Examen de Prácticas de Programación I
(Ingeniería Informática)

Febrero 2008

Se dispone de una clase Java que ayuda en los cálculos necesarios de las trayectorias de un robot móvil que se mueve en un plano X-Y. La clase se llama `Trayectoria` y tiene dos tablas que contienen respectivamente las coordenadas X e Y de los puntos de la trayectoria que debe seguir el robot. El punto i-ésimo tiene su coordenada X en `x[i]` y su coordenada Y en `y[i]`. Además, la clase guarda el punto actual, al que se debe dirigir el robot inmediatamente (coordenadas `actualX` y `actualY`), y el índice del punto objetivo de la trayectoria, que es el punto de la trayectoria al que debe dirigirse el robot a medio plazo. El punto actual no tiene por qué ser un punto de la trayectoria; habitualmente será un punto intermedio entre dos puntos de la trayectoria. Los atributos y cabeceras de métodos se indican a continuación:

```
/**
 * Clase que contiene una trayectoria en el plano X-Y
 */
public class Trayectoria
{
    // puntos de la trayectoria
    private double x[]; // milímetros
    private double y[]; // milímetros

    // punto a donde nos queremos dirigir ahora mismo
    // (punto actual)
    private double actualX; // milímetros
    private double actualY; // milímetros

    // indice del punto de la trayectoria al que queremos llegar
    // (punto objetivo)
    private int objetivo;

    /**
     * Constructor de la trayectoria
     */
    public Trayectoria(double x[], double y[],
                       double actualX, double actualY) {...}

    /**
     * Número de puntos
     */
    public int numPuntos() {...}

    /**
     *  Coordenada x del punto actual,
     *  a donde nos queremos dirigir ahora mismo
     */
    public double coordX() {...}

    /**
     *  Coordenada y del punto actual,
     *  a donde nos queremos dirigir ahora mismo
     */
    public double coordY() {...}

    /**
     * Comprueba si hay puntos seguidos repetidos
     * Se consideran dos puntos iguales si están
     * a menos de 0.1 milímetros de distancia
     */
    public boolean hayRepetidosSeguidos() {...}

    /**
```

```
     * Distancia de (px,py) al punto i-ésimo de la trayectoria
     */
    public double distancia(double px, double py, int i) {...}

    /**
     * Calcula el nuevo punto objetivo, como el indice
     * del punto de la trayectoria mas cercano
     * a (px,py). Además, lo retorna
     */
    public int masCercano(double px, double py)  {...}

    /**
     * Si el punto actual está cerca del objetivo, y éste
     * no está al final de la trayectoria, incrementar el índice
     * del punto objetivo
     */
    public void actualizaObjetivo() {...}

    /**
     * Calcula el siguiente punto al que hay que dirigirse
     * durante el proximo intervalo tiempo
     * para seguir la trayectoria hacia el punto objetivo
     * a la velocidad indicada en mm/s
     */
    public void avanza(double velocidad, double tiempo) {...}
}
```

La implementación de la clase está disponible en la página Web de la asignatura.

Se pide:

1) Añadir a la clase `Trayectoria` el método `pinta`, que pinta la trayectoria y el punto actual en una ventana de la clase `Dibujo`. Suponer que las unidades de pixels de pantalla son equivalentes a milímetros, de modo que la conversión de distancia a pixel es una simple conversión de real a entero. Suponer también que las coordenadas tienen valores positivos que caben en una ventana de tamaño normal. (3 puntos)

2) Crear una nueva clase, llamada `GestionTrayectorias` con un programa principal que haga lo siguiente (7 puntos):
   - Lee de teclado las coordenadas del punto actual, y el número de puntos de la trayectoria (puede usarse un objeto de la clase `Lectura`, o la misma ventana de la clase `CajaTexto` que se menciona a continuación). Luego, leer con una ventana de la clase `CajaTexto`, las coordenadas x e y de la trayectoria.
   - Crea un objeto de la clase `Trayectoria` usando los datos leidos de teclado
   - Pinta la trayectoria con el método `pinta`
   - Llama a `masCercano` para calcular el punto más cercano al punto actual, y muestra en la pantalla el valor obtenido.
   - Llama a `avanza` repetidas veces, hasta que en dos veces sucesivas se obtiene el mismo punto actual (que se puede consultar con `coordX` y `coordY`). A cada paso, muestra en pantalla las coordenadas del punto actual.