**Manual Del Administrador**

**Ingeniería de Sistemas**

**2015**

**Nombre :** Jhon Frayser Guizado Gonzales

**Docente :** Ing. Ivan Soria Solis

**Asignatura:** Ingeniería de Software III

**Introducción**

En este documento mencionaremos las herramientas necesarias para el desarrollo del juego de Juego de Ajedrez, se indicara el lenguaje de programación utilizada y el desarrollo general del proyecto.

**Herramientas Utilizadas**

Para la creación del juego se utilizó el entorno de desarrollo integrado libre NetBeans IDE 7.2 hecho principalmente para el lenguaje de programación Java.

**Prerrequisitos para la instalación de NetBeans**

Debe tener instalado Java Standard Edition, conocido como JDK (Java Developer Kit). Lo puede descargar de <http://www.oracle.com/technetwork/java/javase/downloads/>.



**2**

**1**

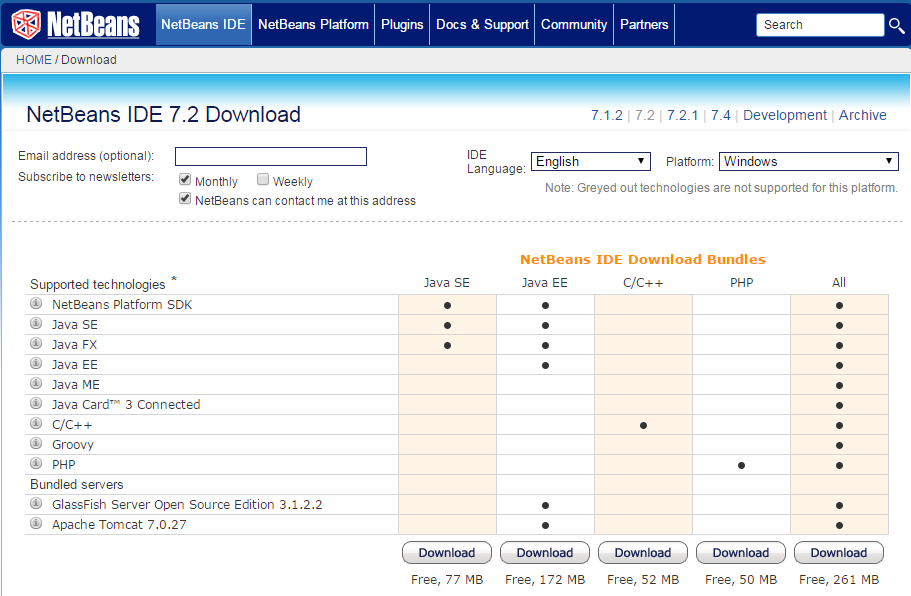
Como se muestra en la figura lo puedes descargar de 2 maneras:

1: la primera opción es que descargues el *jdk* integrado ya con NetBeans.

2: La segunda opción es que lo descargues solamente el  *jdk* sin el entorno de desarrollo integrado libre NetBeans.

**Instalación de NetBeans**

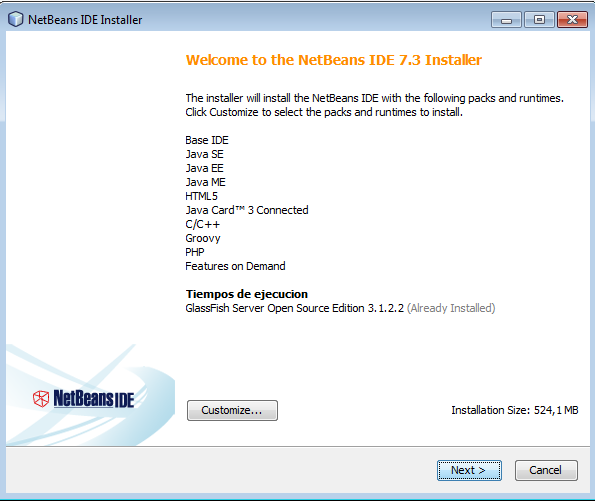
Para poder instalar el programa necesitamos descargo para eso nos vamos a su página <https://netbeans.org/downloads/7.2/> y descargarlo desde ahí.

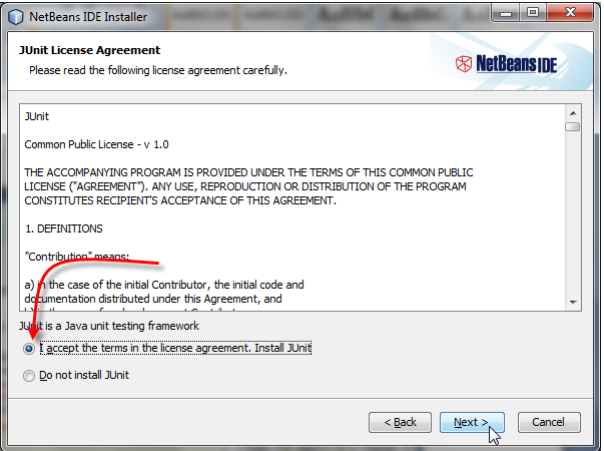


Como vemos en la imagen tenemos varias opciones, se puede elegir el lenguaje o idioma en el que quieres el programa, se elige para que plataforma es el programa si es para sistema operativo Linux, Mac OS X, Windows.

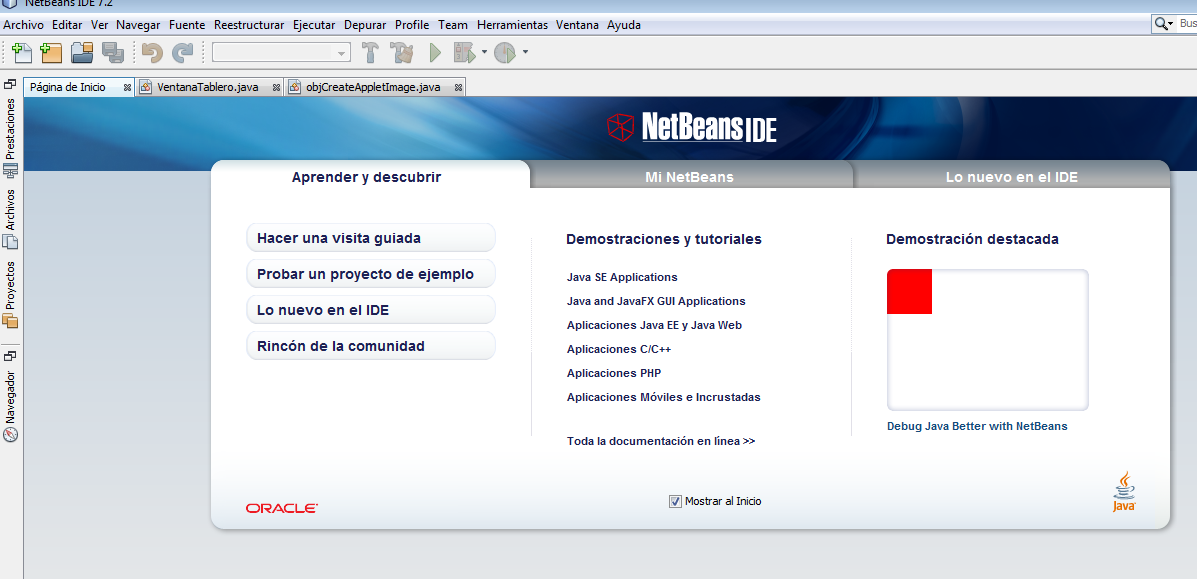
En la parte inferior tenemos varias opciones de descarga según los paquetes que queremos que tenga nuestro programa en este caso elegimos todo y pasamos a la instalación.

En la siguiente imagen que se muestra aparece la venta de instalación solo damos clic en Next, luego aceptamos, de ahí para adelante es todo Next.



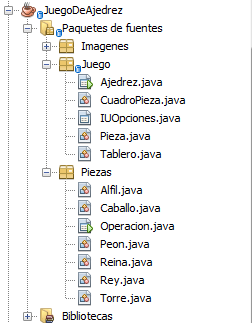


Luego, puede apreciar la ventana de bienvenida de NetBeans que tiene una serie de links para visitar documentación, proyectos demostrativos y otros, como se muestra a continuación:



Cuando decida desinstalar NetBeans, debe hacerlo a través del Panel de Control, seleccionando la desinstalación de Glassfish y Tomcat en el diálogo desinstalar de NetBeans.

**Desarrollo del Juego de Ajedrez**

**Paquete Imágenes:** en este paquete se encuentra todas las imágenes o piezas importadas, que serán utilizados en este juego de estrategia.

**Paquete Juego:** es este paquete se encuentra programación principal, además aquí se encuentra la ventana principal donde mostrara el tablero.

**Paquete de Piezas:** en este paquete se encuentran la programación de cada pieza o ficha del juego, contienen métodos de movimientos.

Figura 1: Paquete de juego

En el paquete juego el archivo Ajedrez.java es la principal clase de entorno grafico del juego

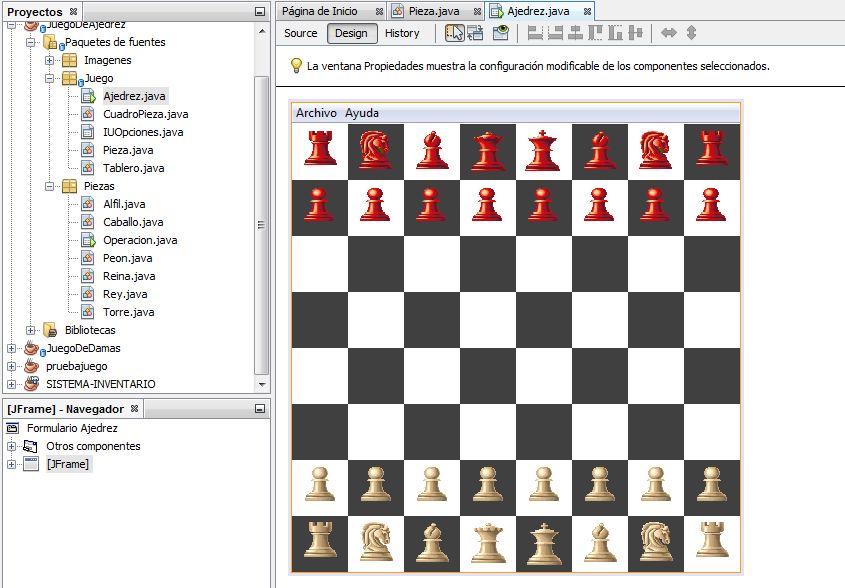


Figura : Ajedrez.java

**PIEZA.JAVA:** En esta clase abstracta establecerá las funciones básicas de toda pieza, será la clase padre, de las demás piezas.

package Juego;

import javax.swing.ImageIcon;

\* Se declara abstracta para que no pueda ser instanciada.

\* @author JhonFrayser

\*/

public abstract class Pieza {

/\*\*

\* Es una variable static o variable de clase, establece cuantos mavimientos hay actualmente sin realizar algun cambio(comer a otras piezas)

\* Esto es para cumplir la regla que dice que no pueden pasar mas de 50 turnos sin comer alguna pieza del oponente.

\*/

private static int cantMovimientosSinCambios = 0;

/\*\*

\* @return the cantMovimientosSinCambios

\*/

public static int getCantMovimientosSinCambios() {

return cantMovimientosSinCambios;

}

/\*\*

\* @param aCantMovimientosSinCambios the cantMovimientosSinCambios to set

\*/

public static void setCantMovimientosSinCambios(int aCantMovimientosSinCambios) {

cantMovimientosSinCambios = aCantMovimientosSinCambios;

}

/\*\*

\* Establece si la pieza ya se ha movido alguna vez, dado que algunas piezas solo pueden realizar determinados movimientos en su primer movimiento

\* Como es el peon que puede moverse dos casillas hacia adelante en su primer movimiento.

\* O el rey que solo puede hacer enroque en su primer movimiento

\*/

private boolean firstmov = true;

/\*\*

\* Establece el peso de la pieza, esto quiere decir cuanto vale una pieza, los valores de las piezas serán establecidos por las clases que hereden.

\* Los valores son los siquientes:

\* Rey: -- No tiene peso es el mas importante, le estableceremos un peso 10

\* Reina: 8

\* Torre: 5

\* Caballo: 3

\* Alfil: 3

\* Peon: 1

\*/

private int peso;

/\*\*

\* Establece el color de la pieza

\* Blanco: 1

\* Negro: -1

\*/

private int color;

/\*\*

\* Es la imagen de la pieza, que será declarada en las clases hijas.

\*/

private ImageIcon imagenPieza;

/\*\*

\* Es el cuadro del tablero en donde actualmente esta la pieza.

\*/

private CuadroPieza cuadroPieza = null;

/\*\*

\* Constructor.

\* @param color

\* Blanco: 1

\* Negro: -1

\*/

public Pieza(int color) {

this.color = color;

}

/\*\*

\* Este metodo valida el movimiento, solo valida, NO Mueve las piezas

\* Este metodo es un metodo general, que todas las piezas van a cumplir, Ademas cada pieza tiene otras validaciones

\* Las clases piezas que hereden pueden sobreescribir el metodo y reutilizarlo a la vez.

\* @param Destino

\* @param tbl

\* @return boolean TRUE si es un movimiento valido

\*/

////////////////////////////////////////////////////////////////////

public boolean validarMovimiento(CuadroPieza Destino, Tablero tbl) {

if (Destino.getPieza() != null) {//Si el cuadro destino esta ocupado entonces:

if (Destino.getPieza().getColor() == getCuadroPieza().getPieza().getColor()) {

//Si la pieza destino tiene el mismo color que la pieza que voy a mover

return false; //El movimiento es invalido, no puedo comer una pieza de mi mismo equipo.

}

}

//---Valido que no su rey no entre en jacke despues de que se mueva.

Pieza tmpDestino = Destino.getPieza();//Guardo las piezas en temporales, para deshacer los cambios

Pieza tmpActual = getCuadroPieza().getPieza();//Guardo las piezas en temporales, para deshacer los cambios

CuadroPieza Actual = getCuadroPieza();//El cuadro dode actualmente esta la pieza, tambien lo guardo.

Actual.setPieza(null);//simulo un movimiento, dejo el cuadro actual en vacio

Destino.setPieza(this);//Muevo esta pieza al cuadro destino

//Valido si despues de que mueva, el rey esta en jacke.

boolean b = tbl.getRey(getColor()).isInJacke(tbl);//Para lo cual obtengo el rey del mismo color de la pieza y le pregunto si esta en jacke

Actual.setPieza(tmpActual);//Deshago los cambios, porque SOLO valido, NO muevo piezas

Destino.setPieza(tmpDestino);//Deshago los cambios, porque solo valido, NO muevo piezas

/\*

\* Si retorna que esta en jacke(b=true), entonces el movimiento es invalido,

\* Si retorna que no esta en jacke(b=false), el mov. es valido.

\*/

return !b;

}

/////////////////////////////////////////////

/\*\*

\* Este metodo mueve la pieza a un cuadro destino

\* @param Destino

\* @param tbl

\* @return boolean FALSE si no logra mover la pieza.

\*/

//////////////////////////////////////////////////////////////////////////////

public boolean MoverPieza(CuadroPieza Destino, Tablero tbl) {

/\*

\* Valido el movimiento, antes de mover, tener en cuenta que en las clases hijas este metodo debe haber sido sobreescrito

\* Por lo que no solo va a validar lo que hay en el metodo validarMovimiento de Pieza, si no va a usar el metodo sobreescrito en la clase hija

\*/

if (validarMovimiento(Destino, tbl)) {

getCuadroPieza().setPieza(null);//Le paso al cuadro donde actualmente esta la pieza el valor de null, que quiere decir que ya no tiene pieza

if (Destino.getPieza() != null) {//Si hay una pieza en el destino

tbl.getPiezasComidas().add(Destino.getPieza());//Agrego la pieza que estoy comiento a un arraylist de piezas comidas.

setCantMovimientosSinCambios(0);//Si come a alguna pieza, reseteo el contador de movimientos sin cambios.

} else {

setCantMovimientosSinCambios(getCantMovimientosSinCambios() + 1);//Si no come alguna pieza, el contador aumenta en uno.

}

Destino.setPieza(this);//Muevo la pieza al cuadro destino

setFirstmov(false);//El siguiente movimiento, ya no sería el primero.

return true;

} else {

return false;

}

}

///////////////////////////////////////////////////////////////////

/\*\*

\* Si ya no existe movimientos posibles. el jugador esta ahogado.

\* @param turno

\* @param tbl

\* @return

\*/

////////////////////////////////////////////////////////////////////////////////////////

public static boolean isJugadorAhogado(int turno, Tablero tbl) {

CuadroPieza cuadroDestino = null;

CuadroPieza cuadroActual = null;

/\*

\* En sintesis, lo que se hace aqui es recorrer todas las piezas, y comparar si tienen algun movimiento posible, a alguna posicion del tablero.

\* Para recorrer todas las piezas uso los dos primeros for.

\* Para recorrer los cuadros posibles a los que se puede mover, uso los 2 siguientes for.

\*/

for (int x = 0; x < 8; x++) {//Busco todas las piezas del jugador del que quiero saber si esta ahogado.

for (int y = 0; y < 8; y++) {

cuadroActual = tbl.getTablero()[y][x];

if (cuadroActual.getPieza() != null) {//Si hay una pieza en el cuadro selecciondo actualmente por el for.

if (cuadroActual.getPieza().getColor() == turno) {//Reviso que sea del color de pieza, que del jugador que quiero averiguar si esta ahogado.

for (int x1 = 0; x1 < 8; x1++) {//Recorro todos los cuadros a del tablero preguntandole si puede moverse a ese cuadro

for (int y1 = 0; y1 < 8; y1++) {

cuadroDestino = tbl.getTablero()[y1][x1];

if (cuadroActual.getPieza().validarMovimiento(cuadroDestino, tbl)) {//Si hay un movimiento posible, entonces no esta ahogado

return false;

}

}

}

}

}

}

}

return true;//Si no encontro algun movimiento posible, pues el usuario esta ahogado, y retorno true.

}

////////////////////////////////////////////////////////////////////////////////

/\*\*

\*

\* @return color de la pieza.

\* Blanco: 1

\* Negro: -1

\*/

public int getColor() {

return color;

}

/\*\*

\*

\* @param color

\* Blanco: 1

\* Negro: -1

\*/

public void setColor(int color) {

this.color = color;

}

/\*\*

\* @return the firstmov

\*/

public boolean isFirstmov() {

return firstmov;

}

/\*\*

\* @param firstmov the firstmov to set

\*/

public void setFirstmov(boolean firstmov) {

this.firstmov = firstmov;

}

/\*\*

\* @return the peso

\*/

public int getPeso() {

return peso;

}

/\*\*

\* @param peso the peso to set

\*/

public void setPeso(int peso) {

this.peso = peso;

}

/\*\*

\* @return the imagenPieza

\*/

public ImageIcon getImagenPieza() {

return imagenPieza;

}

/\*\*

\* @param imagenPieza the imagenPieza to set

\*/

public void setImagenPieza(ImageIcon imagenPieza) {

this.imagenPieza = imagenPieza;

}

/\*\*

\* @return the cuadroPieza

\*/

public CuadroPieza getCuadroPieza() {

return cuadroPieza;

}

/\*\*

\* @param cuadroPieza the cuadroPieza to set

\*/

public void setCuadroPieza(CuadroPieza cuadroPieza) {

this.cuadroPieza = cuadroPieza;

}

}

**TABLERO.JAVA:** Esta clase es la principal del ajedrez, extiende de JPanel, que a su vez extiende de un componente, dado que el ajedrez ha sido creado para usarse como un componente mas, que se puede agregar fácilmente a un frame o otro panel.

public Tablero() {

super();

llenarCuadrosTablero();

ordenarTablero();

init();// para darle eventos mouse

}

/\*\*

\* Este metodo crea el tablero, quiere decir que grafica el tablero en el Jpanel.

\*/

private void llenarCuadrosTablero() {

tmp = new JLabel();

tmp.setBounds(-100, -100, 56, 56);

add(tmp);

tablero = new CuadroPieza[8][8];

setLayout(null);

for (int x = 0; x < 8; x++) {

for (int y = 0; y < 8; y++) {

tablero[x][y] = new CuadroPieza(x, y);

add(tablero[x][y]);// para agregar los elementos

}

}

rePintarTablero();

}

/\*\*

\* Inicializo todos los listener de cada pieza

\*/

public void init() {//lo que es evento con el Mouse

for (int x = 0; x < 8; x++) {

for (int y = 0; y < 8; y++) {

tablero[x][y].addMouseListener(new java.awt.event.MouseAdapter() {

public void mousePressed(java.awt.event.MouseEvent evt) {

seleccionarPieza(evt);//cuando mouse se levantado o presionado haga esto

}

public void mouseReleased(java.awt.event.MouseEvent evt) {

dejarPieza(evt);

}

});

tablero[x][y].addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {

public void mouseDragged(java.awt.event.MouseEvent evt) {

/\*

\* Para simular que la pieza es arrastrada, uso el label tmp, y le cambio sus coordenadas, con la del mouse.

\*/

tmp.setLocation(cuadroSeleccionado.getLocation().x + evt.getX() - 18, cuadroSeleccionado.getLocation().y + evt.getY() - 28);

}

}); } } } /\*\*

También tenemos una clase **IUOPCIONES.JAVA** Al hacer clic en Opciones nos muestra otro formulario, donde podemos elegir si jugar contra maquina o contra otro jugador humano, además tienes opciones de elegir el color de los casilleros y también puedes activar o desactivar alternativas de movimiento:

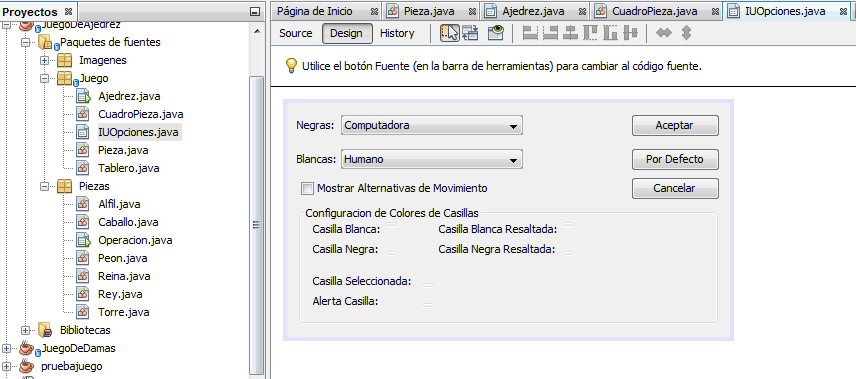


Figura : IUOPCIONES.JAVA

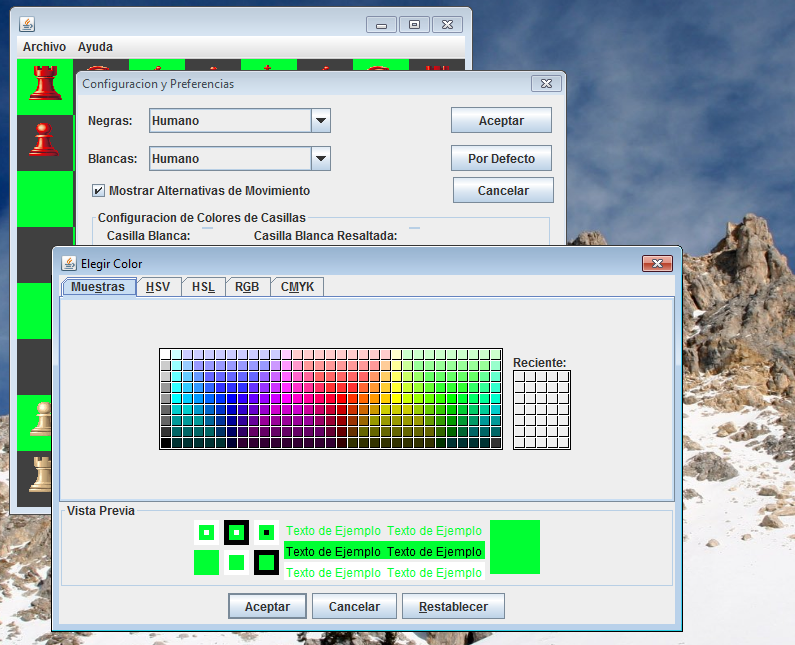


Figura 4: Muestra las opciones que tiene el juego