Programación orientada a objetos. Obligatorio.

31 de mayo de 2008.

Deben implementar un *framework* de simulación de agentes discreto.

Su sistema debe permitir simular mundos con distintos tipos de agentes, modificar el mundo mientras es simulado, iniciar y detener la simulación, y todo un conjunto de prestaciones mínimas que uno esperaría en un sistema como éste.

Para ésto se **agregan** las siguientes prestaciones al sistema y un nuevo tipo de agentes para que realicen pruebas.

Cambios en Observadores

IMainGui fue modificado para respetar el ISP en las interfaces de observadores. Ahora existen tantas interfaces de observador como eventos emite MainGui (hay una para cuando el mundo es seleccionado, otra para cuando una cosa es seleccionada, etc.).

Las interfaces para los observadores están bajo el namespace UcuLifeLib.Listener. Lean la documentación de las interfaces en el código que les proveemos para entender como se usan.

Pueden implementar las interfaces de los eventos que desean recibir y luego registrarlas en un IMainGui de la siguiente forma:

```
using UcuLifeLib.Listener;

using UcuLifeLib;

// variable que apuntará a un objeto que implementa al observador
IWorldSelectedListener listener;
// variable que apuntará a MainGui
IMainGui gui;
/*
 * importantes cantidades de código altamente complejo, correctamente distribuido
 */
///registrar listener en el Gui
gui.AddListener<IWorldSelectedListener>(listener);
```

A partir de ese momento, IMainGui avisará a listener cada vez que se seleccione un mundo en la interfaz. La notación gui.AddListener<IWorldSelectedListener> está usando genéricos, que serán vistos las próximas clases. Sin embargo, el uso en este caso es bastante intuitivo, está diciendo que registre al observador pasado por parámetro para recibir eventos de selección de mundo.

La interfaz IGuiListener sigue existiendo para soportar compatibilidad con versiones anteriores (¡algo que toda biblioteca de clases debería hacer dentro de lo posible!). Sin embargo, les recomendamos que en lo posible no la utilicen y usen las nuevas interfaces; debería ser cuestión de cambiar :IGuiListener por :IWorldSelectedListener, IThingSelectedListener, ICellSelectedListener y el código donde registran al observador.

Parte 1

Permitan al usuario eliminar cosas de su mundo.

Para eliminar una cosa, el usuario selecciona la opción 'Eliminar' del menú de cosas y luego hace click en la celda donde está la cosa para eliminar.

Para que MainGui avise a su programa que el usuario seleccionó 'Eliminar', deben registrar un observador con MainGui que implemente la interfaz IRemoveListener.

```
public interface IRemoveListener : IListener {
    void RemoveClicked();
}
```

MainGui sigue avisando cuando se seleccionó una celda de la forma habitual (al observador registrado como IGuiListener o como ICellSelectedListener).

Parte 2

Permitan al usuario mover cosas de su mundo.

Para mover una cosa, el usuario selecciona la opción 'Mover' del menú de cosas y luego hace click en la celda donde está la cosa que quiere mover, y luego click en la celda donde quiere poner a esa cosa.

Para que MainGui avise a su programa que el usuario seleccionó 'Mover', deben registrar un observador con MainGui que implemente la interfaz IMoveListener.

```
public interface IMoveListener : IListener {
    void MoveClicked();
}
```

Parte 3

Permitan al usuario detener la ejecución del mundo.

Para detener la ejecución del mundo, el usuario selecciona la opción 'Pausar/Continuar' del menú de mundos. Si la ejecución del mundo estaba pausada, se continua la ejecución, en otro caso, se pausa la ejecución del mundo.

Para que MainGui avise a su programa que el usuario seleccionó 'Pausar/Continuar', deben registrar un observador con MainGui que implemente la interfaz IPlayPauseListener.

```
public interface IPlayPauseListener : IListener {
    void PlayPauseClicked();
}
```

Parte 4

Permitan al usuario modificar la forma de su mundo.

Para modificar la forma del mundo, el usuario selecciona la opción 'Poner bloque' del menú de cosas y luego hace click en la celda que no quiere que sea más parte del mundo. A partir de ese momento, esa celda no es parte del mundo, ninguna cosa puede ocupar ese lugar de aquí en adelante. Este mecanismo permite crear mundos de formas arbitrarias (pero siempre compuestos por celdas cuadradas).

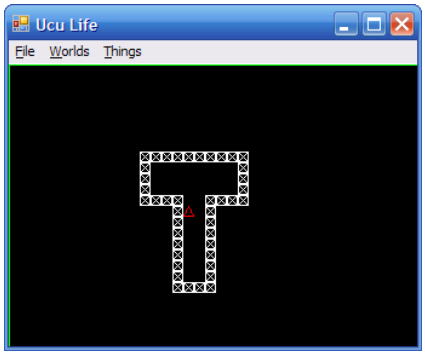

Figura 1: Mundo en forma de T con abeja adentro

Para que MainGui avise a su programa que el usuario seleccionó 'Poner Bloque', deben registrar un observador con MainGui que implemente la interfaz IBlockSelectedListener.

```
public interface IBlockSelectedListener: IListener {
    void BlockSelected();
}
```

Parte 5

Programen una cosa de tipo 'bomba'.

Cuando alguna cosa se acerca a una distancia poco prudencial de una bomba, la bomba explota, eliminando a la cosa y a ella misma.

Usando la interfaz de usuario, deberían poder poner una bomba para eliminar rápidamente un conjunto de abejas. La sugerencia es puramente académica y no debe llevarse a cabo en la vida real.

Parte 6

Programen un entorno de simulación de contagio celular.

Su programa debe tener por lo menos tres tipos de células distintas:

- Células sanas
- Células enfermas
- Anticuerpos

Las células sanas se reproducen creando nuevas células sanas siempre que haya suficiente espacio entorno a ellas.

Las células enfermas contagian células sanas que estén cerca. Una célula enferma muere eventualmente luego de algunos ciclos.

Los anticuerpos matan células enfermas.

Todos los tipos de células se deben mover en el mundo, como mejor les parezca.

Deben tener distintos tipos de enfermedades. Las enfermedades se caracterizan en base a:

- Virulencia: la probabilidad de contagiar a una célula sana.
- Mortalidad: la velocidad con la que matan a una célula enferma.

No se presume que la simulación tenga algo que ver con la realidad. Ya cursarán 'verdadera' simulación de sistemas en semestres posteriores.

Si una anticuerpo se encuentra con una abeja, no está muy claro que pasa.

Parte 7

Provean construcciones en su archivo de *tags* para representar el nuevo mundo y los distintos tipos de células, y a las bombas.

Notas

Es posible que si han trabajado bien durante los ejercicios previos, no les insuma demasiado tiempo completar las prestaciones del obligatorio.

Sin embargo, tengan en cuenta que es muy posible que para los ejercicios no hayan tomado realmente en cuenta la programación de casos de prueba, documentación del código, uso de genéricos, uso de precondiciones/postcondiciones, nombre de clases y variables, uso de interfaces en lugar de clases en la declaración de tipos siempre que sea posible etc..

Si logras implementar las prestaciones pedidas en poco tiempo, dejen su código en óptimas condiciones, y luego, si todavía tienen ganas de hacerlo mejor, vuelvan a releer el código e identifiquen casos donde aparezca código repetido (y este obligatorio tiene bastante lugar para que ésto suceda, por ejemplo, en la forma de moverse de las cosas, o de buscar otras cosas 'cercanas'). Donde tengan código repetido, algunas de las técnicas de reutilización vistas

(herencia, composición y delegación, genéricos) seguro pueden ayudarlos a eliminar esta duplicación de código y mejorar la calidad del mismo.

Si se encuentran trancados intentando que algo funcione, primero trabajen en la calidad del código y luego en su correctitud. Es más fácil hacer código correcto si tiene buena calidad.

El código provisto por los profesores es ahora de su entera propiedad. Siéntanse libres de modificarlo si quieren hacerlo (pese a que no lo sugerimos). También modifiquen, corrijan e integren todo lo necesario de ejercicios anteriores.

Por último, noten que varias de las partes son abiertas y no especifican completamente el comportamiento esperado (¿qué sucede si estoy moviendo y seleccionan otra opción? ¿debo eliminar la cosa del mundo al empezar a moverla? ¿Como se reproducen las células?, etc.)

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. Utilicen ésto como una herramienta para trabajar menos y más seguro, y no como un requisito "burocrático".

Reglas de colaboración: El obligatorio es en grupo de **dos**. Solo pueden formar grupo entre personas que tengan una cantidad similar de ejercicios entregados (un ejercicio de diferencia). Si entregan clases que a nuestro criterio son idéntica a las de otro grupo, pueden estar en problemas.

Entregas tardías: No se aceptarán entregas fuera de fecha.

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 su responsabilidad asegurarse de que el trabajo haya sido recibido.