Lógica Tres En Raya

Clase MainActivity.java

Variables de clase:

- **Int [] Casillas**: Arreglo donde se guarda todos los ID de las casillas.

0	1	2	3	4	5	6	7	8	9

- **Int Jugadores**: Valor entero: 1 = se juega contra la computadora, 2 = se juega entre dos jugadores.
- **Partida partida**: Instancia creada de la clase Partida.

Clase Partida

Variable de clase:

- **Int dificultad:** Valor entero: 0 = fácil, 1 = normal, 2 = difícil.
- **Int jugador:** Valor entero: 1 = jugador uno 2 = jugador dos o la inteligencia artificial.
- **Int [] casillas:** Arreglo donde se almacena las posibles jugadas realizadas: 0 = esta libre para jugar, 1 = jugador uno hizo una jugada, 2 jugador o la maquina hizo una jugada.

0	1	2	3	4	5	6	7	8	9

- **Final int [] [] combinaciones:** Matriz donde se almacena todas las combinaciones para ganar el juego.

0	1	2
3	4	5
6	7	8
0	3	6
1	4	7
2	5	8
0	4	8
2	4	6

Representación del juego con las posiciones del Arreglo

0	1	2
3	4	5
7	8	9

Corrida N°1

- **Un jugador**
- **Fácil**

Clase MainActivity.java

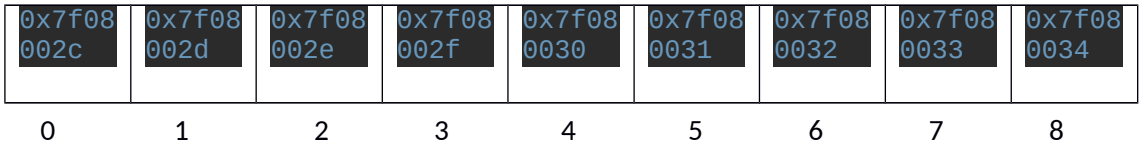
Ni bien se comienza a ejecutar la aplicación se crea las actividades, esto implica que se ejecuta internamente el método onCreate de la clase MainActivity.java

onCreate()

- Se crea el arreglo casillas
- Se llena el arreglo con los IDs en valor entero de las vistas de todas las casillas de nuestra interfaz.
  Para cada vista, Android genera un ID con un valor entero, tener en cuenta que estos IDs son únicos.

```
public static final int casillla1=0x7f08002c;
```

0x7f08002c: Esta en hexadecimal, transformado en decimal seria: 2131230764



Hacer clic en el botón Jugador 1

Se activa un evento, en este caso se activa onClick=aJugar del botón

Parámetros:

View vista = vista que hizo activar la acción aJugar()

aJugar()

- **Limpiar la pantalla. -** Se selecciona cada una de las casillas para cambiar su imagen por la casilla en blanco.

```
ImageView imagen;
for(int cadaCasilla:casillas)
{
    imagen=findViewById((cadaCasilla));
    imagen.setImageResource(R.drawable.casilla);
}
```

Se le pasa dos parámetros al for: una variable y un arreglo, esto significa que el for se ejecutara dependiendo del tamaño del arreglo.

Corrida del for: Se ejecutar 9 veces.

cadaCasilla = 0x7f08002c; corrida 1

cadaCasilla = 0x7f08002d; corrida 2

cadaCasilla = 0x7f08002e; corrida 3

cadaCasilla = 0x7f08002f; corrida 4

cadaCasilla = 0x7f080030; corrida 5

cadaCasilla = 0x7f080031; corrida 6

cadaCasilla = 0x7f080032; corrida 7

cadaCasilla = 0x7f080033; corrida 8

cadaCasilla = 0x7f080034; corrida 9

- **Definimos si se jugara contra la maquina o dos jugadores.**
  Suponemos que se está jugando con la máquina, caso contrario se está jugando entre dos personas.

```
jugadores=1;

if(vista.getId()==R.id.boton2jugadores)
{
    jugadores=2;
}
```

Para esta corrida, jugadores se mantiene en 1 significa que se hizo clic en el botón de "Un Jugador", porque el if no cumple.
Vista=boton1jugador
If(0x7f080024==0x7f080025) //No cumple

- **Definimos que dificultad fue selecciona.**
  Suponemos que este seleccionado fácil, sino es la opción normal o difícil.

```
RadioGroup configDificultad=findViewById(R.id.opcionDificultad);
int id=configDificultad.getCheckedRadioButtonId();
int dificultad=0;
if(id==R.id.botonNormal)
{
    dificultad=1;
}
else
{
    if(id==R.id.botonDificil)
    {
        dificultad=2;
    }
}
```

configDificultad = Objeto RadioButton
id = 0x7f080027
dificultad=0
if(0x7f080027==0x7f080028) //No cumple
else
{
        If(0x7f080027==0x7f080026) //No cumple
}

- **Creamos la clase Partida.**
  Para crear un objeto de la clase Partida necesitamos, un parámetro en este caso dificultad.

```
partida=new Partida(dificultad);
```

- Al crear un objeto siempre se ejecuta el constructor.

```
public Partida(int dificultad)
{
    this.dificultad=dificultad;
    jugador=1;
    casillas=new int[9];
    for(int i=0;i<9;i++)
    {
        casillas[i]=0;
    }
}
```

dificultad=0
jugador=1
casillas

0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9

- **Bloqueamos los botones y el radio button.**

```
(findViewById(R.id.boton1jugador)).setEnabled(false);
(findViewById(R.id.opcionDificultad)).setAlpha(0);
(findViewById(R.id.boton2jugadores)).setEnabled(false);
```

Hacer clic en cualquier casilla

Se activa un evento, en este caso se activa onClick=toque de la casilla que fue clikeada.

Parámetros:

View vista = vista que hizo activar la acción toque()

Toque()

- **Verificar si ya se presionó alguno de los botones para jugar (Un jugador o dos Jugadores).**
  Si la partida no fue creada, la acción toque no hace nada, es decir no podemos jugar.

```
if(partida==null)
{
    return;
}
```

- **Buscar que casilla fue presionado**

```
int casilla=0;

for(int i=0;i<9;i++)
{
    if(casillas[i]==vista.getId())
    {
        casilla=i;
        break;
    }
}
```

Para el ejemplo: Supongamos que se hizo click en la casilla de la posición 4.

casilla = 0;

Corrida del for: este for se ejecuta 9 veces, pero si la casilla seleccionada es encontrada break rompe el flujo del for.

If(0x7f08002c==0x7f080030) //No cumple

If(0x7f08002d ==0x7f080030) //No cumple

If(0x7f08002e ==0x7f080030) //No cumple

If(0x7f08002f ==0x7f080030) //No cumple

If(0x7f080030==0x7f080030) //Cumple

{

casilla=4

}

- **Verificamos si la casilla esta libre para hacer jugar con circulo o aspa**

Si cumple este if, significa que la casilla 4 no esta libre por lo tanto nos salimos del método.

```
if(partida.comprueba_casilla(casilla)==false)
{
  return;
}
```

Clase Partida: Se hace uso de nuestro objeto partida para ver si la casilla esta libre. True si la casilla esta libre, false si la casilla ya fue usada. Tambien nos ayuda a cambiar el estado de nuestro arreglo casillas.

```
public boolean comprueba_casilla(int casilla)
{
  if(casillas[casilla]!=0)
  {
    return false;//no se puede marcar
```

```
  }
  else
  {
    casillas[casilla]=jugador;
  }
  return true; //se puede marcar
}
```

casilla = 4;
if(0 != 0) //No cumple
else
{
  casilla[4]=1;
}
return true; //La respuesta es true

- **Realizar el cambio en la interfaz dependiendo que jugador realizo la acción.**

```
marcar(casilla);/*Cambia la imagen dependiendo que jugador
sea*/
```

```
private void marcar(int casilla)
{
  ImageView
imagen=(ImageView)findViewById(casillas[casilla]);
  if(partida.jugador==1)
  {
    imagen.setImageResource(R.drawable.circulo);
  }
  else
  {
    imagen.setImageResource(R.drawable.aspa);
  }
}
```

casilla=4
if(1==1) //Cumple
{
  cambiamos la imagen en blanco por una imagen con circulo.
}

- **Comprobamos si alguien gano o bien seguir jugando.**

```
int resultado=partida.turno();//0,1,2,3
if(resultado>0)
{
  termina(resultado);
  return;
}
```

```java
public int turno()//Clase Partida
{
  boolean empate=true;/*Suponemos que esta en empate hasta
comprobar lo contrario*/
  boolean ult_movimiento=true;/*true significa que el
jugador gano la partida*/
  for(int i=0;i<combinaciones.length;i++)
  {//ini primer for
    for (int pos:combinaciones[i])
    {//ini segundo for
      if(casillas[pos]!=jugador)
      {
        ult_movimiento=false;
      }
      if(casillas[pos]==0)
      {
        empate=false;//si hay una casilla libre
      }
    }//fin segundo for
    if(ult_movimiento)
    {
      return jugador;
    }
    ult_movimiento=true;
  }//fin primer for
  if(empate)
  {
    return 3;//Resulta empate
  }
  jugador++;
  if(jugador>2)
  {
    jugador=1;
  }
  return 0;
}
```

Resultado metodo turno(): 0 = juego continua, 1 = gano primer jugador, 2 = gano segundo jugador, 3 = empate

Corrida primer for: combinacion.length = cantidad de filas de la matriz

Corrida segundo for: combinaciones[i] = fila de la posición i de la matriz

for(i=0)
    for(pos=0: valores combinaciones[0]= 0 ,1,2)
    if(0 != 1) //casillas[0]=0, jugador=1
        ult_movimiento=false;
    if(0 == 0) //casillas[0]=0
        empate=false;

```
        for(pos=1: valores combinaciones[0]= 0 ,1,2) )

        if(0 != 1)          //casillas[1]=0, jugador=1
                ult_movimiento=false;
        if(0 == 0)          //casillas[1]=0
                empate=false;

        for(pos=2: valores combinaciones[0]= 0 ,1,2) )
        if(0 != 1)          //casillas[2]=0, jugador=1
                ult_movimiento=false;
        if(0 == 0)          //casillas[2]=0
                empate=false;
        Fin del segundo for

        if(false)           //No cumple     //ult_movimiento=false
        Si llegase a cumplir este if significa que gano el jugador 1
        ult_movimiento=true;

for(i=1)
        for(pos=3: valores combinaciones[1]= 3 ,4,5)
        if(0 != 1)          //casillas[3]=0, jugador=1
                ult_movimiento=false;
        if(0 == 0)          //casillas[3]=0
                empate=false;

        for(pos=4: valores combinaciones[1]= 3 ,4,5) )

        if(1 != 1)          //casillas[4]=1, jugador=1
                ult_movimiento=false;
        if(0 == 0)          //casillas[4]=0
                empate=false;

        for(pos=5: valores combinaciones[1]= 3 ,4,5) )
        if(0 != 1)          //casillas[5]=0, jugador=1
                ult_movimiento=false;
        if(0 == 0)          //casillas[5]=0
                empate=false;
        Fin del segundo for

        if(false)           //No cumple     //ult_movimiento=false

        ult_movimiento=true;
Esto continua hasta terminar la matriz

Fin del primer for

if(false)           //No cumple     //empate=false

jugador=2

if(2>2)             //No cumple

return 0            //Nuestra respuesta es 0, significa que le juego continua
```

Continuando con el método toque

resultado = 0;
if(0>0) //No cumple

Si resultado tuviera un valor de: 1, 2 o 3

resultado = 1;
if(1>0)
{
  termina(1); //Muestra el mensaje que el los círculos ganaron
  return;
}

- **Respuesta de la maquina (IA)**

```
if(jugadores==1)
{
 casilla=partida.ia();
 while(partida.comprueba_casilla(casilla)==false)
 {
  casilla=partida.ia();
 }
 marcar(casilla);
 resultado=partida.turno();
 if(resultado>0)
 {
  termina(resultado);
 }
}
```

El método ia() de la clase Partida genera un valor entre 0-8, dependiendo de la situación en la que se encuentre, es decir si esta en modo fácil, normal o difícil.

Antes de de hacer una corrida del método ia() analizaremos el método dosEnRaya(int jugador_en_turno). Los valores que nos puede retornar son:

-1= si no se encontró una opción de dos en raya
0-8 = un valor donde tenga la opción de hacer tres en raya.

DosEnRaya()

jugador_en_turno=2;
casilla = -1
cuantas_lleva = 0;

for(i=0)
for(pos=0: valores combinaciones[0]= 0 ,1,2)
if(0 == 2) //no cumple //casillas[0]=0

```
if(0 == 0)          //cumple          //casillas[0]=0
        casilla=0;

for(pos=1: valores combinaciones[0]= 0 ,1,2)

if(0 == 2)          //no cumple       //casillas[1]=0

if(0 == 0)          //cumple          //casillas[1]=0

        casilla=1;

for(pos=2: valores combinaciones[0]= 0 ,1,2)
if(0 == 2)          //no cumple       //casillas[2]=0

if(0 == 0)          //cumple          //casillas[2]=0
        casilla=2;
        Fin del segundo for

        if(0==2 && 2!=-1)       //No cumple

        casilla = -1
        cuantas_lleva=0

for(i=1)

for(pos=3: valores combinaciones[0]= 3,4,5)
if(0 == 2)          //no cumple       //casillas[3]=0

if(0 == 0)          //cumple          //casillas[3]=0
        casilla=3;

for(pos=4: valores combinaciones[0]= 3 ,4,5)

if(1 == 2)          //no cumple       //casillas[4]=0

if(0 == 0)          //cumple          //casillas[4]=0

        casilla=4;

for(pos=5: valores combinaciones[0]= 3 ,4,5)
if(0 == 2)          //no cumple       //casillas[5]=0

if(0 == 0)          //cumple          //casillas[5]=0
        casilla=5;
        Fin del segundo for

        if(0==2 && 5!=-1)       //No cumple

        casilla = -1
        cuantas_lleva=0

        Esto continua hasta terminar la matriz
```

Para nuestro caso no existe una posibilidad de dos en raya
Return -1

ia()

casilla = -1;
if(-1 != -1) //No cumple

if(0 > 0) //No cumple

if(0 == 2)

casilla = 4 //Este es generado de forma aleatoria entre 0 – 8
return 4 //Respuesta del método es ia() es 4