Universidad de Margarita

Alma Mater del Caribe

Decanato de Ingeniería

Cátedra: Programación II

Proyecto final

Profesor:

Cesar Requena

Realizado por:

Galvys Rodríguez C.I. 30.707.222

Agustin Gil C.I. 30.065.866

El Valle del Espíritu Santo, abril de 2023

¿Quién quiere ser millonario?

Se planteaba la situación de realizar un juego que haga alusión a ¿Quién quiere ser millonario?, que sea referente al arte, por lo que se tomaron los distintos requerimientos con un instructivo de cómo se juega, que el usuario se le permite ingresar su nombre, donde la serie de preguntas son 15 con 4 posibles respuestas, donde tendrá que contestar correctamente para seguir avanzando como máximo donde las recompensas irán desde 100$ hasta 1000000$, el participante contara con dos comodines el de 50:50 para eliminar dos opciones que no sean correctas, el otro comodín será la posibilidad de saltarse una pregunta.

Cada ronda tiene una cantidad de dinero a ganar por contestar correctamente la pregunta, por lo que en esta modalidad de juego tendrá como base la pregunta número 5 y 10, de modo tal que si pierde, pero ya avanzo estas preguntas llevarse el premio garantizado para estas bases, y en el caso de rendirse si te llevas el dinero correspondiente a la cantidad de preguntas contestadas lleva el participante.

Luego la partida de la persona al contestar correctamente las 15 partidas llega será el final y ganara, por en cambio sí contesta de manera incorrecta o decide rendirse, se acabara la partida y la cual quedara registrada en archivos de texto, igualmente dentro del programa podrá visualizar la tabla de clasificación de las mejores participaciones dentro del ¿Quién quiere ser millonario?.

Por lo que ante esta situación se decidió plantear 3 clases dentro las cuales serían, el usuario que está jugando la partida, las preguntas y la interfaz. Las preguntas totales serán un arreglo de un arreglo de 3x10 las dimensiones, para realizar la asignación de las preguntas de manera aleatoria se designan 10 preguntas por cada nivel o base que serían de las preguntas del 1 a 5, 6 a 10 y 11 a 15, por eso el tamaño de este arreglo que se llena por medio de un archivo de texto, al igual que el programa cuenta con un dialogo para obtener el nombre del usuario dentro de la partida.

La clase usuario contiene de atributos nombre, ronda, puntajeMaximo, cincuentaCincuenta, cambiarPregunta que es lo necesario para llevar a cabo su partida, con su método contructor, asi como para obtener y cambiar valores de cada uno de estos atributos,

adicionalmente de los métodos de avanzarRonda que es en caso del atributo ronda en vez de cambiar su valor se va añadiendo, reiniciarRonda e iniciarPartida para poder volver a jugar dentro del juego.

La clase preguna tiene de atributos textoPregunta, opcionCorrecta, imagen y un arreglo de opciones para almacenar las 4 opciones posibles dentro de la pregunta, tiene su método constructor, además de contar con sus métodos para cambiar y obtener los valores, que en el caso del arreglo de opciones se realiza solo una obtención de un valor accediendo con el parámetro que es usado como índice.

La clase interfaz, es la interfaz gráfica de este programa, en la cual tiene de atributos el userActual, la preguntaActual, el arreglo de un arreglo de las preguntas de tamaño 3x10, la tablaPosiciones que es un arreglo de 10 de cadena, tiene métodos relacionados con la parte visual como obtenerImagen, formatearCampos, modificarFormatoTexto, obtenerLayout, actualizarPuntuacion, actualizarPuntuacionAux, por otro lado tiene en la parte lógica del programa crearDirectorios, posicionEnArreglo, leerPregunta, llenarArregloPreguntas, partidaNueva, escogerPreguntaAleatoria, actualizarPregunta, puntuación, actualizarPuntuacion, cincuentaCincuenta, intercambio, ordenamiento, mejoresPuntuaciones, mostrarTabla, historialPartida, bases, respuestaPregunta, darResultados, llenarLabelsNuevaPregunta.

PseudoCódigo

```
clase Usuario
        nombre: cadena
        ronda,puntajeMaximo: entero
        cincuentaCicuenta, cambiarPregunta: boolean

        metodo Usuario(nom:cadena)
                nombre = nom
                ronda = 1
                puntajeMaximo = 0
                cincuentaCincuenta = verdadero
                cambiarPregunta = verdadero
        Fin metodo

        metodo getNombre()
                retornar nombre
        Fin metodo


        metodo getRonda()
                retornar ronda
```

```
Fin metodo

metodo getPuntajeMaimo()
        retornar puntajeMaximo
Fin metodo

metodo getValorCincuentaCincuenta()
        retornar cincuentaCincuenta
Fin metodo

metodo setNombre(nom:cadena)
        nombre = nom
Fin metodo

metodo setPuntajeMaximo(punt:entero)
        puntajeMaximo = punt
Fin metodo

metodo setValorCincuentaCincuenta(val:boolean)
        cincuentaCincuenta = val
Fin metodo

metodo avanzarRonda()
```

```
        ronda += 1
    Fin metodo

    metodo reiniciarRonda()
        ronda = 0
    Fin metodo

    metodo getCambiarPregunta()
        retornar cambiarPregunta
    Fin metodo

    metodo setCambiarPregunta(cambiarPregunta:boolean)
        cambiarPregunta = cambiarPregunta
    Fin metodo

    metodo iniciarPartida()
    cincuentaCincuenta=verdadero
    cambiarPregunta = verdadero
    puntajeMaximo = 0
    Fin metodo

Fin clase
```

```
clase Pregunta
        textoPregunta, imagen :cadena
        opciones: arreglo de cadena[4]
        opcionCorrecta: entero

        metodo Pregunta(pregunta:cadena, opc1:cadena,
opc2:cadena,opc3:cadena,opc4:cadena, opcC: entero, imagen:cadena)
                textoPregunta = pregunta
                opciones[0] = opc1
                opciones[1] = opc2
                opciones[2] = opc3
                opciones[3] = opc4
                opcionCorrecta = opcC
                imagen = img
        Fin metodo

        metodo getTextoPregunta()
                retornar textoPregunta
        Fin metodo

        metodo getOpcionCorrecta()
                retornar opcionCorrecta
```

```
    Fin metodo

    metodo getImagen()
        retornar Imagen
    Fin metodo

    metodo setTextoPregunta(pregunta:cadena)
        textoPregunta = pregunta
    Fin metodo

    metodo setOpcion(n:entero, opcion: cadena)
        opciones[n] = opcion
    Fin metodo

    metodo setOpcionCorrecta(opcionC:entero)
        opcionCorrecta = opcionC

    metodo setImagen(imagen:cadena)
        imagen = img
    Fin metodo

Fin clase
```

```
clase Interfaz
        BARRA, UBICACION, nombreUsuario: cadena
        preguntaActual: Pregunta
        userActual: Usuario
        tablaPosiciones: arreglo de cadena[10]
        arregloPreguntas: arreglo de arreglo de Pregunta[3][10]
        BARRA  = separadorDeArchivo
        UBICACION = ubicacionActual + BARRA + "src" + BARRA + "archivos" + BARRA

        metodo crearDirectorios()
                fichero: Archivo
                fichero = nuevo Archivo(UBICACION)
        Fin metodo

        metodo posicionEnArreglo(pregunta:Pregunta)
                posicion: Punto
                i, j: entero
        posicion = nuevo Punto(0, 0);

        para i=0 hasta 2 hacer
                        para j= 0 hasta 9 hacer
                        si(arregloPreguntas[i][j] == pregunta)entonces
```

```
                    posicion.moverse(i, j)
                    Fin si
                Fin para
        Fin para

        retornar Posicion
        Fin metodo

        metodo leerPregunta(numPregunta: entero)
            lector: lectorArchivo
            nuevaPregunta: Pregunta
            busquedaPregunta, txtPregunta1, opc1, opc2, opc3, opc4, opcC, img : cadena
            busquedaPregunta = ""
            lector = nuevo lectorArchivo(UBICACION + "preguntas.txt")
            Mientras(!busquedaPregunta.contenga("P"+ numPregunta) hacer
                busquedaPregunta = lector.leerSiguienteLinea()
            Fin mientras
            txtPrgunta = lector.leerSiguientePregunta()
            opc1 = lector.leerSiguienteLinea()
            opc2 = lector.leerSiguienteLinea()
            opc3 = lector.leerSiguienteLinea()
            opc4 = lector.leerSiguienteLinea()
```

```
            opcC = lector.leerSiguienteLinea()
            img = lector.leerSiguienteLinea()

            nuevaPregunta = nuevo Pregunta(txtPregunta, opc1, opc2, opc3, opc4, opcC,
img)
            lector.cerrar()
            retornar nuevaPregunta
        Fin metodo

        metodo llenarArregloPreguntas()
            numPregunta, i, k: entero
            numPregunta = 1
            para j = 0 hasta 2 hacer
                para k = 0 hasta 9 hacer
                    arregloPreguntas[j][k] = leerPregunta(numPregunta)
                    numPregunta = numPregunta + 1
                Fin para
            Fin para
        Fin metodo

        metodo partidaNueva(user:Usuario)
            llenarArregloPreguntas()
          actualizarPregunta(user)
```

```
        user.iniciarPartida()
        user.reiniciarRonda()
        actualizarPuntuacion()
Fin metodo

metodo escogerPreguntaAleatoria(user:Usuario)
        seleccion : entero
        seleccion = numeroAleatorio(0,10)
        Mientras(arregloPreguntas[nivel][seleccion] == null) hacer
                seleccion = numeroAleatorio(0,10)
        Fin mientras
        retornar arregloPreguntas[nivel][seleccion]
Fin metodo

metodo actualizarPregunta(user:Usuario)
        si(user.getRonda() < 6) entonces
                preguntasAcutal = escogerPreguntaAleatoria(0)
        sino si(user.getROnda() < 11)
                preguntasAcutal = escogerPreguntaAleatoria(1)
        sino
                preguntasAcutal = escogerPreguntaAleatoria(2)
        Fin si
        llenarLabelsNuevaPregunta(preguntaActual)
```

```
Fin metodo


metodo puntuacion()
        i, aux: entero
        i =  userActual.getRonda()
        aux = 0
      Segun (i)
        caso 1:
           aux = 100
        caso 2:
           aux = 200
        caso 3:
           aux = 300
        caso 4:
           aux = 500
        caso 5:
           aux = 1000
        caso 6:
           aux = 2000
        caso 7:
           aux = 4000
        caso 8:
           aux = 6000
```

```
        caso 9:
            aux = 8000
        case 10:
            aux = 10000
        case 11:
            aux = 15000
        case 12:
            aux = 30000
        case 13:
            aux = 100000
        case 14:
            aux = 300000
        case 15:
            aux = 1000000
        retornar aux
Fin metodo

metodo intercambio(a:entero, b:entero)
        aux: cadena
        aux = tablaPosiciones[a]
        tablaPosiciones[a] = tablaPosiciones[b]
        tablaPosiciones[b] = aux
Fin metodo
```

```
metodo cincuentaCincuenta()
        valor, eliminar1, eliminar2 : entero
        valor = preguntaAcutal.getOpcionCorrecta()
        eliminar1 = 1
        eliminar2 = 3
        hacer
                eliminar1 = numeroAleatorio(1,5)
        mientras(eliminar1 == valor)
        Fin mientras

        hacer
                eliminar2 = numeroAleatorio(1,5)
        mientras(eliminar2 == eliminar1 || eliminar2 == valor)
        Fin  mientras

Fin metodo

metodo ordenamiento(max: entero)
        puntuaciones, rondas : arreglo de enteros[10]
        aux : arreglo de cadena[3]
        i, j : entero
        para i = 0 hasta max hacer
```

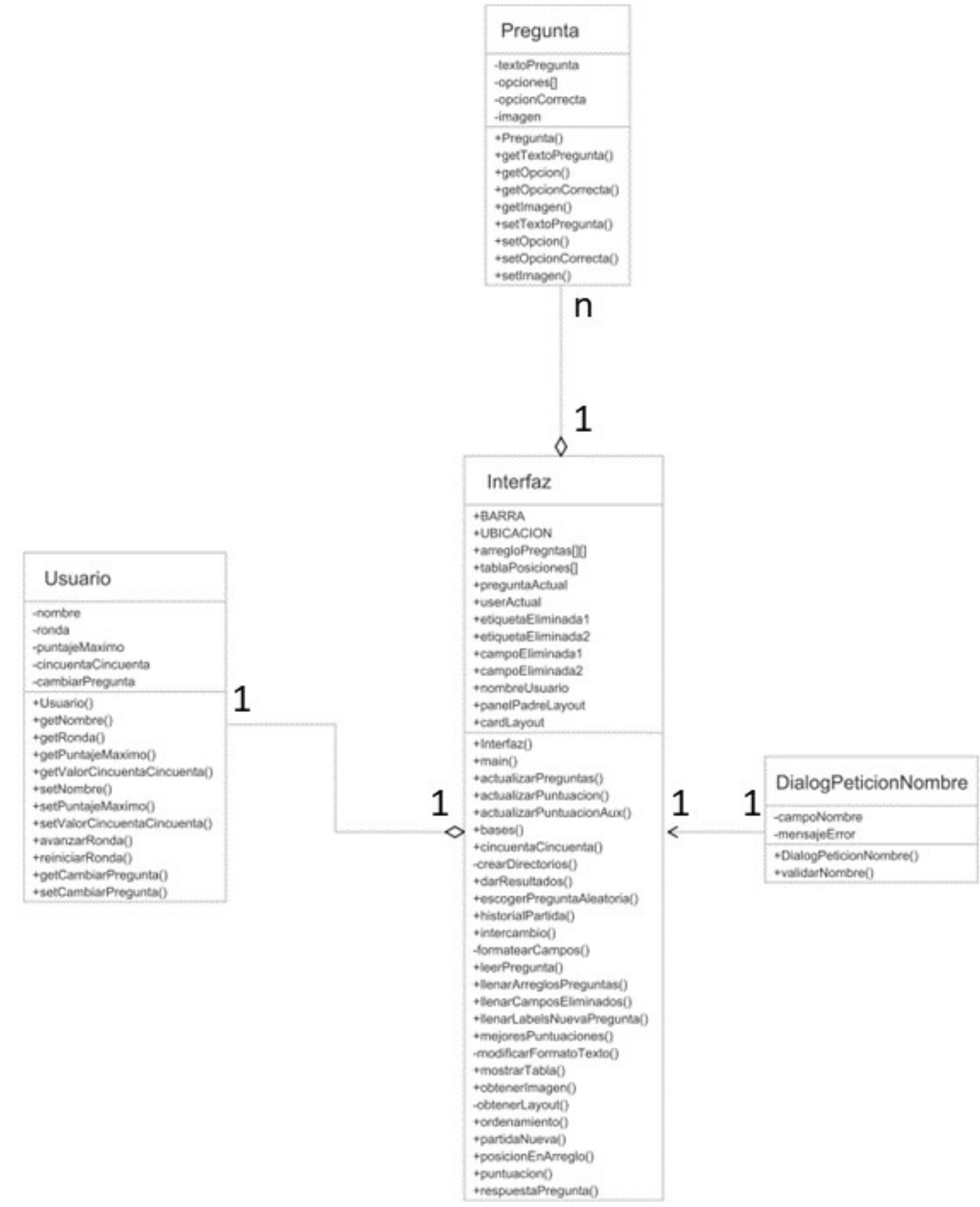
```
                    para j = 0 hasta max-1 hacer
                        aux = tablaPosiciones[j].separarPor(:)
                        puntuaciones[j] = aux[3]
                        rondas[j] = aux[2].Subcadena(0, aux[2].indiceDe(" Premio"))
                        si(puntuaciones[j] > puntuaciones[j+1]) entonces
                            intercambio(j, j + 1)
                        sino si((puntuaciones[j] == puntuaciones[j + 1] && rondas[j]
>= rondas[j + 1])
                            intercambio(j, j + 1)
                        Fin si
                    Fin para
                Fin para
        Fin metodo



        metodo mejoresPuntuaciones(linea:cadena)
            aux, aux2: arreglo de cadena[3]
            valor, puntaje, rondaActual, rondaPrevia: entero
        aux = tablaPosiciones[0].separarPor(":")
        aux2 = linea.separarPor(":")
        valor = aux2[3]
        puntaje = aux[3]
        rondaActual = aux2[2].subCadena(0, aux[2].indiceDe(" Premio"))
```

```
    rondaPrevia =  aux2[2].subCadena(0, aux[2].indiceDe(" Premio"))
    si(valor > puntaje) entonces
                    tablaPosiciones[0] = linea
                    ordenamiento(10);
    sino si (valor == puntaje && rondaActual > rondaPrevia)
                    tablaPosiciones[0] = linea
                    ordenamiento(10)
    Fin si
Fin metodo

    metodo mostrarTabla()
            archivo: Archivo
            ubicacion, linea, texto1, texto2, texto3 : cadena
            lector: escaner
            aux: arreglo de cadena[3]
            i, k: entero
    ubicacion = UBICACION + "archivos" + BARRA + "resultados.txt"
    archivo = nuevo Archivo(ubicacion)

    lector = nuevo escaner(archivo)
            i = 0
            mientras(lector.tieneSiguienteLinea())hacer
            linea = lector.SiguienteLinea()
```

```
        si (i < 10) entonces
                tablaPosiciones[i] = linea
                ordenamiento(i+1)
        sino
                mejoresPuntuaciones(linea)
                Fin si
                i = i + 1
        Fin mientras
        lector.cerrar()
        para k = i hasta 0 hacer
        aux = tablaPosiciones[k].separarPor(":")
        texto1 += texto1 + aux[1].subCadena(0, aux[1].indiceDe(" Ronda")) + "/n"
        texto2 += texto2 + aux[2].subCadena(0, aux[2].indiceDe(" Premio")) + "/n"
        texto3 += texto3 + aux[3].trim() + "<br>"
        Fin para
Fin metodo

metodo historialPartida()
        ubicacion : cadena
        archivoNuevo : Archivo
        escritor: escribirArchivo
        ubicacion = UBICACION + "archivos" + BARRA + "resultados.txt"
        archivoNuevo = nuevo Archivo(ubicacion)
```

```
            escritor = nuevo escribirArchivo(archivoNuevo)
            escritor.escrbirSiguienteLinea("Nombre: " + userActual.getNombre() + "
Ronda:" + userActual.getRonda() + " Premio: " + userActual.getPuntajeMaximo())
            escritor.cerrar()
        Fin metodo

        metodo bases()
        si(userActual.getRonda() > 5) entonces
                    userActual.setPuntajeMaximo(1000)
        sino si(userActual.getRonda() > 10)
                    userActual.setPuntajeMaximo(10000)
            sino
                    userActual.setPuntajeMaximo(0)
        Fin si
        Fin metodo

        metodo respuestaPregunta(preguntaActual:Pregunta, eleccion:entero, user:Usuario)
            i, j : entero
        si (preguntaActual.getOpcionCorrecta() == eleccion) entonces
                    i = posicionEnArreglo(preguntaActual).x
                    j = posicionEnArreglo(preguntaActual).y
                    user.avanzarRonda()
                    actualizarPuntuacion()
```

```
                si (user.getRonda() == 15) entonces
                user.setPuntajeMaximo(1000000)
                historialPartida()
                darResultados()
                sino
                actualizarPregunta(user)
                arregloPreguntas[i][j] = null
                Fin si
            sino
                bases()
                historialPartida()
                darResultados()
                user.reiniciarRonda()
            Fin si
        Fin metodo
Fin clase
```


Pregunta
-textoPregunta
-opciones[]
-opcionCorrecta
-imagen
+Pregunta()
+getTextoPregunta()
+getOpcion()
+getOpcionCorrecta()
+getImagen()
+setTextoPregunta()
+setOpcion()
+setOpcionCorrecta()
+setImagen()
n
1
Interfaz
+BARRA
+UBICACION
+arregloPregntas[][]
+tablaPosiciones[]
+preguntaActual
+userActual
+etiquetaEliminada1
+etiquetaEliminada2
+campoEliminada1
+campoEliminada2
+nombreUsuario
+panelPadreLayout
+cardLayout
+Interfaz()
+main()
+actualizarPreguntas()
+actualizarPuntuacion()
+actualizarPuntuacionAux()
+bases()
+cincuentaCincuenta()
-crearDirectorios()
+darResultados()
+escogerPreguntaAleatoria()
+historialPartida()
+intercambio()
-formatearCampos()
+leerPregunta()
+llenarArreglosPreguntas()
+llenarCamposEliminados()
+llenarLabelsNuevaPregunta()
+mejoresPuntuaciones()
-modificarFormatoTexto()
+mostrarTabla()
+obtenerImagen()
-obtenerLayout()
+ordenamiento()
+partidaNueva()
+posicionEnArreglo()
+puntuacion()
+respuestaPregunta()
Usuario
-nombre
-ronda
-puntajeMaximo
-cincuentaCincuenta
-cambiarPregunta
+Usuario()
+getNombre()
+getRonda()
+getPuntajeMaximo()
+getValorCincuentaCincuenta()
+setNombre()
+setPuntajeMaximo()
+setValorCincuentaCincuenta()
+avanzarRonda()
+reiniciarRonda()
+getCambiarPregunta()
+setCambiarPregunta()
1
1
1
1
DialogPeticionNombre
-campoNombre
-mensajeError
+DialogPeticionNombre()
+validarNombre()