Programación orientada a objetos. Tercer ejercicio.

1 de mayo de 2008.

Parte 1

Hasta el momento tu programa te permite mostrar un mundo con cosas por pantalla en base a un archivo de *tags*. Sin embargo, no provee una interfaz de usuario que te facilite el uso del programa y te permita explorar más fácilmente lo que puedes hacer con tu mundo.

Como no queremos que pierdas el tiempo aprendiendo como usar las ventanitas en C#, nosotros ya te resolvemos el problema y ¡te damos la interfaz programada!. Claro que, siguiendo buenas practicas de POO, la interfaz de usuario no hace demasiado (solo despliega ventanas y recibe instrucciones del usuario) y delega la mayor parte del trabajo a un objeto instancia de una clase que **tu** debes programar y que debes darle al objeto responsable de la interfaz de usuario para que use.

La interfaz de usuario que ya te proveemos esta programada en un nuevo proyecto, llamado UcuLifeLib, en una clase que implementa la siguiente interfaz:

```
public interface IMainGui: IGui {
    void AddWorld(String worldName);
    Boolean ContainsWorld(String worldName);

    void AddThing(String thingName);
    Boolean ContainsThing(String thingName);

    void AddGuiListener(IGuiListener listener);
    Boolean ContainsGuiListener(IGuiListener listener);

    void Run();
}
```

Como puedes ver, la interfaz IMainGui extiende a IGui por lo que además de proveer las operaciones declaradas en la propia interfaz, provee operaciones para dibujar una pizarra (en este caso, sobre la interfaz).

La primera operación, void AddWorld(String worldName), te permite registrar un mundo en la interfaz pasando su nombre como argumento. A partir de ese momento, la interfaz mostrará un ítem en un menú cuyo texto es el nombre del mundo, y le permitirá al usuario seleccionarlo.

La operación void AddThing(String thingName), te permite registrar una cosa en la interfaz pasando su nombre como argumento. Al igual que con el mundo, la interfaz mostrará un ítem en un menú cuyo texto es el nombre de la cosa.

Ten en cuenta que, hasta ahora, tu archivo de *tags* sólo permite especificar mundos con un identificador y cosas asociadas a un mundo. Para este ejercicio, el archivo deberá poder incluir cosas que no están asociadas a ningún mundo particular, pero que incluyen un nombre que las identifica.

Por lo menos, debes soportar definir bloques como en el ejercicio 2, siguiendo el siguiente formato:

```
<block id="un bloque"/>
```

En este caso, definirías un bloque suelto con nombre 'un bloque'. Quedará clara su utilidad cuando termines de leer la letra.

Notarás que en la interfaz hay una operación void AddGuiListener(IGuiListener listener). Esta operación permite registrar un "oyente" (u observador) de la interfaz de usuario para que ésta le reporte los eventos y acciones que haga el usuario.

Este sistema de observadores es muy común en programas con interfaz de usuario. En general, los eventos pueden ser de diverso tipo, por ejemplo, que alguien apretó un botón, ingresó un texto, cerró una ventana, etc., pero en nuestro caso solo usaremos tres: cuando se seleccione un mundo en el menú de mundos, cuando se seleccione una cosa en el menú de cosas, o cuando se haga click en una celda de la pizarra que se está mostrando por pantalla.

Cada vez que se genera un evento, la interfaz invoca una operación apropiada sobre el observador. La operación es una de las provistas en el tipo del observador:

```
public interface IGuiListener{
        void WorldSelected(String worldName);
        void ThingSelected(String thingName);
        void CellSelected(IPoint selectedCell);
}
```

Cuando la interfaz detecta que alguien seleccionó un mundo, le avisa al observador invocando la operación void WorldSelected(String worldName).

Cuando la interfaz detecta que alguien seleccionó una cosa, le avisa al observador invocando la operación void ThingSelected(String thingName).

Cuando la interfaz detecta que alguien hizo click sobre la celda de un mundo mostrado en la interfaz, le avisa al observador invocando la operación void CellSelected(IPoint selectedCell), donde selectedCell corresponde al punto de la pizarra seleccionado en la interfaz.

Tu deberás programar una clase que implemente el tipo del observador para procesar el evento y tomar alguna acción.

La operación void Run() de IMainGui puedes llamarla en lugar del Application.Run() que utilizabas anteriormente.

Debes tomar las siguientes acciones dependiendo del mensaje recibido:

void WorldSelected(String worldName): Debes mostrar en pantalla el mundo identificado por el nombre en el argumento.

void ThingSelected(String thingName): Debes recordar cual fue el nombre de la cosa que te pasan en el argumento, para poder procesar el siguiente evento.

void CellSelected(IPoint selectedCell): Debes agregar al mundo la cosa que habían seleccionado en el evento anterior, en la posición pasada por argumento, y luego volver a pintar el mundo (mediante la operación void Redraw(IBoard board) que usaste para el ejercicio anterior).

El siguiente es un ejemplo del flujo de eventos:

1. Se inicia el programa y creas un objeto de tipo IMainGui (en base a la clase que nosotros te proveemos)
2. Registras en el IMainGui todos los nombres de mundos y nombres cosas definidos en el archivo de *tags*.

3. Registras el observador que tu implementaste con IMainGui y ejecutas el método Run() del objeto de tipo IMainGui para iniciar la aplicación.
4. El usuario selecciona un nombre de mundo desde el menú mostrado por IMainGui.
5. IMainGui avisa a tu observador del evento enviándole el mensaje WorldSelected(String worldName).
6. Tu observador muestra el mundo asociado a ese nombre en IMainGui.
7. El usuario selecciona el nombre de una cosa (por ejemplo, "un bloque" como en el *tag* del comienzo) desde el menú mostrado por IMainGui.
8. IMainGui avisa a tu observador del evento enviándole el mensaje ThingSelected(String thingName).
9. Tu observador recuerda el nombre de está cosa seleccionada.
10. El usuario selecciona una celda de la pizarra mostrada por la interfaz.
11. IMainGui avisa a tu observador del evento enviándole el mensaje CellSelected(IPoint selectedCell).
12. Tu observador agrega la cosa que había sido seleccionada previamente (en el aso 8) al mundo en la posición pasada por parámetro y vuelve a pintar el mundo.

- **ATENCIÓN 1:** el argumento selectedCell es el punto de la pizarra, no del mundo (¡una celda del mundo puede ocupar varias celdas de la pizarra!). Si quieres saber a que punto del mundo corresponde, debes dividirlo por el tamaño de celda usado por el pintor.
- **ATENCIÓN 2:** las cosas que agregas a tu mundo deberían ser siempre cosas distintas. Es decir, no deberías tener *la misma* cosa agregada dos veces a tu mundo. Por lo tanto, cuando IMainGui te avisa que se seleccionó una celda y debes agregar allí la cosa que había sido seleccionada previamente, deberías agregar una nueva copia de esa cosa.

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.