Tecnología y paradigmas de programación. Laboratorio 2. Grupos 1, 2 y 3. Enunciado prácticas presenciales.

Ejercicio 1. Crear un proyecto nuevo de tipo aplicación de consola. Añadir a la solución, como proyecto existente o de otra forma, el proyecto de la clase anterior (geometria) en donde se definía el tipo Punto2d. Implementar las siguientes funciones (ver `utility.classes` en solución `encapsulation`, `parameter.passing`, `named.optional.parameters` en `overload`):

1. IncrementaDosDouble. Recibe cuatro double, los dos primeros por referencia. Modifica el valor de los dos primeros incrementándolos en el valor indicado por el tercer y cuarto parámetros, respectivamente. Por ejemplo IncrementaDosDouble(ref x,ref y,1.5,2.0) suma 1.5 a x y 2.0 a y.
2. GeneraTrayecto. Recibe la longitud (int, valor por defecto 10), coordenadas x e y de inicio (double, por defecto 0.0 ambas), incremento para x e y (double, valores por defecto 1.0). Genera un trayecto, representado como un array de Punto2d en el que cada elemento salvo el primero se obtiene incrementando las coordenadas de los anteriores en las cantidades indicadas. Usa la función del apartado 1. Ver proyecto `arrays` en solución `encapsulation`.
3. MuestraTrayecto, recibe un array como el descrito y un booleano opcional, que es true si se quiere mostrar sólo los puntos en el primer cuadrante (x e y positivas o cero) false si se quieren mostrar todos los puntos. Muestra por consola los Punto2d del trayecto.
4. Longitud, recibe un array como el descrito, devuelve su longitud (usar el método Distancia de Punto2d).
5. LongitudInicioFin, recibe un array como el descrito y, por referencia sólo de salida, tres parámetros más en los que se devuelve la longitud del trayecto, el primer punto y el último. ¿Que sucede si se modifican los parámetros después de la llamada?. Mostrar el recorrido para comprobarlo.
6. En Main declarar un array de Punto2d para representar un trayecto, incluir las llamadas oportunas a las funciones implementadas. Si es el caso, con distinto número de parámetros o en distinto orden.

Ejercicio 2. Examinar `extension.methods` en la solución `overload`. Crear un proyecto de tipo biblioteca de clases que extienda int, con los siguientes métodos (los ejemplos se han escrito usando constantes por simplicidad):

1. InvierteOrden, devuelve el entero que se forma con las cifras en orden inverso. Ej, 123456.InvierteOrden() devuelve 654321.
2. EsPrimo, devuelve true si el entero es primo, false en caso contrario. Ej, 13.EsPrimo() devuelve true, 14.EsPrimo() devuelve false.
3. Mcd y Mcm, opcional. Ej, 24.Mcd(36) es 12.
4. Concatena, opcional. Ej,123.Concatena(456) es 123456.
5. Crear un proyecto de tipo aplicación de consola de prueba.

Ejercicio 3. Crear un proyecto de tipo test unitario usando el código fuente que se proporciona. Implementar la clase correspondiente, métodos y operadores, de modo que pasen los tests. Ver `operators` en `overload`.

Ejercicio 4 (Opcional). Crear un proyecto de tipo biblioteca de clases, implementar la clase Circunferencia con los siguientes atributos, métodos y operadores:

1. Atributo Centro, de tipo Punto2d.
2. Atributo Radio, de tipo double.
3. Método tangenteCentro, recibe un punto, devuelve la circunferencia tangente de centro ese punto. Por concretar, si hay dos posibilidades, la de menor radio, no es un aspecto crítico.
4. Operador <, devuelve true si el operando izquierdo está contenido en el operando derecho, false en caso contrario. Será necesario definir también el operador >, por diseño del lenguaje.
5. Constructores, destructor, propiedades, etc. y una aplicación de consola de prueba.

NOTA: recordar entregar la tarea autónoma antes del siguiente laboratorio, en el examen de prácticas se usará.