UD03.POO.Clases
Ejercicios Propuestos

DAM1-Programación 2024-25

Sobre los ejercicios.

1. Organiza los ficheros de código fuente en los paquetes que se indican en cada apartado.
2. Nombra los ficheros y las clases con el nombre/código del ejercicio en negrita.
3. Utiliza un *comentario de documentación* para indicar tu nombre y apellidos como autor del código.
4. Opcional. Entrega el paquete comprimido en el Aula Virtual al terminar la UD.

Clases

Paquete: ejercicios`clases`

EP0711. Escribe la clase `MarcaPagina`, que ayuda a llevar el control de la lectura de un libro. Deberá disponer de métodos para incrementar la página leída, para obtener información de la última página que se ha leído y para comenzar desde el principio una nueva lectura del mismo libro.

EP0712. Implementa una clase que permita resolver ecuaciones de segundo grado. Los coeficientes pueden indicarse en el constructor y modificarse *a posteriori*. Es fundamental que la clase disponga de un método que devuelva las distintas soluciones y de un método que nos informe si el discriminante es positivo.

EP0714. Crea una clase que sea capaz de mostrar el importe de un cambio, por ejemplo, al realizar una compra, con el menor número de monedas y billetes posibles.

EP0715. Diseña la clase `Calendario` que representa una fecha concreta (año, mes y día). La clase debe disponer de los métodos:

- `Calendario(int año, int mes, int dia)`: que crea un objeto con los datos pasados como parámetros, siempre y cuando, la fecha que representen sea correcta.
- `void incrementarDia()`: que incrementa en un día la fecha del calendario.
- `void incrementarMes()`: que incrementa en un mes la fecha del calendario.
- `void incrementarAño(int cantidad)`: que incrementa la fecha del calendario en el número de años especificados. Ten en cuenta que el año 0 no existió.
- `void mostrar()`: muestra la fecha por consola.
- `boolean iguales(Calendario otraFecha)`: que determina si la fecha invocante y la que se pasa como parámetro son iguales o distintas.
- Implementa getters para obtener los atributos de un objeto Calendario.

Por simplicidad, solo tendremos en consideración que existen meses con distinto número de días, pero no tendremos en cuenta los años bisiestos.

Amplía el programa para tener en cuenta también los años bisiestos.

EP0716. Escribe la clase `Punto` que representa un punto en el plano (con un componente *x* y un componente *y*), con los métodos:

- `Punto(double x, double y)`: construye un objeto con los datos pasados como parámetros.
- `void desplazaX(double dx)`: incrementa el componente *x* en la cantidad *dx*..
- `void desplazaY(double dy)`: incrementa el componente *y* en la cantidad *dy*.

- `void desplaza(double dx, double dy)`: desplaza ambos componentes según las cantidades *dx* (en el eje *x*) y *dy* (en el componente *y*).
- `double distanciaEuclidea(Punto otro)`: calcula y devuelve la distancia euclidea entre el punto invocante y el punto `otro`.
- `void muestra()`: muestra por consola la información relativa al punto. Ejemplo: (1.5, 4.6)

EP0717. Jugador. Queremos gestionar la plantilla de un equipo de fútbol. Para ello vamos a crear la clase `Jugador`. De cada jugador se guarda el DNI, el nombre, la posición en el campo - para simplificar, los jugadores pueden ser porteros, defensas, centrocampistas y delanteros - y su estatura. Define la clase `Jugador` y un enumerado para la posición en el campo. Implementa un método para mostrar la ficha de un jugador y constructores variados que permitan crear:

- jugadores sólo con nombre.
- jugadores con nombre y posición
- jugadores con todos los datos.

Crea getters que permitan recuperar los atributos de los jugadores. Crea un método que permita cambiar a un jugador de posición.

Crea un pequeño programa principal con ejemplos de uso de la clase Jugador.

Punto Geográfico

PuntoGeografico. Investiga en Internet el funcionamiento del sistema GPS y las coordenadas de latitud y longitud. Puedes utilizar el siguiente prompt en chatGPT:

Prompt. Explícame el funcionamiento de la latitud y longitud en las coordenadas geográficas.

Implementa la clase `PuntoGeografico` con las siguientes especificaciones:

- Atributos privados para almacenar la `latitud` y `longitud` y una cadena de caracteres a modo de `nombre` o etiqueta.
- Un constructor que permita introducir los valores de todos los atributos y que devuelva una excepción si las coordenadas de latitud o longitud no son válidas.
- Un constructor por defecto que inicialice el punto al paso del meridiano de Greenwich por el Ecuador.
- Un constructor que permita introducir coordenadas de latitud y longitud válidas o que devuelva una excepción si no lo son.
- Un método que muestre por pantalla la información del punto en formato de grados decimales.

  ```
  public void mostrarGradosDecimales();
  ```

- Un método estático que devuelva un punto geográfico cuya latitud y longitud se generen aleatoriamente.

  ```
  public PuntoGeografico generarAleatorio();
  ```

- Un método que devuelve una cadena de caracteres como la siguiente:

  https://www.google.es/maps/@42.3716382,-8.6897279,12z

  Donde los valores coloreados se sustituyan por la latitud y longitud del punto.

  ```
  public String urlGoogleMaps();
  ```

- Un método que devuelve una cadena de caracteres como la siguiente:

  https://www.openstreetmap.org/#map=14/42.3716382/-8.6897279

  Donde los valores coloreados se sustituyan por la latitud y longitud del punto.

  ```
  public String urlOpenStreetMaps();
  ```

Investiga el funcionamiento de la API https://nominatim.org/ basada en OpenStreetMaps.

Añade a la clase los siguientes métodos

- Un método que devuelve una cadena de caracteres como la siguiente:

  https://nominatim.openstreetmap.org/reverse?format=xml&lat=42.3716382&lon=-8.6897279&zoom=14

  Donde los valores coloreados se sustituyan por la latitud y longitud del punto.

  ```
  public String urlNominatimXml();
  ```

- Un método que devuelve una cadena de caracteres como la siguiente:

  https://nominatim.openstreetmap.org/reverse?format=jsonv2&lat=42.3716382&lon=-8.6897279

  Donde los valores coloreados se sustituyan por la latitud y longitud del punto.

  ```
  public String urlNominatimJson();
  ```

User

Implementa la clase `User` con las siguientes especificaciones:

- Atributos privados para almacenar:
  - `username`, `password`, `email` como cadenas de caracteres.
  - `createdAt`, `lastLogin`, como LocalDateTime.
  - `enabled` como boolean.
- Un constructor que acepte username, password, email y enabled y que inicializará el atributo createdAt al valor del momento actual.
- Un constructor que acepte `username` y `password` y cree un usuario activo, en la fecha actual, y con `email` igual a `null`.
- Un constructor como el anterior pero que acepte sólo el `username` y lo cree con sin contraseña (igual a `null`).

- Sobreescribe (*overrride*) el método toString() para que al imprimir un objeto de clase User imprima el valor de su atributo username.

Viaje

Implementa la clase `Viaje` que representa un viaje compartido ofrecido por un conductor, que será un usuario (clase `User`) previamente creado, y que saldrá en una fecha y a una hora determinada desde un `PuntoGeografico` de origen hasta otro de destino. En cada viaje se registrarán también el número de plazas de pasajeros ofrecidas por el conductor.

- Crea al menos un constructor que permita crear un viaje con todos sus atributos.
- Crea getters y setters si es necesario.

- Crea un método mostrar que presente detalladamente y con claridad toda la información relativa a un viaje.
- Redefine el método toString() para que devuelva una cadena con el siguiente formato:

```
dd/mm/aaaa hh:mm - conductor - PuntoOrigen >> PuntoDestino
```

Añade un método que devuelva una cadena de caracteres como la siguiente:

https://www.openstreetmap.org/directions?engine=fossgis_osrm_car&route=42.38972%2C-8.71068%3B42.60752%2C-8.47149

Donde los valores coloreados se sustituyan por la latitud y longitud del punto.

```
public String urlOpenStreetMaps();
```