EJERCICIO 1

Crea una clase llamada Smartphone con los siguientes atributos privados:

- marca (Texto)
- modelo (Texto)
- imei (Texto)
- precio (Número decimal)
- fechaFabricacion (LocalDate)
- activo (Booleano)

Añade 3 constructores:

- Constructor por defecto
- Constructor que reciba e inicialice el imei
- Constructor que reciba e inicialice el imei, la marca y el modelo

Todos los constructores deben inicializar el smartphone como activo y a precio 0.

Añade los siguientes métodos:

- setPrecio() → permite cambiar el precio del Smartphone
- activar → activa el Smartphone
- desactivar → inactiva el Smartphone
- getEdad() → devuelve los años transcurridos desde el 1 de abril del 2000 hasta la fecha de fabricación del Smartphone
- cambiarMarcaModelo() → recibe una marca y modelo para cambiar los del Smartphone
- establecerFabricacion() → recibe y cambia la fecha de fabricación del Smartphone
- getPrecioMasIva() → devuelve el precio del Smartphone sumándole el IVA (21%)
- isAltaGama() → recibe un precio base y devolverá un booleano indicando si el Smartphone es de alta gama o no. Los Smartphone de alta gama son aquellos cuyo precio es mayor al precio base recibido
- isDatosCompletos() → devuelve un booleano indicando si los datos del Smartphone están completos. Para que lo estén, al menos el imai debe no estar vacío ni contener sólo espacios en blanco, y el precio no puede ser null.
- isGratis() → devuelve si el precio del Smartphone es o no es 0 euros.
- rebajar() → modifica el precio del Smartphone restándole 10 euros.
- toString() → sobrescribe el método toString de Object. Debe devolver una cadena con el formato “marca (modelo)”. Si el Smartphone además está inactivo, la cadena será así “marca (modelo) – INACTIVO”.

EJERCICIO 2

Tenemos dos tipos de teléfonos:

- Fijo
- Móvil

Cada teléfono cuenta con:

- Número → Entero que es el número de teléfono
- EnLlamada → Booleano indicando si el teléfono está en una llamada

Además, los fijos tienen una dirección y los móviles una posición GPS

Crea las clases necesarias para poder trabajar con estos dispositivos. Sólo tendrán un constructor que reciba el número de teléfono y dirección o GPS según se trate de un Fijo o un Móvil. El atributo "enLlamada" se inicializa a false en los constructores.

Además, todas las clases tendrán estos métodos:

1. consultarNumero() → Devuelve el número de teléfono
2. marcar() → Recibe un número de teléfono.
    a. Debe imprimir "LLAMANDO A … " y el número recibido
    b. Si el número recibido es igual al del propio teléfono, imprimirá "COMUNICANDO" y no hará nada más.
    c. En caso contrario, imprimirá "EN COMUNICACIÓN", y cambiará su atributo enLlamada a true
3. colgar() → No recibe ni devuelve nada.
    a. Si estaba enLlamada → debe imprimir "COMUNICACIÓN TERMINADA" y cambiar el atributo enLlamada a false
    b. Si no estaba enLlamada, no hará nada

Crea un programa que haga lo siguiente:

- Crea un teléfono móvil y otro fijo con los números 654654654 y 954954954 respectivamente, y pasa una dirección y una posición GPS según corresponda.
- Usa el método consultarNumero() de cada uno para imprimirlo.
- Luego llama a marcar() del móvil pasando como número de destino el 654654654
- Luego llama a marcar() de nuevo del móvil pasando como número el 610610610
- Luego llama a colgar() del móvil
- Vuelve a llamar a colgar() del móvil

EJERCICIO 3

¿Cómo has creado el atributo GPS en el ejercicio anterior? Si lo has hecho como un Integer o un String, mejóralo. Crea una clase que sea LocalizacionGPS que tenga dos atributos que sean las coordenadas GPS, con su get y set y su constructor.
Haz que el atributo de la clase Móvil sea un objeto de esta clase.

EJERCICIO 4

Una librería se encarga de vender libros de 2 tipos: libros físicos normales y libros digitales. Todos los libros tienes título, autor, fecha edición y número de páginas. Los libros físicos también tienen un tamaño (ancho y alto en cm) y un peso en gramos. Números enteros.

Crea los get() y set() que consideres.

La librería tendrá por tanto una lista de libros y además una dirección y un número de teléfono.
Crea en la librería los siguientes métodos:

- obtenerLibroMasAntiguo() que devuelve el libro que sea más antiguo de todos los registrados
- obtenerPesoTotalDeLibros() que devuelve la suma del peso de todos los libros (los libros digitales tienen peso 0)
- obtenerNumPaginasMedio() que devuelve el número de páginas que tiene un libro de media.

Crea una aplicación que cree:

- Una Librería en la dirección “Avda. de los sueños perdidos, 89” y con el tlfno 954696954.
- Dos libros físicos con los datos que quieras
- Dos libros digitales con los datos que quieras
- Añade los 4 libros a la librería
- Llama a los métodos creados en la librería e imprime los resultados para comprobar que funcionan correctamente.

EJERCICIO 5

En una biblioteca existen una serie de publicaciones. Tenemos tres tipos de publicaciones: Libros, Revistas y CD. Cada publicación tiene:

- Código
- Autor
- Título
- Año de publicación

Además, los libros y CDs tienen:

- Atributo booleano que indica si están prestados o no

Y los libros y revistas tienen:

- Atributo booleano que indica si están siendo consultado en este momento o no

Se solicita:

1. Crear las clases necesarias para poder trabajar con la biblioteca.
2. Crear métodos get() y set() para todos los atributos comunes a las publicaciones (código, autor, título, año publicación)
3. Crear Constructor que reciba todos los atributos comunes a las publicaciones (código, autor, título, año publicación). Los atributos booleanos de cada clase se inicializarán a FALSO por defecto en los constructores.
4. Crear método toString() que devuelva una representación en cadena del valor de todos los atributos.

5. Crear método equals() que devuelva true cuando dos publicaciones tengan el mismo código.
6. Crea además una interfaz IPrestable que tenga estos métodos:
    a. prestar() → cambia el atributo prestado a true
    b. devolver() → cambia el atributo prestado a false
    c. estaPrestado() → devuelve true/false indicando si está o no prestado
7. Haz que las clases Libro y CD implementen la interfaz IPrestable
8. Crea otra interfaz IConsultable que tenga estos métodos:
    a. retirarParaConsulta() → cambia el atributo consultado a true
    b. terminarConsulta() → cambia el atributo consultado a false
    c. estaConsultando() → devuelve true/false indicando si está o no siendo consultado
9. Haz que las clases Libro y Revista implementen la interfaz IConsultable
10. Modifica los métodos que consideres para garantizar que una publicación que está siendo consultada no se puede prestar, ni viceversa.

Crea una aplicación que haga lo siguiente:

1. Crea una publicación de cada tipo y añádela a la biblioteca
2. Presta la publicación libro y la publicación CD
3. Intenta retirar para consulta la publicación libro y la publicación revista
4. Imprime todas las publicaciones
5. Devuelve el libro y luego retíralo para consulta
6. Intenta pedir prestado el libro
7. Imprime todas las publicaciones.
8. Devuelve el CD y termina la consulta del libro y la revista
9. Imprime el estado de prestado y de consultando de todas las publicaciones

EJERCICIO 6 (avanzado)

¿Recuerdas el ahorcado del tema anterior? Vuelve a hacerlo, pero utilizando un diseño de clases para que sea todo orientado a objetos. Te aconsejo tener estas clases:

- SelectorPalabras
- Tablero
- Partida
- UserInterface (se encarga de comunicar con el usuario mensajes)

Para probarlo, crea una clase App que únicamente debería crear una Partida (sin pasar ningún parámetro) y llamar a un método start() que comience el juego.