1 **Personas:**

Realiza los siguientes apartados creando un paquete para cada uno. Introduce en los paquetes dos clases: Persona.java y App.java. En cada apartado, copia ambas clases del apartado anterior y después realiza lo correspondiente a dicho apartado.

Pesona.java contiene lo siguiente:

```
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;
	}
}
```

App.java contendrá el main que cree una persona y haga lo que pida el enunciado del correspondiente apartado.

1.1 Crea una persona. Muestra su nombre. Solicita al usuario un nuevo nombre y asígnaselo a la persona. Muestra por consola el nuevo nombre.

1.2 Añade a la clase `Persona` los métodos que faltan para poder consultar y modificar el valor de todos los atributos. Crea una persona. Muestra sus datos. Solicítale al usuario nuevos datos de la persona y asígnaselos. Muestra por consola los nuevos datos de la persona.

1.3 Crea un 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 una persona utilizando el constructor. Muestra por consola los datos de la persona.

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`*. Crea una persona utilizando el constructor. Muestra por consola los datos de la persona.

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. Créale los métodos para poder consultar y modificar el nuevo atributo. Crea un método *sumarSueldo* que reciba una persona por parámetro. Dicho método tiene que incrementar el sueldo con el sueldo de la persona recibida como parámetro. Crea otro método llamado doblarSueldo que doble el sueldo utilizando el método *sumarSueldo* y this. Crea un programa para probar ambos métodos.

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 todos sus datos, incluida 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. La película tiene dos nombres, el original y el traducido. Realiza los siguientes métodos para la clase DVDCine:
- esThriller: este método devuelve cierto si la película pertenece a este género cinematográfico (tratar los géneros de las películas con enum)
- tieneResumen: devuelve cierto si la ficha de la película tiene el resumen escrito.
- muestraDuracion: devuelve una cadena con la duración en minutos. Ej: 114 min.
- toString: este método devuelve una descripción completa de la película de la siguiente manera:
  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...

Si la película no tiene resumen, la última línea no aparece.
Escribe una aplicación que solicite los datos de una película, genere un objeto DVD y pruebe todos los métodos.

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 mediante un enum compuesto ALTO, MEDIO y BAJO. Dicho enum tendrá un atributo código con las iniciales: A, M y B respectivamente.
- Contenido en minerales expresado con el mismo enum que las vitaminas.

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 **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 *IllegalArgumentException* 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 sean correctos.
- 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.