UD6 POO – Ejercicios III: Arrays de Objetos

Para hacer estos ejercicios hay que programar vectores de objetos utilizando las clases realizadas en el boletín Ejercicios II. Se recominda utilizar las soluciones propuestas.

Ejercicio 1. Array de Asignaturas

Escribe un programa con un vector con 5 asignaturas (puedes poner sus datos all llamar al constructor, no es necesario pedirlo al usuario). Recuerda primero crear el vector y luego instanciar 5 objetos de la clase Asignatura en las 5 posiciones del vector. Recorre el vector con un bucle for e imprime la información de todas las asignaturas con el método imprime().

Ejercicio 2. Array de Coches

Escribe un programa que haga lo siguiente:

1. Pedir por teclado los datos de 3 coches y guardarlos en un vector de coches.
2. Imprimir por pantalla lo siguiente:
    1. La información de todos los coches. Añade a la clase Coche el método imprimir().
    2. Cuantos coches tienen la pintura metalizada.
    3. Cuantos coches fueron fabricados antes del año 2000.
    4. Cuantos coches tienen cada modalidad de seguro (a terceros y a todo riesgo).

Ejercicio 3. Array de CuentaCorriente

Escirbe un programa con un vector de 5 cuentas corrientes (instancia los 5 objetos con valores de ejemplo, esto no hay que pedirlo al usuario). Luego, mostrará al usuario un pequeño menú con las opciones indicadas abajo. El menú se repetirá una y otra vez hasta que el usuario decida salir del programa.

1. **Ver cuentas:** Imprimirá por pantalla la información de las 5 cuentas.
2. **Ingresar:** Permitirá al usuario ingresar dinero en una cuenta. Elegirá una cuenta e indicará la cantidad a ingresar. Imprimirá la información de la cuenta tras el ingreso.
3. **Retirar:** Como ‘ingresar’ pero para retirar dinero de una cuenta.
4. **Transferencia:** Permitirá mover dinero de una cuenta a otra. Pedirá al usuario elegir las cuentas de origen y de destino, realizará la transferencia y luego imprimirá la información de ambas cuentas.
5. **Salir:** Termina el programa.

Ejercicio 4. Array de Relojes

Haz un programa que cree un array de 5 relojes y luego:

1. Imprime por pantalla sus cinco horas.
2. Haz 'tick' en todos los relojes para que pase un segundo en su hora. Imprime sus horas para comprobarlo.
3. Haz 'tick' 60 veces en todos los relojes para simular que ha pasado un minuto (necesitarás usar un bucle doble). Vuelve a imprimir sus horas para comprobarlo.
4. Haz 'tick' en todos los relojes tantas veces como sea necesario para que pase una hora (tamién bucle doble). Vuelve a imprimir sus horas para comprobarlo.

EXTRA: Añade a la clase Reloj un segundo método **tick** con un parámetros **int s**: **public void tick(int s).** Prográmalo para que el método haga que pasen **s** segundos en el reloj **(*)**. Puedes hacerlo llamando **s** veces al método tick() (así es poco eficiente pero fácil de programar). Ahora modifica el programa usando este método y verás que es mucho más fácil porque es suficiente con un bucle simple. Lo que has hecho es poner en Reloj la lógica (el código) de que pasen muchos segundos, en lugar de hacerlo en el main del programa. Así tiene mucho más sentido y es más fácil programar usando la clase Reloj.

(*) A esto se le llama sobrecarga de métodos. Sucede cuando una clase tiene varios método con el mismo nombre pero distintos parámetros. Es muy útil cuando queremos distintas variantes del mismo proceso, como en este caso que nos viene bien poder adelantar el reloj un solo segundo con **objeto.tick()** o varios segundos con **objeto.tick(segundos)**.

Ejercicio 5. Array de DNIs

Añade a la clase DNI el método **public DNI newRandomDNI() {...}** que instancie un nuevo objeto DNI con número de dni aleatorio y lo devuelva.

Escribe un programa que cree 10000 (diez mil) objetos DNI aleatorios, los imprima por pantalla y luego indique cuántos DNI hay con las letras 'A', 'E', 'I', 'O' y 'U'.