

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN  
PROGRAMACION ORIENTADA A OBJETOS  
SEGUNDA EVALUACIÓN -2012-08-29

**TEMA 1. (5 puntos)**

**1**

**5**

**2**

**3**

**4**

**Verticales**

1. Layout Manager
3. Recorre colección

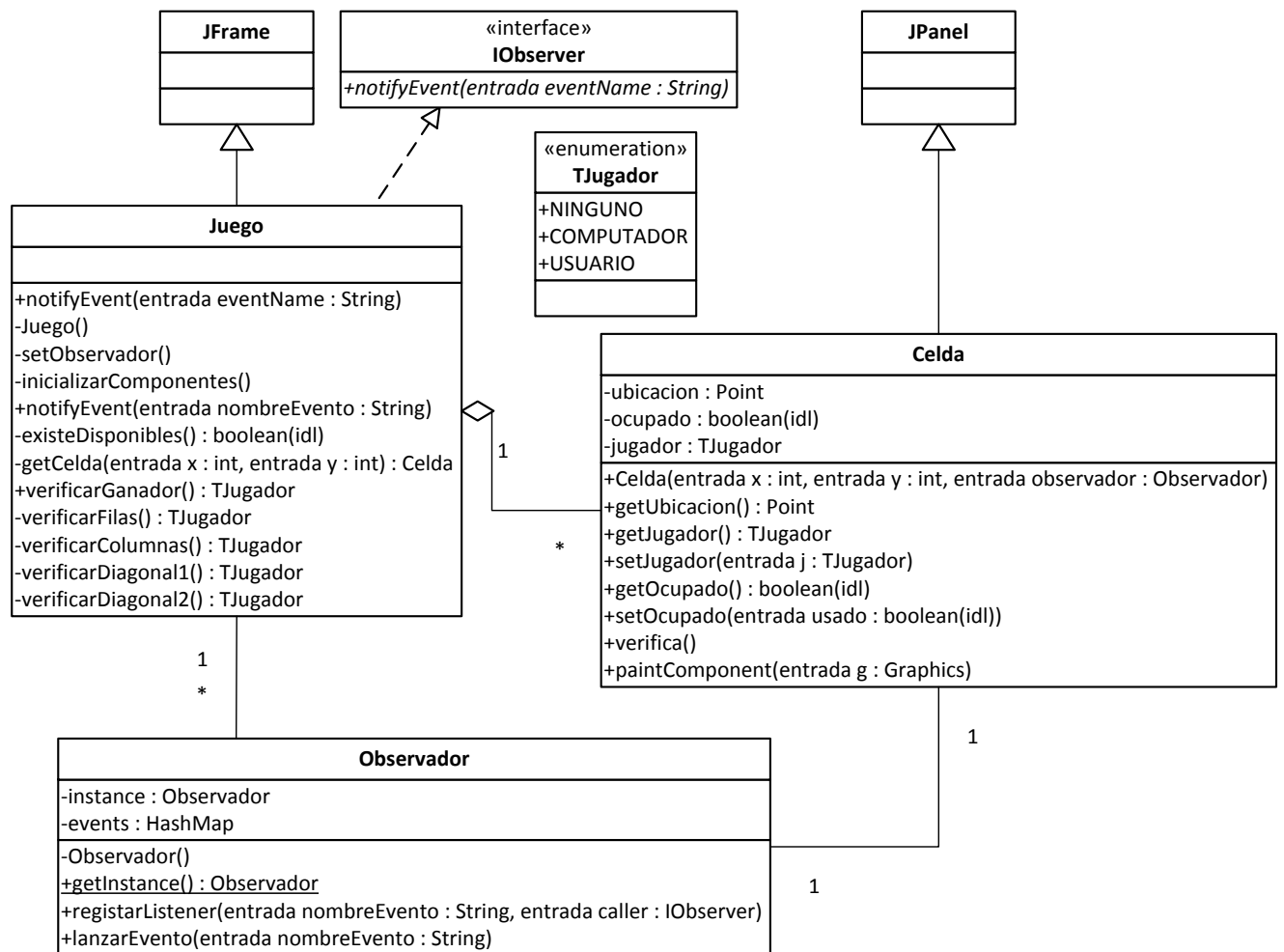
**Horizontales**

2. Manejo de excepciones
4. Pendiente de eventos
5. Adaptador

## This image shows a full page of blank white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page, providing a template for writing or drawing. There are no margins, text, or other markings on the paper.

### TEMA 3. (75 puntos)

Usted ha sido contratado por una empresa dedicada a los juegos y su supervisor quiere saber si usted cuenta con todos los conocimientos de la Programación Orientada a Objetos o no. Su supervisor le pide implementar el juego "Tres en raya" para lo cual le da las siguientes especificaciones.



El juego consta de un grid de 9 celdas en la cual cada celda puede estar en tres estados: Vacía, Usuario o Computador. Tenga en consideración que estos estados pueden ser utilizados para indicar quien es el ganador del juego.



#### Clase Juego

a) **(5pts.)** Cree la ventana Juego y establezca los siguientes parámetros:

- Título: Tres en raya
- Por defecto cerrar la ventana al salir
- Tamaño: 305 x 330
- No redimensionable
- Visible

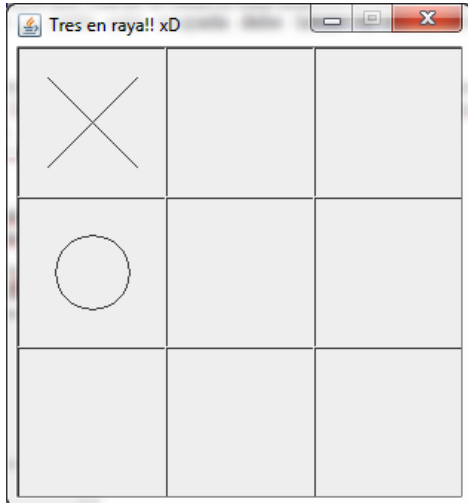
b) **(7 pts.)** Cree el método inicializarComponentes de la clase Juego, este método crea la interfaz de usuario de la ventana. Tenga en consideración que debe generar dinámicamente las Celdas del grid.

c) **(5 pts.)** Cree el método verificaDiagonal2, el cual dinámicamente verifica las celdas de la

diagonal que no es la diagonal principal y establece si hay un ganador o no.

### Clase Celda

- d) **(5 pts.)** Cree el método constructor de la clase Celda:
- Inicialice los atributos
  - Establezca el tamaño de la celda en 100 x 100
  - Establezca el color de fondo en blanco

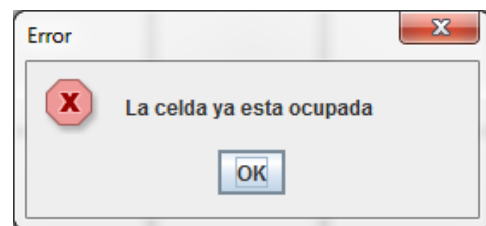


- e) **(4 pts.)** Cree la excepción CeldaOcupadaException

- f) **(6 pts.)** Cree el método **marcar**, el mismo que cuando la celda no esta utilizada la marca como usada por el jugador Usuario. Si la celda esta ocupada debe lanzar la excepción CeldaOcupadaException.

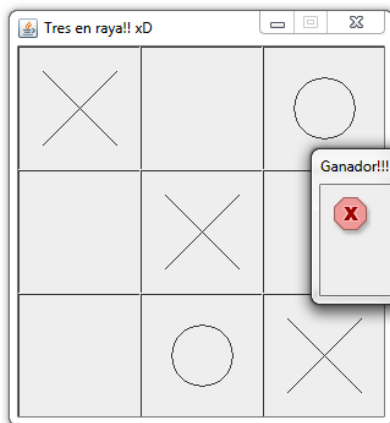
- g) **(8 pts.)** Cree una clase anónima que maneje el evento de dar clic sobre una Celda, la marque como usada y adicionalmente lance el evento "JugarComputador" mediante la instancia del objeto Observador que mantiene como referencia. Maneje la excepción CeldaOcupadaException y muestre un messageDialog indicando que la celda esta ocupada.
- ```
public void mouseClicked(MouseEvent me)
```

- h) **(5 pts.)** Sobre-escribir el método paintComponent para que en caso de que este usada por el usuario la celda se presente una "X", si esta ocupada por el computador una "O" y sino este vacía. No olvide que siempre debe aparecer el borde de la celda



### Clase Juego

- i) **(10 pts.)** Implemente el método notifyEvent(String eventName), el cual será invocado cada que el usuario le de clic a una celda para manejar el evento "JugarComputador" y debe hacer que el computador juegue. No olvide de validar si ya hay un ganador o no.



### Clase FrmGanador

- j) **(5 pts.)** Cree la ventana FrmGanador, la cual será invocada cuando el usuario gane.
- k) **(5 pts.)** Cree el método cargarRanking el cual pobla un HashMap con los pares <usuario, vecesGanador> a partir del archivo "Ranking.txt".
- l) **(5 pts.)** Cree el método guardarRanking el cual actualiza el contenido del archivo "Ranking.txt".
- m) **(5 pts.)** Cree el manejador del evento del botón el mismo que será el encargado de actualizar el archivo "Ranking.txt".

#### **TEMA 4. (10 puntos)**

Analice y explique que hace el siguiente código. Tenga en cuenta que la respuesta no es que imprime el código.



```
public class Test{
    public static void main(String []args){
        A objeto;
        objeto = A.getInstance();
        objeto = A.getInstance();
        objeto = A.getInstance();
    }
}
```

```
class A{
    private static A singleton;
    private A(){
        System.out.println("Instancia");
    }

    public static A getInstance(){
        if(singleton == null){
            singleton = new A();
        }
        return singleton;
    }
}
```

---

---

---

---

---