo utilizando el método sumar_sueldo y this.

2 **Visibilidad:**

Dados los modificadores de acceso private, protected, public y friendly, probar la visibilidad en los atributos y métodos en los siguientes casos:

2.1 Misma clase

2.2 Clase en el mismo paquete

2.3 Clase en otro paquete

3 **Alumnos:**
Realizar una clase de nombre Alumno que cumpla estas especificaciones:
el constructor admite como argumentos el número de matrícula del alumno y el nombre y almacena éstos en los correspondientes atributos.
Contiene los siguientes métodos:

- *ponNota*: con dos argumentos de tipo double que corresponden a dos notas de un examen. El método almacena éstos en dos atributos de tipo double.
- *dameMedia*: retorna la media de las notas.
- *toString*: retorna una descripción del alumno con el número de matrícula, el nombre y la nota media.

Realiza una aplicación de nombre VerAlumno que solicite los datos de un alumno y sus notas , construya un objeto de la clase Alumno y muestre los datos de éste.

4 **Películas en DVD:**
Escribe una clase que represente una película en DVD de nombre DVDCine con los atributos necesarios para mostrar su ficha. Esta clase contará con un constructor que admite como argumentos todos los atributos de la clase.
Escribe los siguientes métodos para la clase DVDCine:

- toString: este método retorna una descripción completa de la película, por ejemplo:
  UN FINAL MADE IN HOLLYWOOD (HOLLYWOOD ENDING)
  De: Woody Allen
  Con: Woody Allen y George Hamilton
  Comedia – 114 min
  Resumen: Los Oscars ganados en el pasado por el ex-genio del cine Val Waxman...
  Este método debe usar el método muestraDuracion. La película tiene dos nombres, el original y el traducido.
- esThriller: este método retorna cierto si la película pertenece a este género cinematográfico.
- tieneResumen: retorna cierto si la ficha de la película tiene el resumen escrito.
- muestraDuracion: retorna la duración con el siguiente formato: 114 min

Escribe una aplicación que solicite los datos de una película, genere un objeto DVD y muestre éste con el formato del método toString. Prueba también los métodos esThriller y tieneResumen.

5 **Alimentos:**
Realizar una clase de nombre *Alimento* cuyos objetos representen alimentos. Éstos tendrán los atributos siguientes:

- Nombre.
- Contenido en lípidos expresado en tanto por ciento.
- Contenido en hidratos de carbono expresado en tanto por ciento.
- Contenido en proteínas expresado en tanto por ciento.
- Si es o no de origen animal.
- Contenido en vitaminas expresado en los códigos A alto, M medio y B bajo.
- Contenido en minerales expresado en los códigos A alto, M medio y B bajo.

La clase tiene dos constructores: uno que admite como argumentos el nombre del alimento, y otro que admite todos los atributos.
La clase contiene los siguientes métodos:

- *esDietetico*. Este método retorna cierto si el alimento contiene menos del 20% de lípidos y el contenido en vitaminas no es bajo.
- *toString.* Retorna una descripción del alimento.
- *calculaContenidoEnergetico.* Este método retorna el contenido en Kcal de un gramo de alimento, considerando que un gramo de lípidos contiene 9.4 Kcal, un gramo de proteínas contiene 5.3 y un gramo de hidratos de carbono contiene 4.1 Kcal.
- *esRecomendableParaDeportistas.* Este método retorna cierto si el alimento cumple la siguiente lista: proteínas: 10-15%, lípidos:30-35 %, hidratos de carbono: 55-65%.

Hacer una aplicación en la que se creen dos alimentos usando los dos constructores. Mostrar los datos de los alimentos, sus contenidos energéticos, si son dietéticos y recomendables para deportistas.

6 **Vehículos:**
Realiza una clase de nombre Vehículo que contenga como atributos el modelo de tipo String, la potencia de tipo double y la tracción a las cuatro ruedas(cRuedas) de tipo boolean.
El constructor de la clase admitirá como argumento el modelo. La clase tendrá como métodos de tipo get y set para la potencia y para la tracción a las cuatro ruedas. La clase contará con el método toString el cual retorna los datos de cada vehículo y si tiene tracción a las cuatro ruedas. Realiza una aplicación que solicite al usuario los datos de varios vehículos hasta que el usuario escriba como modelo la cadena “fin” en mayúscula o en minúscula, en cuyo caso no se generará el objeto de la clase Vehículo. Una vez introducidos todos los vehículos, la aplicación terminará mostrando los datos de todos los vehículos y emitiendo un mensaje de despedida.

7 **Dimensiones:**
Construye una clase Dimensiones con tres atributos de tipo real, en concreto float, que representen las tres dimensiones (el alto, ancho y fondo) de un paquete:

x (sería el ancho).
y (sería el alto).
z (sería el fondo).

Crea los siguientes métodos para la clase. A continuación, crea un programa para probar todos sus métodos.

- Dimensiones(): Constructor que inicializa los atributos a cero.
- Dimensiones(float x, float y, float z): Constructor que inicializa los atributos a los valores indicados por los parámetros.
- Métodos set y get para cada dimensión.
- float alturaMaximaParaApilar(float alturaMaxima): Devuelve la altura máxima que puede tener otro paquete para que se pueda apilar con el que invoca al método sin sobrepasar entre ambos la altura que se indica como parámetro.
- float sumaDimensionesX(Dimensiones dimensiones): Devuelve la suma de las anchuras de dos paquetes.
- float sumaDimensionesY(Dimensiones dimensiones): Devuelve la suma de las alturas de dos paquetes.
- public float sumaDimensionesZ(Dimensiones dimensiones): Devuelve la suma de los fondos de dos paquetes.
- String toString(). Devuelve la cadena formada por cada uno de los atributos y su valor, por ejemplo, para una dimensión concreta se obtendría algo como lo que se

muestra: "Dimensiones: x = 3.3 ; y = 5.5 ; z = 7.7".
- boolean esPosibleApilarPaquetes(Dimensiones dimensionesOtroPaquete, float alturaMaximaParaApilar): Indica si dadas las dimensiones de nuestro paquete, sería posible o no apilarlo con otro paquete cuyas dimensiones se indican como parámetro, sabiendo que la altura máxima permitida para la pila de los dos paquetes no puede exceder la indicada como segundo argumento. Devuelve verdadero si entre ambos no exceden esa altura y falso si la superan.
- void voltearDimensionX():Gira el paquete una vez alrededor del eje x
- void voltearDimensionY():Gira el paquete una vez alrededor del eje y
- void voltearDimensionZ():Gira el paquete una vez alrededor del eje z
- public void ApilarPaquetes(Dimensiones dimensionesOtroPaquete, float alturaMaximaParaApilar): Indica todas las posibles soluciones de apilación entre nuestro paquete y el indicado por parámetro. Para ello, utiliza métodos anteriores.

8 **Productos:**
Una tienda de informática nos ha contratado para hacerle una aplicación en java. De los productos que tiene, quiere almacenar el modelo, el stock, el procesador y el precio, con las siguientes características:
- El modelo tiene el siguiente formato: tres dígitos, guión y cuatro letras mayúsculas(incluida la ñ). Ejemplos: 112-ACER,334-HHPP,435-ASUS
- El procesador solamente puede ser de dos tipos: Intel o Amd. Los intel traen una memoria de 8Gb y los Amd de 4Gb (utiliza enum compuesto).
- El precio en formato decimal y el stock. Ambos deben ser positivos.

Crea la clase **Producto** haciendo lo siguiente:
- Los atributos no serán visibles desde fuera de la clase.
- Crea un método para consultar el modelo que solamente sea visible para las clases del mismo paquete.
- Crea un método para consultar el stock visible para todas las clases.
- Crea métodos para actualizar los atributos que sean visibles también para las clases de otro paquete. Se encargarán de hacer las comprobaciones necesarias para garantizar que el formato o los rangos de valores son correctos y lanzarán la excepción *Exception* cuando no sean válidos.
- Crea un constructor con parámetros que inicialice los atributos del objeto con los valores indicados en los parámetros. El constructor no construirá el objeto si algún atributo no es válido. Para ello, utilizará los métodos anteriores para garantizar que los valores de los atributos son correctos, pero no tratará la excepción, sino que obligará al código que lo haya invocado a tratar la excepción.
- Crea un constructor que inicialice los atributos del objeto con los siguientes valores. Utiliza el *this(parámetros).*
  - Modelo: 000-NNNN
  - Stock: 0
  - Procesador Intel
  - Precio: 0
- Crea el método toString(). Muestra el precio con 2 decimales.
- Crea el método disminuirStock() que decremente en 1 el stock del producto.

9 **Sorteo:**

El dueño de la tienda de informática está preocupado con la crisis, entonces se le ha ocurrido hacer un sorteo para captar clientes. Para ello, utilizando la clase Producto del ejercicio anterior, realizar una clase Sorteo que muestre el siguiente menú:

1. Organizar sorteo
2. Mostrar sorteo
3. Sorteo
4. Salir

En la opción 1. Organizar sorteo, el dueño introducirá datos de cuatro productos a sortear. Dichos productos, se introducirán aleatoriamente en una matriz 4x4.

En la opción 2. Mostrar sorteo, se mostrarán las casillas con premio y el producto que contienen. Las casillas se indicarán con un número: 1, 2, ….16

En la opción 3. Sorteo, un cliente probará suerte eligiendo una casilla del 1 al 16. Si acierta, se mostrarán todos los datos del producto. Si no le ha tocado nada, se mostrará el mensaje "No ha tenido suerte". En caso de que acierte, se le regalará el producto al cliente por lo que el stock habrá que disminuirlo. Si llega a cero, se quita el producto del sorteo, si no, se cambia de casilla.

En la opción 4. Salir, se termina el programa sorteo.