**Ejercicios fase I**

1. Escribe una clase llamada **Ordenador**, que disponga de un atributo con su nombre y sistema operativo que lleva, y otros para indicar el tipo de procesador y el cantidad de RAM. La clase debe contener un método llamado numerodeOrdenadores() que devuelve el número total de ordenadores que se han creado. Cada elemento deberá tener también un atributo q sea un numero *identificador único* y no modificable para cada elemento que se crea.

      Prueba la clase en un programa creando diferentes ordenadores que tengamos en nuestro almacen,

***Nota:****un atributo con valor común a todos los objetos se define como static*

1. Escribe un programa para representar el consumo de energía de una instalación eléctrica. Para ello, se dispondrá de una clase **AparatoElectrico** que representa a los aparatos conectados en la instalación eléctrica. Cada aparato tiene un nombre, un consumo eléctrico determinado por su potencia. Al encender un aparato eléctrico, el consumo de energía de toda la instalación, se incrementa en la potencia de dicho aparato. Al apagarlo, se disminuye el consumo en la misma cantidad. Inicialmente, los aparatos están todos apagados. Además se desea consultar el consumo total de la instalación.  ¿Cómo sabemos si un aparato está encendido?

 El programa debe declarar 3 aparatos eléctricos, un microondas de 800w, un radiador de 2000w y una plancha de 1200w. Se debe visualizar el consumo nada mas crear los objetos. Posteriormente, se enciende el microondas y la plancha, mostramos el consumo. Luego se apaga la plancha y se enciende el radiador, volver a mostrar el consumo.

1. Recoge números por teclado que representan las edades de varias personas, usando sc.next(), es decir como texto, **conviértelos** a numérico, súmalos y  calcula la media de  todos.

1. Crea una clase con un método que genere números aleatorios de forma infinita en un rango de 1 a 1000000. Cuando genere el numero 42089 debe generar una ***excepción***  y por tanto el programa terminar *(recuerda que cuando un método lanza una excepción el que llama a ese método debe recogerla con un try-catch)*. Indica cuantos números se han tenido que generar para obtenerlo.

1. Añade a la clase **Ordenador** una comprobación de que la memoria RAM no puede ser ni inferior a 2Gb ni superior  20Gb. En el caso que no cumpla la condición (compruébalo en el constructor y en el setMemoria), deberás lanzar una **excepción** **MemoriaRAMIncorrecta,**define en el mensaje la cantidad de RAM que se estaba intentando introducir.

**Ejercicios fase II**

1. Escribe un programa **Fechas** que muestre la fecha y hora actual  y la convierta según los formatos:
   1. “Viernes, día 20 de Enero del año 2017 y son las 13:30”.
   2. 15/04/16
   3. 2016.01.15 15:40:15:55
   4. viernes, Ene 20, '17
   5. 1:30 PM
   6. 13:30:55:624
   7. 2017.enero.20 13:30
   8. suma 25 dias
   9. que fecha era hace 5 dias
   10. cuantos dias faltan para la cremà de fallas
   11. cuantos meses quedan para Nochevieja

1. Escribe en Java una clase **Enumerados** que incluya los siguientes tipos enumerados:
   1. Meses del año
   2. Evaluaciones de un curso

En  archivos independientes define los siguientes enumerados públicos:

* 1. Ciclos de la familia de informática
  2. Categorías profesionales de informática

Crea dos variables de cada tipo. Una la inicializas con alguno de los  valores posibles y la otra asigna su valor recogiéndolo como cadena de texto por teclado y convirtiéndolo al enumerado, controla posibles excepciones de conversión. Compara sus valores y muestra resultado por pantalla. Importa el enumerado **b**. Lista todos los valores de uno de los enumerados.

1. Define una clase **BilleteTren** con los atributos: origen, destino, tipo (Turista, Superior, Lujo), precio base. Define un método **double** **aPagar()** que calcule cuánto cuesta el billete (precio base si es turista, preciobase mas 15% si es Superior y preciobase mas 25% si es Lujo). Incluye otro método **double** **aPagar(int descuento)** que calcula el precio a pagar teniendo en cuenta un descuento expresado en % sobre precio base. Y un tercer método **double aPagar(boolean mascota),** que en el caso de llevar una mascota incrementará el precio en 7%, calculado igual que**aPagar()**

## Ejercicios fase III

**1.** Programa **Trocear** que recoja varias palabras separadas por “=” por teclado. Indique por pantalla de cuantas palabras está compuesta y cuáles son éstas en líneas independientes.

**2. Padron**. Utilizando las clases Persona, Ciudadano y ExtranjeroTransito, crea una nueva clase**Padrón** que almacenará en una colección todas las personas de nuestro municipio. Para ello esta clase deberá implementar una interfaz **GestionaPadron**, que deberá tener los siguientes métodos:

 -         **Boolean darAlta(Persona p)**à almacenará la persona p comprobando que no estuviera registrada.

(método buscar sobrecargado)

-         **List<Persona> buscar(boolean candidato) à**si el parámetro candidato es true devolverá una lista de las personas que son candidatas, si es false las que no lo son.

-         **List<Persona> buscar(int edad) à**devuelve todas las personas que tienen la edad indicada por parámetro.

-         **List<Persona> buscar(String pais) à**devuelve todas las personas cuyo país de origen es el pasado por parámetro.

En la clase principal crea varias personas (entre 4 a 8) y un objeto Padron. Incluye estas personas en el padron y ofrece un menú para probar las tres implementaciones del método buscar.

**3.** Aplica la interfaz **Comparable** en una clase **Atleta**, donde tenemos nombre, codigoFederado, mejorMarca100m, mejorMarcaSalto, de manera que se ordenen según su mejorMarca100m de menor a mayor. Crea un programa con una lista de Atletas e introduce datos de al menos 4 atletas y ordenalos usando Collections.sort().

¿Qué deberíamos modificar para conseguir la ordenación de mayor a menor?

Última modificación: sábado, 1 de febrero de 2020, 21:58