Programación orientada a objetos. Cuarto ejercicio.

14 de mayo de 2008.

Parte 1

Es tiempo de dar un poco de vida a tu mundo.

Los mundos que vamos a usar son muy sencillos, y su vida se cuenta en ciclos. En cada ciclo, se les pide a los elementos del mundo que actúen, modificando su estado, pidiéndole cosas a otros elementos del mundo que conozcan o puedan encontrar, saliendo del mundo y muriendo, o moviéndose de lugar. Para poder dar vida a su mundo, tienen que tener una forma de ir diciéndole cuándo transcurrió un nuevo ciclo.

Para ésto, les proveemos un tipo adicional, el IExecutor, que se encarga de ejecutar acciones registradas con él.

El ejecutor tiene el siguiente tipo:

```
public interface IExecutor
{
    void AddAction(IAction action);
    Boolean ContainsAction(IAction action);
}
```

Las acciones que pueden agregarse al ejecutor tienen el siguiente tipo:

```
public interface IAction
{
    void Run();
    Boolean Done { get; }
}
```

Cuando se agrega una acción al ejecutor, este la ejecutará una y otra vez hasta que la acción retorne verdadero en su operación `Boolean Done { get; }`, que dice si la acción terminó o debe seguir ejecutándose. La operación `void Run();` es implementada por cada acción dependiendo de que se quiera hacer.

Para poder mostrar la evolución del mundo luego de cada ciclo, es cuestión de registrar una acción con el ejecutor que le mande un mensaje diciendo al mundo que paso un ciclo y luego pida a la interfaz de usuario que vuelva a dibujar el mundo. Cada vez que el ejecutor ejecute la acción, ésta le avisa al mundo que transcurrió un ciclo, y éste se encarga de actualizar su estado y el de sus cosas como sea conveniente.

Ustedes deberán definir esta acción, el mensaje a enviar al mundo para avisarle que transcurrió un ciclo y el código en el mundo para actualizar su estado.

El tipo IExecutor es implementado por MainGui, por lo cual deberían pedirle a la instancia de MainGui que ya utilizan que agregue las acciones que necesitan. No es lo más conveniente desde el punto de vista de POO hacer que MainGui implemente IExecutor (¡se podría decir que ya tenía bastantes responsabilidades!), pero simplifica problemas de sincronización entre hilos que en otro caso deberían resolver ustedes.

Parte 2

¡Implementa tus primeras IThing activas!.

En este caso, te pedimos que implementes una colonia de abejas. Las colonias tienen dos tipos de abejas, reinas y trabajadoras (es una colonia anti-bélica, por lo que no hay guerreras). Ambos tipos de abeja son bastante tontas. Las reinas se dedican a pasear por el mundo sin hacer demasiado, simplemente vagando aleatoriamente. Las trabajadoras, simplemente siguen a la reina abeja siempre que ésta está cerca.

Para acercarse a la reina, las abejas trabajadoras tratan de seguir el camino más corto hacia ella, una línea recta. Para simplificarte el trabajo, te proveemos un breve fragmento de código que muestra como calcular la siguiente posición de una abeja trabajadora si se sabe cual es la posición de la reina abeja:

```
void MoveToQueen(IWorld world, IPoint workerPosition, IPoint queenPosition)
{
        Int32 dx = queenPosition.X - workerPosition.X;
        Int32 dy = queenPosition.Y - workerPosition.Y;
        Int32 steps = Math.Max(Math.Abs(dx), Math.Abs(dy));
        if (steps != 0)
        {
           Int32 x = (Int32)Math.Round(workerPosition .X + (dx / (Double)steps));
           Int32 y = (Int32)Math.Round(workerPosition .Y + (dy / (Double)steps));

           // solo moverse si el lugar no está ocupado. No es requerido, si quieren
           // prueben como se comporta su programa si se mueven a un
           // espacio ocupado.
           if (world.GetThing(x, y) == null)
           {
                   // aquí hay que mover la trabajadora a esa posición...
                   // deberes para el lector.
              MoveTo(world, x, y);
           }
        }
}
```

Ten en cuenta que no decimos nada más sobre el comportamiento de las reinas y trabajadoras ni definimos que consideramos 'cerca' en 'siempre que ésta está cerca'. Tampoco decimos que pasa cuando hay más de una reina en juego (¿a quién siguen las trabajadoras?). Ni que sucede cuando tu reina está rodeada de trabajadoras y no tiene a donde moverse. Impleméntalo como mejor te parezca, y si quieres prueba diversas variaciones para ver como se comporta tu programa. No nos importa que tus abejas sean Turing, solo que implementes al menos abejas tontas que se muevan.

Parte 3

Permite definir abejas y reinas abejas en tu archivo de configuración.

Para eso te sugerimos el siguiente formato, pero puedes modificarlo para agregar otros atributos que requieras en tu programa (velocidad de la reina, radio de percepción de la trabajadora, colores, probabilidad de perderse de la trabajadora).

```
<bee-worker id="abejita trabajadora">
```

Para una abeja trabajadora.

```
<bee-queen id="abeja reina">
```

Para una reina.

Recuerda que un programa *debe* incluir casos de prueba y documentación ndoc. Los casos de prueba deben asegurarse de que cada método de cada clase que programaste funcione como es debido. Utiliza ésto como una herramienta para trabajar menos y más seguro, y no como un requisito "burocrático".

Reglas de colaboración: Los ejercicios son **individuales**. Puedes diseñar una solución en conjunto con otros compañeros y aprovechar comentarios o correcciones de los profesores, pero debes entregar un código que tú comprendiste, escribiste, compilaste, ejecutaste y probaste. Si entregas una clase que a nuestro criterio es idéntica a la de un compañero, puedes estar en problemas.

Entregas tardías: No se aceptarán entregas fuera de fecha. Los trabajos se entregan para que sigas constantemente el curso y no con objetivo de evaluarte (pese a que influyen en tu evaluación final en caso de que estés por exonerar o en riesgo de perder el curso). Un trabajo a las apuradas no tiene valor en este contexto.

Entregas por email: No se aceptarán entregas por email excepto que Web Asignatura no esté disponible seis horas antes a la fecha final de entrega. La entrega por email debe enviarse a **todos** los profesores y pedirles confirmación de entrega. Es tu responsabilidad asegurarte de que el trabajo haya sido recibido.