Ejercicios Tema 7. Programación Orientada a Objetos I

*Todos los ejercicios de este boletín deben estar dentro de un **package** llamado **tema07** (siguiendo las reglas que hemos visto en clase, por ejemplo com.germangascon.tema07).*

1. Crea un programa para poder representar información sobre coches. Para desarrollar el ejercicio:
   a) Crea una clase llamada **Principal** que es la que contendrá el método main().
   b) Crea una clase llamada **Coche** que represente la siguiente información:
      - el modelo
      - el color
      - si la pintura es metalizada o no
      - la matrícula
      - el tipo de coche, que puede ser MINI, UTILITARIO, FAMILIAR o DEPORTIVO
      - el año de fabricación
      - la modalidad del seguro, que puede ser a terceros o a todo riesgo

   Crea un constructor con valores iniciales (los que tú quieras), otro constructor con parámetros, los getters, los setters y el toString con todos los atributos.
   - Crea varios coches utilizando los diferentes constructores y muestra sus características por pantalla.
   - Cambia algunos de los atributos de los coches y vuelve a mostrar sus características.

2. Escribe un programa que disponga de una clase para representar las asignaturas de un ciclo formativo:
   a) Una asignatura tiene un nombre, un código numérico y el curso en el cual se imparte.
   b) Los valores iniciales deben proporcionarse en el constructor.
   c) La clase tiene que tener métodos para obtener los valores de los atributos.
   d) El programa tiene que construir un objeto con los siguientes valores: nombre “Programación”, código 1017, curso 1.
      - A continuación, el programa deberá imprimir los valores del objeto por pantalla.

3. Crea un programa para gestionar alumnos. De cada alumno se desea saber:
   a) nia (número de identificación del alumno)
   b) nombre
   c) Apellidos
   d) fecha de nacimiento
   e) Grupo al que pertenece
   f) Teléfono de contacto

   El programa deberá presentar un menú que permita gestionar alumnos similar al siguiente:

   ****

   ** GESTIÓN ALUMNOS **

****

1. Nuevo alumno …
2. Baja de alumno …
3. Consultas …

---

0. Salir

- Nuevo alumno. Solicitará los datos del alumno, los validará y lo creará. ¡CUIDADO! No puede haber dos alumnos con el mismo nia.
- Baja de alumno. Solicitará el nia del alumno, si el nia corresponde a un alumno existente en el sistema, solicitará un mensaje de confirmación para borrarlo, en caso de confirmación positiva finalmente será borrado.
- Buscar alumnos. Mostrará el siguiente submenú:

  ****

  ** CONSULTAS **

  ****

  1. Por grupo …
  2. Por edad ...
  3. Por nia ...
  4. Por apellidos …

  ---

  0. Volver al menú principal

  - Por grupo. Solicitará el grupo y mostrará todos los alumnos que pertenecen a ese grupo.
  - Por edad. Preguntará una edad en años y el programa mostrará todos aquellos alumnos que tienen esa edad. Deberá calcularse a partir de la fecha de nacimiento.
  - Por nia. Preguntará el nia, buscará si existe algún alumno con ese nia y en caso afirmativo mostrará los datos de ese alumno.
  - Por apellidos. Preguntará parte de las letras de los apellidos y buscará aquellos alumnos que sus apellidos empiezan por esas letras.
  - Volver al menú principal. Volverá al menú principal del programa.

  Podemos ver un ejemplo de la salida de una de las consultas en la siguiente tabla:

Anida	Nombre	Apellidos	Data nacimiento	Grupo	Teléfono
1036652	Juan	Mengual	20-02-2002	1DAM	965174521
1025456	María	Piera	05-03-2004	1DAM	965165854
1035622	Vicent	Tormo	18-09-1991	1DAM	632546598

CONSIDERACIONES:

Con finalidad de agilizar las pruebas a medida que vamos haciendo el programa, crea un método que registre unos cuántos alumnos en el sistema con datos pseudo-aleatorios.

4. Crea un programa para poder representar puntos en dos dimensiones (x,y). Para desarrollar el ejercicio:
   a) Crea una clase llamada **Punto** que:
      - Tenga dos atributos de tipo double.
      - Tenga un constructor con dos parámetros de tipo double que inicialice los dos atributos.
      - Tenga un constructor por defecto (sin parámetros) que inicialice los dos atributos a 0.
      - Tenga un getter y un setter para cada uno de los atributos.
      - Tenga un método llamado distancia que reciba un parámetro de tipo Punto y que devuelva un double con la distancia entre el punto actual y el recibido como parámetro.
      - Tenga un método toString que permita muestrar las coordenadas del punto como se muestra en el siguiente ejemplo:
        - (2.0, 6.2)

   Una vez creada la clase Punto, desde la clase Principal crea varios objetos Punto y muéstralos por pantalla.

5. Crea una clase llamada **Circunferencia** que:
   a) Tenga un atributo de tipo Punto (que represente el centro del circunferencia) y uno de tipo double (que represente el radio de la circunferencia).
   b) Tenga un constructor con dos parámetros, uno de tipo Punto y otro double que inicialice los dos atributos.
   c) Tenga un constructor con tres parámetros de tipo double que inicialice los dos atributos.
   d) Tenga un constructor por defecto (sin parámetros) que inicialice los dos atributos al valor deseado.
   e) Tenga un getter y un setter para cada uno de los atributos.
   f) Tenga un método llamado distancia que reciba un parámetro de tipo Punto y devuelva un double (este método calculará la distancia desde el centro de la circunferencia hasta el punto pasado como parámetro).
   g) Tenga un método llamado area que no reciba ningún parámetro y devuelva un double con el área de la circunferencia.
   h) Tenga un método perimetro que no reciba ningún parámetro y devuelva un double con el perímetro/longitud de la circunferencia.
   i) Tenga un método toString que permita mostrar su información como se muestra en el siguiente ejemplo:
      - Círcunferencia de radio 2.5 cm situada en el punto (3.2, 4.0)
      - NOTA: Si el centro del círculo está situado en el punto (0, 0), el mensaje mostrado será:
        - Círcunferencia de radio 3.2 cm situada en el origen de coordenadas.

   Desde la clase Principal crea varios objetos Circunferencia y para cada uno de ellos, muestra donde está situado, su perímetro, y el área. Después muestra la distancia que hay entre dos de las Circunferencias creadas.

6. Crea un programa que permita gestionar las bicicletas que tiene una tienda. De cada **bicicleta** se quiere saber:
   a) la referencia (que actúa como campo único).
   b) la marca
   c) el modelo
   d) el peso en Kg.
   e) el tamaño de las ruedas en pulgadas
   f) si tiene motor
   g) la fecha de fabricación
   h) el precio
   i) el número de existencias que tienen de esa referencia

   El programa debe mostrar un menú similar al siguiente:

   ****
   ** GESTIÓN DE BICICLETAS **
   ****
   1.- Añadir bicicleta …
   2.- Vender bicicleta ...
   3.- Consultar bicicleta …
   4.- Mostrar stock

   0.- Salir

   - *Añadir bicicleta*. Solicitará los datos de una bicicleta y la añadirá al sistema. Se debe comprobar si la referencia introducida por el usuario existe, en caso afirmativo, se incrementará el stock de esa referencia, si no existe, solicitará el resto de datos para dar una nueva bicicleta de alta.
   - *Vender bicicleta*. Si escogemos la opción 2 nos pedirá la referencia de la bicicleta que queremos vender, comprobará que hay existencias y en el supuesto de que queden, mostrará un mensaje del tipo "bicicleta xxxxx vendida correctamente" y si no hay stock mostrará un mensaje indicando que no hay stock.
   - *Consultar bicicleta*. Si escogemos la opción 3 nos mostrará un submenú similar al siguiente:

     ****
     ** CONSULTA BICICLETA **
     ****
     1.- Consultar por referencia …
     2.- Consultar por marca …
     3.- Consultar por modelo …

     0.- Volver al menú principal

- *Consultar por referencia*. Solicitará la referencia y buscará si existe una bicicleta con esa referencia. En caso afirmativo mostrará la información asociada a esa bicicleta. Como el campo referencia es único no puede existir otra bicicleta con esa referencia.
- *Consultar por marca*. Solicitará la marca y mostrará las bicicletas de dicha marca.
- *Consultar por modelo*. Solicitará el modelo y mostrará las bicicletas de dicho modelo.
- *Volver al menú principal*. Volverá a mostrar el menú principal.

- *Mostrar stock*. Si escogemos la opción 4 del menú principal mostrará el stock que hay de cada bicicleta.
- *Salir*. Al escoger la opción 0 saldrá del programa.

CONSIDERACIONES:

- Para evitar tener que introducir datos cada vez que inicias el programa para poder hacer pruebas, crea varías bicicletas por código.

7. Crear una aplicación para gestionar un pequeño centro de salud de un pueblo para atender las urgencias. Cuando los pacientes llegan al centro son registrados en el sistema con los siguientes datos:
   - El número de la tarjeta sanitaria (Sip)
   - Nombre
   - Sexo
   - Edad
   - Fecha y hora de entrada al centro
   - Sintomatologia

   y esperan su turno en la sala de espera.

   Cuando un paciente es atendido se toman sus constantes vitales: la temperatura, las pulsaciones por minuto y los valores de la tensión arterial (sistólica y diastólica) se guardan en un atributo preRev que será vector de 4 posiciones.

   Se desea saber los pacientes que están siendo atendidos actualmente así como un histórico de los pacientes atendidos a lo largo del mes en curso.

Sip	Nombre	Sexo	Fecha de entrada	Hora de entrada	Edad	Sintomatologia	Temp	ppm	TenSis	Ten Dias	Fecha de alta	Hora de alta	Motivo del alta
94964183	Sonia Moran	M	15-12-2018	22:55	20	Dolor abdominal	36.5	75	130	60	16-12-2018	01:25	Mejora
37109682	Juan Soler	V	15-12-2018	23:21	38	Mareo generalizado	38.2	71	135	55	16-12-2018	01:39	Mejora
52046314	Pedro Ferrer	V	16-12-2018	02:44	81	Dolor fuerte en el pecho	36.4	98	175	98	16-12-2018	02:59	Defunción
92367866	Sergio Signes	V	16-12-2018	05:21	45	Escalofríos	40.1	115	180	110	16-12-2018	05:30	Derivación hospital
79058213	Ana	M	16-12-2018	07:50	21	Migraña	36.7	68	125	62			

	Peralta					intensa							
25152798	María Piera	M	16-12-2018	08:13	15	Contusión en el brazo derecho							

Figura 1: ejemplo del histórico de pacientes

Suponemos que el número máximo de pacientes que atenderemos en un día será de 40.

El programa mostrará un menú similar al siguiente:

****
** URGENCIAS **
****

1. Nuevo paciente …
2. Atender paciente …
3. Consultas …
4. Alta médica …

---

0. Salir

Veamos que debemos hacer en cada caso:

- *Nuevo paciente*. Solicitará los datos del paciente y la sintomatología (la fecha y hora de entrada las obtendremos automáticamente de la fecha y hora del sistema) y registrará el paciente en el sistema. Deberá comprobarse si el paciente ya está siendo atendido actualmente. Pueden existir varías entradas del mismo paciente pero solo una sin fecha de alta.
- *Atender paciente*. El paciente es atendido. Solicitará el sip del paciente, si se trata de un sip de un paciente que está esperando a ser atendido, es decir, está registrado en el sistema y no tien fecha de alta, mostrará los datos del paciente y solicitará los datos de las constantes vitales y las registrará al sistema.
- *Consultas*. Mostrará un submenú similar al siguiente:

  ****
  ** CONSULTAS **
  ****

  1. Por Sip …
  2. Por fechas …
  3. Estadísticas
  3. Mostrar histórico mensual

  ---

  0. Volver al menú principal

  - *Consulta por Sip*. Solicitará el Sip del paciente y, si existe en el histórico, mostrará los datos del histórico asociadas a dicho paciente. Recuerda que un paciente puede ser atendido más de una vez a lo largo del mes.

- *Consulta por fechas*. Solicitará una fecha de inicio y una fecha de finalización y mostrará todos los datos del histórico de pacientes que han sido atendidos entre esas fechas. Si no se especifica fecha finalización, mostrará desde la fecha de inicio hasta la fecha actual.
- *Estadísticas*. Mostrará las siguientes estadísticas:
  - media de la temperatura
  - media de pulsaciones por minuto
  - media de los dos valores de la tensión arterial
  - media de edad de los pacientes
  - porcentaje de pacientes que son hombres
  - porcentaje de pacientes que son mujeres
- *Mostrar histórico mensual*. Mostrará el histórico completo.
- *Volver al menú principal*. Volverá al menú principal de nuestro programa.

- *Alta médica*. Solicitará el Sip del paciente y un motivo del alta y dará el alta médica al paciente. La fecha y la hora de alta las cogerá automáticamente del sistema.
- *Salir*. Saldrá del programa.

Con el fin de agilizar las pruebas que vamos realizando en el programa, crea un método que registre unos cuántos pacientes en el sistema con datos pseudo-aleatorios.