PAC DESARROLLO

CFGS Desarrollo de Aplicaciones Multiplataforma

Módulo 07:
Desarrollo de interfaces

INFORMACIÓN IMPORTANTE

Para la correcta realización de la PAC el alumno deberá consultar los contenidos recogidos en el material didáctico.

Requisitos que deben cumplirse en vuestros trabajos:

- Siempre que utilicéis información de Internet para responder / resolver alguna pregunta, tenéis que citar la fuente (la página web) de dónde habéis sacado esta información.
- No se aceptarán copias literales de Internet. Podéis utilizar Internet para localizar información, pero el redactado de las respuestas debe ser de elaboración propia.
- Es responsabilidad del alumno comprobar que el archivo subido en la plataforma es el correcto, ya que en ningún caso el profesor revisará el documento antes del periodo de corrección.
- Si no se entrega una PAC de desarrollo, la calificación equivaldrá a un 0.
- La PAC debe entregarse en **formato ZIP**.
- Este ZIP, contendrá el proyecto **realizado en Java**
- En el caso de **no** realizarse la entrega en dicho formato **el alumno se hace responsable** de posibles incompatibilidades en la visualización de su entrega y, por ende, afectará a su calificación.

CRITERIOS DE CORRECCIÓN

1. Todos los programas realizados en la PAC deben realizarse con IDE **NetBeans**
2. Para la realización de esta PAC es necesario que se utilicen las estructuras de control y las estructuras repetitivas siempre que sea posible.
3. Se deben poner comentarios para su mejor comprensión. Estos comentarios explicarán la funcionalidad del código. Se valorarán los comentarios en la parte de presentación.

INTRODUCCIÓN

En esta actividad se evaluarán vuestros conocimientos en programación orientada a objetos. Leed bien el enunciado ya que tenéis toda la información importante en relación con las clases y métodos que os encontrareis en el proyecto.

** Todas las clases, métodos y variables son obligatorios con el mismo nombre y parámetros, únicamente se tienen que modificar las instrucciones de los métodos.*

DESCRIPCIÓN

Ilerna Games es una empresa de diseño de videojuegos y ha contactado con nosotros.

Nos comentan que su programador se ha puesto de baja y necesitan acabar urgentemente un juego. Consiste en realizar un **tres en raya**, y por suerte, tenemos la parte de diseño y la estructura del juego montada. Únicamente faltan rellenar los métodos y la parte lógica del juego.

Podremos observar que está dividido en dos partes:

- La parte gráfica **Juego.java**
- La parte funcional **LogicaJuego.java**

Utilizando el programa **NetBeans** finalizaremos este juego, os encontraréis que tenéis los métodos detallados con las funcionalidades que se esperan comentados dentro del proyecto, de todas formas, se van a explicar a continuación:

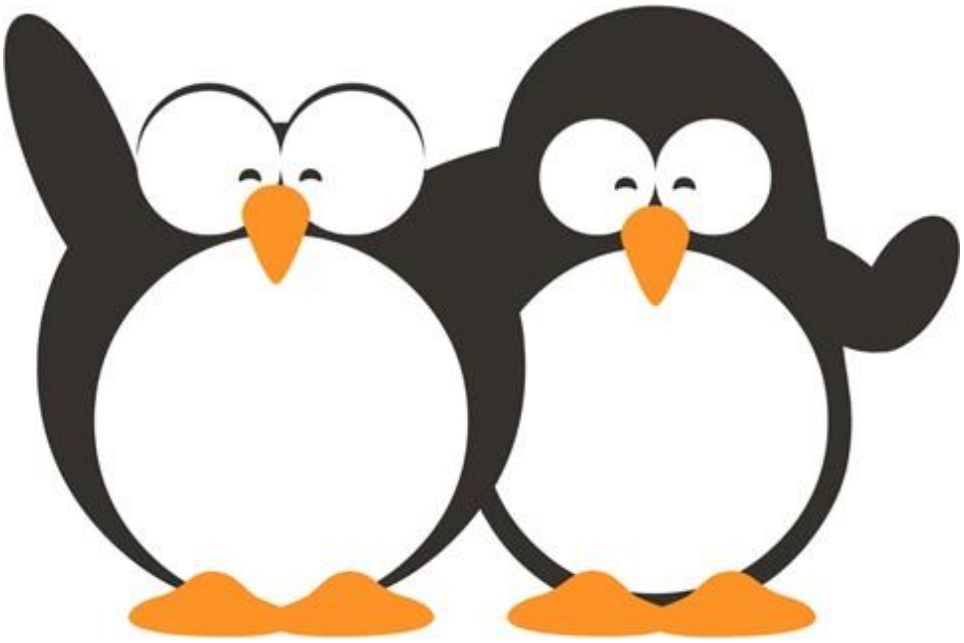
En la clase **Juego.java**

- Están declaradas de forma global la matriz y la clase LogicaJuego.
- *iniciarJuego()*: Es el método que inicializará una nueva partida, tendremos que llamar al método *iniciarPartida* de la clase *LogicaJuego.java* y vaciar los botones.
- *clearButtons()*: Vacía todos los botones para poder empezar una nueva partida.

Clase **LógicaJuego.java**

- De forma global están declaradas las variables turno, pX, pO
    - Turno: Turno de cada jugador (0 o 1)
    - pX: Turno de las “X”
    - pO: Turno de las “O”
- *getTurno()*: Método que nos permite obtener el turno actual.
- *cambioTurno():* Llamaremos a este método para cambiar de turno, es decir, si nos encontramos en el turno 1, que pase al turno 0 y viceversa.
- *comprobarJuego():* Es una de las partes más importantes del juego, ya que sin este método no gana nadie. Aquí comprobaremos si alguno de los jugadores ha conseguido 3 en raya, se tendrá que comprobar de forma ***horizontal, vertical y diagonal.***
- *pintarFicha():* Pinta la ficha en el botón de juego visual, si nos encontramos en el turno 0, se tendrá que poner un fondo al texto de color rojo **(Color.red)** y el texto **X**, por otro lado, el turno 1 se pintará de color azul **(Color.blue)** y se insertará el texto **O**.
- *ponerFicha():* Inserta la ficha en la matriz y llama al método adecuado para pintar la ficha en el botón.
- *tiradaJugador():* En este método, deshabilitaremos el botón seleccionado para evitar que se vuelva a tirar en esa misma casilla, insertaremos la ficha en la matriz llamando al método adecuado y comprobaremos el ganador. **Si existe ganador**, se llamará al método adecuado para que nos incremente la puntuación de cada jugador y deshabilitará el tablero en caso de haber ganador. **En caso contrario**, cambiará de turno.
- *ganador():* Método que actualizará la puntuación de cada jugador, y cambiará de turno para que la siguiente partida empiece el otro jugador.
- *habilitarTrablero():* Habilitará o deshabilitará **todos** los botones dependiendo de la variable ***habilitado*** *(setEnabled.(habilitado))*, es decir, todos los componentes del Jpanel que recibirá por parámetro.
- *iniciarPartida():* Inicializa una nueva partida, reinicia la matriz (Tablero de juego) y habilita el tablero.

INSTRUCCIONES DE ENTREGA

1. Se debe entregar el proyecto en formato .zip siguiendo la misma estructura, no es posible realizar modificaciones en los archivos.
2. El nombre del fichero .zip debe seguir la siguiente sintaxis: *Apellido_Nombre_UF1.zip*
3. Para que el ejercicio sea corregido, este debe entregarse obligatoriamente en la tarea. No se corregirán proyectos entregados por otros medios, como los adjuntos de los comentarios de la tarea o los mensajes privados.

¡Buen trabajo!