Ejercicios Tema 5. Programación orientada a objetos.

Diseña una clase **Punto** que modele puntos en el plano X-Y. Ten en cuenta las siguientes indicaciones:

Métodos:

- Método para recuperar y modificar las propiedades del punto.
- Método que devuelva el estado del objeto de forma textual, con el formato (x,y).
- Método que desplace el punto actual una distancia dada por los valores recibidos como parámetro para las coordenadas x e y.
- Método que mueva el punto actual a la nueva posición dada por las coordenadas recibidas como parámetro.
- Método que calcule la distancia entre el punto actual y un segundo punto recibido como parámetro.

  NOTA: Dados los puntos en el plano *A(x1,y1)* y *B(x2,y2)*; se define la distancia entre ellos de la siguiente forma:

$$\sqrt{(x2-x1)^2+(y2-y1)^2}$$

- Método que mueva el punto actual al centro de la pantalla.

  NOTA: La clase *ToolKit* del paquete java.awt dispone de una serie de métodos que permiten acceder al api nativo del S.O. directamente. Esta clase se instancia con el método *getDefaultToolkit()*.

Ejercicios Tema 5. Programación orientada a objetos.

Realiza un programa que permita trabajar con **Figuras Geométricas**. En concreto con círculos, rectángulos, cuadrados y triángulos.

Para ello debes diseñar las clases necesarias para modelar cada figura geométrica, con sus datos y métodos.

Indicaciones:

- Un círculo se creará con un radio determinado.
- Un rectángulo se creará con un acho y un alto.
- Un cuadrado se creará con un valor determinado para su lado.
- Un triángulo se creará con una base y altura determinados.

Además, todas las figuras tendrán su centro en un punto del plano x-y (utiliza la clase Punto ya diseñada).

De todas las figuras nos interesa conocer su área y perímetro, además de la diagonal (con 2 decimales) para los rectángulos y cuadrados.

Una figura se puede desplazar una distancia dada por los valores para las coordenadas x e y de su centro. También se pude mover a otro punto en el plano con respecto a su centro.

Todas las figuras deben incluir el método *toString()* el cual devolverá su representación textual.

Implementa el/los método/s necesario/s para permitir la ordenación de figuras por área ascendentemente.

Ejercicios Tema 5. Programación orientada a objetos.

Diseña la clase **Persona**, teniendo en cuenta las siguientes indicaciones:

Datos:

- DNI
- Nombre
- Apellidos
- Sexo (*char*)
- Fecha nacimiento (*LocalDate*)
- Edad en años.
- Altura en cm.
- Peso en kg.
- Casado (*boolean*)

Constructor:

- Para crear una persona, será obligatorio proporcionar un DNI válido, es decir, un número de 8 dígitos.

Métodos:

- Métodos para modificar todas las propiedades (menos el DNI y la edad), y para recuperar sus valores.

  ➔ La fecha de nacimiento será mostrada con el formato día-mes-año (el mes con tres letras). Ejemplo: 27-nov-2014.

  ➔ La propiedad *Edad*, deberá estar acorde a la fecha de nacimiento.

- Método que calcula la letra del DNI de la Persona.
- Método que devuelva los datos de una persona en formato cadena.
- Método que determina si la persona está en su peso ideal (0,75 * (altura en cm. – 150) + 50). Devuelve un -1 si está por debajo de su peso ideal, un 0 si está en su peso ideal y un 1 si tiene sobrepeso.
- Método que devuelve un booleano indicando si la persona es o no mayor de edad.
- Método que devuelva la edad de la Persona.

Ejercicios Tema 5. Programación orientada a objetos.

Diseña la clase **Racional**, teniendo en cuenta las siguientes indicaciones:

Atributos:

- Numerador
- Denominador

Constructor/es:

- Para crear un Racional (numerador y denominador) se darán valores al numerador y denominador.

Métodos:

- Métodos para recuperar/asignar valor a los atributos.
- Métodos para realizar las siguientes operaciones:
  - sumar(Racional r): suma dos Racionales, el actual y el recibido como parámetro.
  - restar(Racional r): resta al Racional actual el recibido como parámetro.
  - producto(Racional r): multiplica dos Racionales, el actual y el recibido como parámetro.
  - dividir(Racional r): divide el Racional actual entre el Racional recibido como parámetro.
  - toDecimal(): devuelve la representación numérica del Racional.

  En todos los casos el resultado es un nuevo Racional, simplificado al máximo.
- *toString():* devuelve el Racional con el siguiente formato:

  `<numerador>/<denominador>`

  - equals(Racional r): determina si el Racional actual es igual al pasado como parámetro.
  - compareTo(Racional r): determina si el Racional actual es mayor, menor o igual que el pasado como parámetro.

Ejercicios Tema 5. Programación orientada a objetos.

Realiza un diseño orientado a objetos que simule el comportamiento de un televisor. Para ello ten en cuenta las siguientes indicaciones:

- El televisor se creará con un nº de serie alfanumérico, un tamaño en pulgadas y el nº de canales.
- Puede encenderse y apagarse.
- Nada mas encenderse el televisor se pone en el canal 1.
- Para cambiar de canal se puede hacer de uno en uno (ascendente o descentemente), o cambiar a un nº de canal concreto de forma directa. En este caso, si el nº de canal no existe, el televisor se mantendrá en el último canal válido en el que estaba.

  El estado de los canales es cíclico, es decir, si el televisor está en el ultimo de los canales y se pasa al siguiente, entonces se sitúa en el primer canal, y de forma análoga si el televisor está en el canal 1 y se pasa al anterior, entonces el televisor se situará en el ultimo de los canales.
- El volumen está comprendido dentro del rango [0,30]. Éste se puede subir y bajar de una unidad en una unidad.

  También dispone de una opción "mute" para quitar el sonido, de tal forma, que cuando se vuelva a poner, tendrá el volumen que tuviera al quitarlo.

  En el caso de que se baje el sonido hasta el valor 0, será equivalente a ponerlo en mute. Y cuando se vuelva a subir, se "desmuteará".