

UNIVERSIDAD DE MARGARITA

ALMA DEL MATER DEL CARIBE

VICERRECTORADO ACADEMICO

DECANATO DE INGENIERIA Y AFINES

ESTRUCTURA DE DATOS

**LLANOPOLY BY A.A.S**

Docente: Realizado por: Cesar Requena Samuel Marcano - 30.728.764

Angel Pérez – 30.399.625

Abdl Taktak – 30.401.549

El Valle del Espíritu Santo, 10 de julio del 2023

**Objetivo del trabajo**

Aplicar los conceptos enseñados en la materia a la resolución de un problema, trabajando en forma grupal y utilizando un lenguaje de tipado estático (Java netbeans)

**Consigna general**

Desarrollar la aplicación completa, incluyendo el modelo de clases e interfaz gráfica. La aplicación deberá ser acompañada por pruebas unitarias e integrales y documentación de diseño. En la siguiente sección se describe la aplicación a desarrollar.

**Descripción de la aplicación a desarrollar**

Se deberá desarrollar una aplicación que implemente un juego relacionado con el clásico juego de mesa MonoPoly. En el juego habrá 3 jugadores. Cada jugador comenzará con la misma cantidad de dinero equivalente a $100000 (cien mil dólares) desde la posición de salida. El tablero tendrá la siguiente disposición:

**Casillas**

1. Salida: Los 3 jugadores comienzan el juego en el mismo lugar. Se elige aleatoriamente quién tira primero, segundo y tercero.

2. UNIMAR: El jugador que caiga en esta casilla recibe un premio de $50000 (cincuenta mil dólares). Si ese jugador ya ganó una vez en la casilla UNIMAR, la segunda vez que caiga en esa casilla recibe un premio de $30000 (treinta mil dólares). Las sucesivas veces que el mismo jugador caiga en este casillero no cobrará ningún dinero.

3. Nueva Esparta:

a) Precio terreno: $20000 (veinte mil dólares)

b) Alquiler: $2000

c) Alquiler con 1 casa: $3000

d) Alquiler con 2 casas: $3500

e) Alquiler con Hotel: $5000

f) Construir casas cuestan $5000 y hotel $8000

4. CORPOELEC:

a) Comprar la compañía: $35000

b) Cobra x 500 lo que dice los dados. O sea que si un jugador no dueño de la empresa cae en esa casilla tras haber sacado 12 en los dados => deberá pagar: 12 x 500 = $6000

c) Si también tiene la compañía HIDROCARIBE cobrará x 1000 lo sacado en los dados.

5. Caracas:

a) Precio terreno: $25000 (veinte mil dólares)

b) Alquiler: $2500

c) Alquiler con 1 casa: $3500

d) Alquiler con 2 casas: $4000

e) Alquiler con Hotel: $6000

f) Construir casas cuestan $5500 y hotel $9000

6. Cárcel: un jugador al caer en este casillero debe esperar 3 turnos para salir. Es decir, recién cuando le toque por 4ta vez podrá moverse. Salvo que cuando ya haya pasado 1 turno de la cárcel (o sea en el turno 2 y 3 de espera) pague una fianza de $45000.

7. Valencia:

a) Precio terreno: $18000

b) Alquiler: $1000

c) Alquiler con 1 casa: $1500

d) Alquiler con 2 casas: $2500

e) Alquiler con Hotel: $3000

f) Construir casas cuestan $2000 y hotel $3000

8. Avance dinámico: El jugador avanzará tantos casilleros como lo indica la siguiente lógica:

a) Si sacó 2, 3, 4,5 o 6 => entonces avanza el número sacado menos 2 unidades

b) Si sacó 7, 8,9 o 10 => entonces avanza 3 espacios

c) si sacó 11 o 12 => avanza el número sacado menos la sumatoria de propiedades del jugador. Tanto los terrenos como las casas como los hoteles suman como propiedad.

9. Banesco:

a. Comprar la compañía: $40000

b. Cobra x 600 lo que dice los dados. O sea que si un jugador no dueño de la empresa cae en esa casilla tras haber sacado 12 en los dados => deberá pagar: 12 x 600 = $7200

c. Si también tiene la compañía Bus Margarita cobrará x 1100 lo sacado en los dados.

10. Mérida:

a) Precio terreno: $20000

b) Alquiler: $1300

c) Alquiler con 1 casa: $1800

d) Alquiler con 2 casas: $2900

e) Alquiler con Hotel: $3500

f) Construir casas cuestan $2200 y hotel $3500

11. Impuesto al Lujo: El jugador que caiga en esta casilla debe pagar el 10 % de todo su efectivo

12. Punto Fijo:

a) Precio terreno: $15000

b) Alquiler: $1500

c) Alquiler con 1 casa: $3500

d) Construir la única casa posible cuesta $4000

13. HIDROCARIBE:

a) Comprar la compañía: $30000

b) Cobra 300 veces lo que dice los dados. O sea que si un jugador no dueño de la empresa cae en esa casilla tras haber sacado 12 en los dados => deberá pagar: 12 x 300 = $3600

c) Si también tiene la compañía EDESUR cobrará 500 veces lo sacado en los dados.

14. Maturín:

a) Precio terreno: $23000

b) Alquiler: $2000

c) Alquiler con 1 casa: $3250

d) Alquiler con 2 casas: $3850

e) Alquiler con Hotel: $5500

f) Construir casas cuestan $4500 y hotel $7500

15. Monagas:

a) Precio terreno: $23000

b) Alquiler: $2000

c) Alquiler con 1 casa: $3250

d) Alquiler con 2 casas: $3850

e) Alquiler con Hotel: $5500

f) Construir casas cuestan $4500 y hotel $7500

16. Policía: Al caer en este casillero el jugador va a la cárcel.

17. Bus Margarita:

a) comprar la compañía: $38000

b) Cobra x 450 lo que dice los dados. O sea que si un jugador no dueño de la empresa cae en esa casilla tras haber sacado 12 en los dados => deberá pagar: 12 x 450 = $5400

c) Si también tiene la compañía Banesco cobrará 800 veces lo sacado en los dados.

18. Barinas:

a) Precio terreno: $17000

b) Alquiler: $1800

c) Alquiler con 1 casa: $3800

d) Construir la única casa posible cuesta $4800

19. Retroceso dinámico: El jugador avanzará tantos casilleros como lo indica la siguiente lógica:

a) Si sacó 2, 3, 4,5 o 6 => retrocede el número sacado menos la sumatoria de propiedades del jugador. Tanto los terrenos como las casas como los hoteles suman como propiedad.

b) Si sacó 7, 8,9 o 10 => retrocede 3 casillas

c) si sacó 11 o 12 => retrocede el número sacado menos 2 unidades

20. Anzoátegui:

a) Precio terreno: $25000

b) Alquiler: $2500

c) Alquiler con 1 edificio histórico: $4500

d) Construir la casa de hierro (única construcción posible) cuesta $7000

**Jugabilidad**

**★** Es un juego por turnos. Hay 3 jugadores. En cada turno cada jugador debe lanzar un par de dados (números del 1 al 6 cada dado). El número obtenido será la suma de ambos resultados.

★ Si un jugador al tirar los dados obtiene un doble número (o sea 1 y 1, o 3 y 3, etc) entonces tira nuevamente. Si vuelve a sacar otro doble, no tira otra vez, sino que le toca el turno al jugador siguiente.

★ El jugador debe mover en el sentido que indica la flecha del casillero de salida.

★ Los efectos ocurren al caer en un casillero y no al pasar por ellos.

★ Para poder edificar es necesario que el jugador compre las zonas del mismo color para recién poder edificar.

★ No se puede edificar en ningún terreno en el mismo turno en que se lo compra

★ Antes de lanzar los dados, cada jugador tiene la posibilidad de edificar si se encuentra en condiciones de hacerlo: Ya sea tanto contar con el dinero que sale + el terreno.

★ Para poder edificar un hotel, debe estar lleno de casas en su capacidad máxima (4). Es decir que si un jugador tiene 2 casas NO puede construir hotel. Recién al contar con 4 casas puede empezar a construir los hoteles (máximo 1 por cada terreno).

★ Los hoteles reemplazan a las casas, es decir que cuando se edifica un hotel las casas desaparecen.

★ Un jugador puede decidir vender tanto sus terrenos (con las casas u hoteles que tenga edificados) o las compañías antes de lanzar los dados. En ese caso el jugador perderá la titularidad de los mismos, los cuales pasan a estar disponibles para la compra para el resto de los jugadores. El jugador que vende cobra un 15% menos de la suma de todo lo que vende.

★ Al venderse un terreno que tenía edificaciones las mismas desaparecen del mapa. Es decir que cuando un nuevo jugador compre ese terreno, estará vacío y deberá volver a construir las casas y luego los hoteles.

**Fin del juego**

1) Cuando un jugador se quede sin dinero y sin propiedades quedará eliminado.

2) Si un jugador no tiene dinero, pero sí propiedades y debe afrontar un gasto, está obligado a vender (con la quita del 15% ya explicada) para afrontar esos gastos. Si aun vendiendo todo, no puede afrontar el gasto, el jugador quedará eliminado también.

3) Ganará el último jugador restante.

**Interfaz gráfica**

Se debe desarrollar una interfaz visual para la interacción entre los jugadores. En la misma se pondrá mucho énfasis y se evaluará como parte de la consigna la USABILIDAD de la misma.

**Entregables**

1) Código fuente de la aplicación completa, incluyendo también: código de las pruebas, archivos de recursos.

2) Pseudocódigo.

3) Informe, manual de usuario y diagrama de clases.

**Análisis del problema**

El programa debe desarrollarse en el lenguaje de programación orientado a objetos Java y además deberá estar diseñado para compilar en NetBeans; el programa a desarrollar consiste en un juego inspirado en el juego de mesa Monopoly, con algunas variaciones en las reglas y características específicas. De acuerdo con las exigencias del programa se determinan los siguientes aspectos a tomar en cuenta:

1. Estructura de datos: El juego requiere de una lista circular, la cual deberá ser doblemente enlazada para desarrollarse adecuadamente, esto de acuerdo con el funcionamiento de ciertas mecánicas especificas requeridas del programa tales como avance y retroceso dinámico.
2. Mecánica del juego: El juego se desarrolla por turnos, por lo que es necesario implementar un sistema para alternar entre los jugadores y permitirles lanzar los dados. Se debe determinar qué jugador tira primero de forma aleatoria al inicio del juego. Se debe verificar si un jugador obtiene un doble al lanzar los dados para permitirles lanzar nuevamente.
3. Movimiento de jugadores: Según los resultados de los dados, se debe implementar la lógica para mover a los jugadores a través del tablero. Esto implica cambiar la posición del jugador y verificar los efectos de la casilla en la que caen. Se deben considerar los casos especiales, como la cárcel y las casillas de avance y retroceso dinámico.
4. Interacción con casillas: Cada casilla del tablero tiene diferentes efectos, como cobrar alquileres, permitir la compra de propiedades o activar eventos especiales. Se debe implementar la lógica correspondiente para cada casilla y actualizar los atributos del jugador según sea necesario. Por ejemplo, al caer en una propiedad, se debe verificar si pertenece a otro jugador y cobrar el alquiler correspondiente.
5. Compra y construcción de propiedades: Los jugadores deben tener la opción de comprar propiedades si caen en una casilla disponible. Se debe verificar si el jugador tiene suficiente dinero y actualizar los atributos correspondientes. Además, se debe implementar la lógica para permitir la construcción de casas y hoteles en propiedades específicas.
6. Gestión del dinero: Se debe realizar un seguimiento del dinero de cada jugador y actualizarlo según las transacciones, como pagos de alquiler, premios o compras. Se deben aplicar las deducciones del impuesto al lujo cuando corresponda.
7. Eliminación de jugadores: Se debe implementar la lógica para verificar si un jugador ha quedado sin dinero y sin propiedades. En ese caso, el jugador debe ser eliminado del juego. Además, si un jugador no puede pagar un gasto y no puede vender suficientes propiedades para cubrirlo, también debe ser eliminado.
8. Finalización del juego: El juego continúa hasta que solo quede un jugador en pie. En ese momento, se debe declarar al último jugador restante como el ganador.

**Explicación del diseño y funcionamiento general a nivel lógico y grafico**

***JFrame “Inicio”***

Es una pestaña de inicio principal la cual cuenta con 3 botones los cuales otorgan las opciones de:

* Nuevo Juego: Cierra la pantalla de inicio e inicia la pantalla de juego con una llamada al JFrame Tablero.
* Salir: Cierra la aplicación por completo.
* Guía: Usa una función de llamado a un archivo de Word que contiene un manual de usuario para el jugador.

A nivel grafico se opto por un fondo que referencia directamente a los llanos venezolanos.

***JFrame “Tablero”***

Es una pestaña que otorga al usuario la vista del tablero, los iconos de jugadores y un botón para empezar la partida. Al presionar “EMPEZAR PARTIDA” se inicializan todas las lógicas necesarias para el desarrollo del videojuego, esto mediante el método initLogic() dentro del constructor del Tablero, además de otorgar a los jugadores la posibilidad de designar un Nick/nombre de jugador para cada jugador, siendo este mostrado luego en la pestaña de juego, indicando así a que jugador pertenece cada icono de juego.

A nivel lógica existe un método llamado initGame(), el cual se encarga de dar inicio al juego mediante la bienvenida al jugador, la asignación de turnos y nombres para cada jugador, luego ejecuta a un método principal llamado “mainLoop” el cual en si mismo contiene el conjunto de lógicas y métodos que componen al bucle principal que permite el desarrollo del juego hasta que este alcance su fin al haber un ganador, el método “mainLoop” en si se compone por un conjunto de algoritmos que determinan los siguientes turnos y los métodos:

* mover(): Este método se encarga de validar a que jugador pertenece el turno, si este se encuentra en la cárcel (de ser asi se llama al metodo opcionesCarcel() el cual le otorga al jugador la posibilidad de pagar fianza y salir o esperar al siguiente turno), asignarle su nueva posición de acuerdo con los dados los cuales son lanzados y determinados sus valores mediante el método lanzarDados() y de posicionar visualmente al jugador en su nueva posición mediante el método moverJugador(), el cual mediante la validación de a quién pertenece el turno se encarga de posicionar visualmente al jugador adecuado.
* validarCasilla(): Este método se encarga de validar en que casilla ha caído el jugador y de llamar a los métodos de evento determinados en caso de caer en casilla de evento, en caso de caer en casilla de propiedad o compañía se le cobra al jugador en caso de ser necesario y se le otorgan las siguientes posibilidades mediante el método opcionesGenerales() el cual despliega un panel con las opciones de comprar propiedad si es posible (esto haciendo un llamado al método comprarPropiedad() el cual valida en que propiedad se encuentra el jugador y altera su atributo de adquirida a true), pasar de turno o edificar (esta última función no está implementada). Este método es recursivo ya que en si se encuentran llamadas a si mismo en caso de ser necesario.
* validarBancarrota(): Este se encarga de validar que un jugador no haya caído en bancarrota (tener cero de dinero o menos) en cada proceso de transacción de dinero para cualquier jugador.

A nivel grafico se muestra un tablero inspirado en elementos de cultura popular venezolana, además de disponer información crucial al jugador mediante la pestaña, paneles informativos ocurrentes y en reducidos casos la consola, de fondo hay una imagen de un atardecer en los llanos venezolanos, los iconos de los jugadores son animales por que en los llanos venezolanos hay muchos animales.

***Clase Nodo***

Es la clase que compone a la lista circular doblemente enlazada, cada instancia de un nodo almacena una instancia de la clase casilla y dos instancias de la clase nodo, una llamada siguiente y otra llamada anterior.

***Clase Lista***

Es la clase que tiene todos los métodos y la lógica funcional para la incorporación de la lista circular doblemente enlazada. Se compone de un nodo cabeza y un nodo cola y cuenta con los siguientes métodos:

* estaVacia(): Verifica si la lista esta vacía y devuelve un booleano de la verificación, true en caso de estar vacía, false en caso de no estarlo.
* insertar(): Algoritmo que agrega un nodo de manera adecuada a la lista.
* eliminar(): Algoritmo que busca en la lista un nodo especifico para luego eliminarlo de la lista y ajustarla de acuerdo con su eliminación.
* imprimir(): Algoritmo que imprime en consola toda la lista y la información de cada nodo dentro de la lista.
* buscarPosicionX(): De acuerdo con la información suministrada al método, este devuelve la posición en x para disponer visualmente al jugador que se encuentra ejecutando su turno.
* buscarPosicionY(): De acuerdo con la información suministrada al método, este devuelve la posición en y para disponer visualmente al jugador que se encuentra ejecutando su turno.

***Clase Jugador***

Es la clase que permite la implementación de jugadores de acuerdo con los atributos específicos necesarios para el jugador, mediante sus métodos getters and setters se desarrollan todas las lógicas necesarias dentro del JFrame “Tablero”.

***Clase Casilla***

Es la clase que permite la implementación de casillas de acuerdo con los atributos específicos necesarios para el desarrollo del juego, mediante sus métodos getters and setters se desarrollan todas las lógicas necesarias dentro del JFrame “Tablero”.

***Clase Propiedad***

Hereda de la super clase “Casilla” y es la clase que permite la implementación de casillas de Propiedad de acuerdo con los atributos específicos necesarios para el desarrollo del juego, mediante sus métodos getters and setters se desarrollan todas las lógicas necesarias dentro del JFrame “Tablero”.

***Clase PropiedadExtendida***

Hereda de la super clase “Casilla” y es la clase que permite la implementación de casillas de PropiedadExtendida de acuerdo con los atributos específicos necesarios para el desarrollo del juego, mediante sus métodos getters and setters se desarrollan todas las lógicas necesarias dentro del JFrame “Tablero”.

***Clase Compañía***

Hereda de la super clase “Casilla” y es la clase que permite la implementación de casillas de compañia de acuerdo con los atributos específicos necesarios para el desarrollo del juego, mediante sus métodos getters and setters se desarrollan todas las lógicas necesarias dentro del JFrame “Tablero”.

***Clase Evento***

Hereda de la super clase “Casilla” y es la clase que permite la implementación de casillas de evento de acuerdo con los métodos específicos necesarios para el desarrollo del juego, mediante sus métodos getters and setters se desarrollan todas las lógicas necesarias dentro del JFrame “Tablero”. Este cuenta con los siguientes métodos:

* avanceDinamico(): Hace el cálculo de cuantas casillas deberá moverse el jugador hacia delante de acuerdo con el valor de los dados y las propiedades y retorna ese valor para poder ser usado.
* retrocesoDinamico(): Hace el cálculo de cuantas casillas deberá moverse el jugador hacia atrás de acuerdo con el valor de los dados y las propiedades y retorna ese valor para poder ser usado.
* policía(): Transforma el atributo booleano de cárcel del jugador otorgado al método (jugador del turno actual) en true y cambia su posición a la 8 (posición de la casilla cárcel).
* unimar(): Le otorga $50000 si es la primera vez que cae en la casilla o $30000 si es la segunda vez que cae en la casilla o nada si es la tercera vez en adelante al jugador del turno actual, además incrementa el numero de veces que ese jugador ha caído en unimar para permitir las validaciones de manera correcta.
* impuestoLujo(): Le resta el 10% de su efectivo al jugador del turno actual.

**PSEUDOCODIGO**

***JFrame Inicio***

package monopolio;

Clase pública Inicio extiende JFrame {

metodo Inicio() {

método initComponents();

this.setLocationRelativeTo(null);

}

metodo initComponents() {

// Inicializar los componentes de la interfaz

}

Metodo privado jButton1ActionPerformed() {

Tablero btnTablero = nuevo Tablero();

btnTablero.setVisible(verdadero);

this.setVisible(falso);

}

Metodo privado jButton2ActionPerformed() {

File archivoWord = nuevo File (“MANUEAL DE USUARIO LLANOPOLY.docx”)

Si (archivoWord.exists()) {

Intenta{

Desktop.getDesktop().open(archivoWord);

} catch (IOException e) {e.printStackTrace();}

} else{ Imprime (“El archive no existe”);

}

}

Metodo privado jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

System.exit(0);

}

Metodo main(String args[]) {

intentar {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} Atrapar (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Inicio.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} Atrapar (InstantiationException ex) {

java.util.logging.Logger.getLogger(Inicio.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} Atrapar (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Inicio.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} Atrapar (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Inicio.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

java.awt.EventQueue.invokeLater(new Runnable() {

Metodo run() {

nuevo Inicio().setVisible(verdadero);

}

});

}

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JButton jButton3;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JPanel jPanel3;

}

***JFrame Tablero***

package monopolio;

//Importar las clases y paquetes necesarios

Clase Tablero extiende JFrame {

metodoTablero() {

initComponents();

this.setLocationRelativeTo(null);

}

Metodo jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

// Empezar partida boton

jButton2.setVisible(falso);

método initLogic();

metodo initGame();

}

Metodo initLogic(){

// Inicializacion de la lista //

lista.insertar(salida);

lista.insertar(puntoFijo);

lista.insertar(busMargarita);

lista.insertar(caracas);

lista.insertar(unimar);

lista.insertar(nuevaEsparta);

lista.insertar(corpoelec);

lista.insertar(carcel);

lista.insertar(valencia);

lista.insertar(avanceDinamico);

lista.insertar(banesco);

lista.insertar(merida);

lista.insertar(impuestoLujo);

lista.insertar(maturin);

lista.insertar(hidrocaribe);

lista.insertar(policia);

lista.insertar(monagas);

lista.insertar(barinas);

lista.insertar(retrocesoDinamico);

lista.insertar(anzoategui);

//lista.imprimir();

jugadores[0] = J1;

jugadores[1] = J2;

jugadores[2] = J3;

}

Metodo initGame(){

JOptionPane.showMessageDialog(null, "Bienvenido a LLANOPOLY ", "¡Bienvenido!", JOptionPane.INFORMATION\_MESSAGE);

//Permite a los jugadores asignar nombres a sus personajes

boolean nombrevalido=false;

Para(int i=0; i<3; i++){

Mientras (nombrevalido==false) {

jugadores[i].setNombre(JOptionPane.showInputDialog("Ingrese el nick para el Jugador " + (i+1)));

nombrevalido=true;

Si (jugadores[i].nombre == null || jugadores[i].nombre.trim().isEmpty()) {

nombrevalido=false;

JOptionPane.showMessageDialog(null, "Debes introducir algo.");

}

}

Fin mientras

nombrevalido=false;

}

Fin para

jLabel11.setText(J1.nombre);

jLabel15.setText(J2.nombre);

jLabel19.setText(J3.nombre);

JOptionPane.showMessageDialog(null, "A continuacion se decidira quien comenzara la partida", "...", JOptionPane.INFORMATION\_MESSAGE);

JOptionPane.showMessageDialog(null, "Esto mediante el lanzamiento de 3 pares de dados", "...", JOptionPane.INFORMATION\_MESSAGE);

JOptionPane.showMessageDialog(null, "El numero mas alto sera el primero en jugar", "...", JOptionPane.INFORMATION\_MESSAGE);

dJ1= random.nextInt(6) + 1 + random.nextInt(6) + 1;

dJ2= random.nextInt(6) + 1 + random.nextInt(6) + 1;

Si (dJ2 == dJ1) dJ2 = dJ2-1;

Fin si

dJ3= random.nextInt(6) + 1 + random.nextInt(6) + 1;

si(dJ2 == dJ3 | dJ3 == dJ1) dJ3 = dJ2-1;

fin si

JOptionPane.showMessageDialog(null, "El jugador 1 obtiene el numero: " + dJ1, "Resultado", JOptionPane.INFORMATION\_MESSAGE);

JOptionPane.showMessageDialog(null, "El jugador 2 obtiene el numero: " + dJ2, "Resultado", JOptionPane.INFORMATION\_MESSAGE);

JOptionPane.showMessageDialog(null, "El jugador 3 obtiene el numero: " + dJ3, "Resultado", JOptionPane.INFORMATION\_MESSAGE);

// Valida y asigna los turnos de los jugadores

Si (dJ1 > dJ2 && dJ1 > dJ3) {

J1.turnoJugador = 3;

JOptionPane.showMessageDialog(null, "El Jugador "+ J1.getNombre() + " tendra el primer turno", "Primer turno", JOptionPane.INFORMATION\_MESSAGE);

Si (dJ2 > dJ3) {

J2.turnoJugador = 2;

J3.turnoJugador = 1;

} Sino {

J2.turnoJugador = 1;

J3.turnoJugador = 2;

}

} Sino entonces (dJ2 > dJ1 && dJ2 > dJ3) {

J2.turnoJugador = 3;

JOptionPane.showMessageDialog(null, "El Jugador "+ J2.getNombre() + " tendra el primer turno", "Primer turno", JOptionPane.INFORMATION\_MESSAGE);

Si (dJ1 > dJ3) {

J1.turnoJugador = 2;

J3.turnoJugador = 1;

} Sino {

J1.turnoJugador = 1;

J3.turnoJugador = 2;

}

} Sino {

J3.turnoJugador = 3;

JOptionPane.showMessageDialog(null, "El Jugador "+ J3.getNombre() + " tendra el primer turno", "Primer turno", JOptionPane.INFORMATION\_MESSAGE);

Si (dJ1 > dJ2) {

J1.turnoJugador = 2;

J2.turnoJugador = 1;

} Sino {

J1.turnoJugador = 1;

J2.turnoJugador = 2;

}

}

Fin si

Fin si

Fin si

Fin si

//System.out.println("Turnos: " + J1.turnoJugador + " " + J2.turnoJugador + " " + J3.turnoJugador);

System.out.println("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

System.out.println("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

System.out.println("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

Metodo mainLoop();

jButton2.setVisible(Verdadero);

}

Metodo validarCasilla(){

Para(int i=0; i<3; i++){

Si(jugadores[i].turnoJugador == 3){

Si (jugadores[i].posicionJugador == 5 | jugadores[i].posicionJugador == 8 | jugadores[i].posicionJugador == 10 | jugadores[i].posicionJugador == 13 | jugadores[i].posicionJugador == 16 | jugadores[i].posicionJugador == 19| jugadores[i].posicionJugador == 1){

Casos de (jugadores[i].posicionJugador){

Caso 5:

JOptionPane.showMessageDialog(null, "Caiste en UNIMAR", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

if (jugadores[i].turnoJugador== 3) unimar.unimar(jugadores[i]);

System.out.println("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

System.out.println("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

System.out.println("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

Fin si

Fin caso;

Caso 8:

JOptionPane.showMessageDialog(null, "Caiste en Carcel", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Fin caso;

Caso 10:

JOptionPane.showMessageDialog(null, "Caiste en Avance Dinamico", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Para(int j=0; j<3; j++){

Si (jugadores[j].turnoJugador== 3) {

totaldados= avanceDinamico.avanceDinamico(totaldados, jugadores[j].cantidadPropiedades);

Si (totaldados == 0) totaldados = 1;

JOptionPane.showMessageDialog(null, "Avanzaras la siguiente cantidad de casillas: "+totaldados, "Avance Dinamico", JOptionPane.INFORMATION\_MESSAGE);

Si (J1.turnoJugador==3) {

x= jLabel1.getX();

y= jLabel1.getY();

Para (int movFal = 0; movFal < totaldados; movFal++){

J1.setPosicionJugador((J1.getPosicionJugador() + 1));

Si(J1.getPosicionJugador()>20){

J1.setPosicionJugador(1);

}

Fin si

Imprime (J1.getPosicionJugador());

xn = lista.buscarPosicionX(J1.getPosicionJugador(), xn, J1, J2, J3);

yn = lista.buscarPosicionY(J1.getPosicionJugador(), yn, J1, J2, J3);

}

Fin para

Metodo moverJugador(x, y, xn, yn);

}

Fin si

Si (J2.turnoJugador==3) {

x = jLabel2.getX();

y = jLabel2.getY();

Para (int movFal = 0; movFal < totaldados; movFal++){

J2.setPosicionJugador((J2.getPosicionJugador() + 1));

Si(J2.getPosicionJugador()>20){

J2.setPosicionJugador(1);

}

Fin si

Imprime (J2.getPosicionJugador());

xn= lista.buscarPosicionX(J2.getPosicionJugador(), xn, J1, J2, J3);

yn= lista.buscarPosicionY(J2.getPosicionJugador(), yn, J1, J2, J3);

}

Fin para

Metodo moverJugador(x, y, xn, yn);

}

Fin si

Si (J3.turnoJugador==3) {

x= jLabel4.getX();

y= jLabel4.getY();

Para (int movFal = 0; movFal < totaldados; movFal++){

J3.setPosicionJugador((J3.getPosicionJugador() + 1));

Si(J3.getPosicionJugador()>20){

J3.setPosicionJugador(1);

}

Fin si

Imprime (J3.getPosicionJugador());

xn= lista.buscarPosicionX(J3.getPosicionJugador(), xn, J1, J2, J3);

yn= lista.buscarPosicionY(J3.getPosicionJugador(), yn, J1, J2, J3);

}

Fin para

Metodo moverJugador(x, y, xn, yn);

}

Fin si

}

Fin si

}

Fin para

Metodo validarCasilla();

Fin caso;

Caso 13:

JOptionPane.showMessageDialog(null, "Caiste en Impuesto al Lujo", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

impuestoLujo.impuestoLujo(jugadores[i]);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

validarBancarrota(jugadores[i]);

Fin caso;

Caso 16:

JOptionPane.showMessageDialog(null, "Caiste en Policia\n¡Iras a la carcel!", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

policia.policia(jugadores[i]);

Imprime (jugadores[i].getPosicionJugador());

xn= lista.buscarPosicionX(jugadores[i].getPosicionJugador(), xn, J1, J2, J3);

yn= lista.buscarPosicionY(jugadores[i].getPosicionJugador(), yn, J1, J2, J3);

Metodo moverJugador(x, y, xn, yn);

Fin caso;

Caso19:

JOptionPane.showMessageDialog(null, "Caiste en Retroceso Dinamico", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Para(int j=0; j<3; j++){

Si (jugadores[j].turnoJugador== 3) {

totaldados= retrocesoDinamico.retrocesoDinamico(totaldados, jugadores[j].cantidadPropiedades);

Si (totaldados == 0) totaldados = 1;

JOptionPane.showMessageDialog(null, "Retrocederas la siguiente cantidad de casillas: "+totaldados, "Retroceso Dinamico", JOptionPane.INFORMATION\_MESSAGE);

Si (J1.turnoJugador==3) {

x= jLabel1.getX();

y= jLabel1.getY();

Para (int movFal = 0; movFal < totaldados; movFal++){

J1.setPosicionJugador((J1.getPosicionJugador() - 1));

Imprime (J1.getPosicionJugador());

xn = lista.buscarPosicionX(J1.getPosicionJugador(), xn, J1, J2, J3);

yn = lista.buscarPosicionY(J1.getPosicionJugador(), yn, J1, J2, J3);

}

Fin para

Metodo moverJugador(x, y, xn, yn);

}

Fin si

Si (J2.turnoJugador==3) {

x = jLabel2.getX();

y = jLabel2.getY();

Para (int movFal = 0; movFal < totaldados; movFal++){

J2.setPosicionJugador((J2.getPosicionJugador() - 1));

Imprime(J2.getPosicionJugador());

xn= lista.buscarPosicionX(J2.getPosicionJugador(), xn, J1, J2, J3);

yn= lista.buscarPosicionY(J2.getPosicionJugador(), yn, J1, J2, J3);

}

Fin para

Metodo moverJugador(x, y, xn, yn);

}

Fin si

Si (J3.turnoJugador==3) {

x= jLabel4.getX();

y= jLabel4.getY();

Para (int movFal = 0; movFal < totaldados; movFal++){

J3.setPosicionJugador((J3.getPosicionJugador() - 1));

Imprime (J3.getPosicionJugador());

xn= lista.buscarPosicionX(J3.getPosicionJugador(), xn, J1, J2, J3);

yn= lista.buscarPosicionY(J3.getPosicionJugador(), yn, J1, J2, J3);

}

Fin para

Metodo moverJugador(x, y, xn, yn);

}

Fin si

}

Fin si

}

Fin para

Metodo validarCasilla();

Fin caso;

Caso 1:

JOptionPane.showMessageDialog(null, "Caiste en Salida", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Fin caso;

}} else {

Casos de (jugadores[i].posicionJugador){

Caso 2:

JOptionPane.showMessageDialog(null, "Caiste en Punto Fijo", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Si (puntoFijo.adquirida == true){

jugadores[i].setDinero(jugadores[i].getDinero() - puntoFijo.alquilerInicial);

JOptionPane.showMessageDialog(null, "Se acaban de pagar "+ puntoFijo.alquilerInicial + " al propietario", "Cobro", JOptionPane.INFORMATION\_MESSAGE);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Fin si

Metodo validarBancarrota(jugadores[i]);

Fin caso;

Caso 3:

JOptionPane.showMessageDialog(null, "Caiste en Bus Margarita", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Si (busMargarita.adquirida == Verdadero) {

jugadores[i].setDinero(jugadores[i].getDinero() - (busMargarita.cobro \* totaldados));

JOptionPane.showMessageDialog(null, "Se acaban de pagar "+ (busMargarita.cobro \* totaldados) + " al propietario", "Cobro", JOptionPane.INFORMATION\_MESSAGE);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Fin si

validarBancarrota(jugadores[i]);

Fin caso;

Caso 4:

JOptionPane.showMessageDialog(null, "Caiste en Caracas", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Si (caracas.adquirida == Verdadero) {

jugadores[i].setDinero(jugadores[i].getDinero() - caracas.alquilerInicial);

JOptionPane.showMessageDialog(null, "Se acaban de pagar "+ caracas.alquilerInicial + " al propietario", "Cobro", JOptionPane.INFORMATION\_MESSAGE);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Fin si

validarBancarrota(jugadores[i]);

Fin caso;

Caso 6:

JOptionPane.showMessageDialog(null, "Caiste en Nueva Esparta", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Si (nuevaEsparta.adquirida == Verdadero){

jugadores[i].setDinero(jugadores[i].getDinero() - nuevaEsparta.alquilerInicial);

JOptionPane.showMessageDialog(null, "Se acaban de pagar "+ nuevaEsparta.alquilerInicial + " al propietario", "Cobro", JOptionPane.INFORMATION\_MESSAGE);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Fin si

Metodo validarBancarrota(jugadores[i]);

Fin caso;

Caso 7:

JOptionPane.showMessageDialog(null, "Caiste en Corpoelec", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Si (corpoelec.adquirida == Verdadero){

jugadores[i].setDinero(jugadores[i].getDinero() - (corpoelec.cobro \* totaldados));

JOptionPane.showMessageDialog(null, "Se acaban de pagar "+ (corpoelec.cobro \* totaldados) + " al propietario", "Cobro", JOptionPane.INFORMATION\_MESSAGE);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Metodo validarBancarrota(jugadores[i]);

Fin caso;

Caso 9:

JOptionPane.showMessageDialog(null, "Caiste en Valencia", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Si (valencia.adquirida == Verdadero){

jugadores[i].setDinero(jugadores[i].getDinero() - valencia.alquilerInicial);

JOptionPane.showMessageDialog(null, "Se acaban de pagar "+ valencia.alquilerInicial + " al propietario", "Cobro", JOptionPane.INFORMATION\_MESSAGE);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Fin si

Metodo validarBancarrota(jugadores[i]);

Fin caso;

Caso 11:

JOptionPane.showMessageDialog(null, "Caiste en Banesco", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Si (banesco.adquirida == Verdadero){

jugadores[i].setDinero(jugadores[i].getDinero() - (banesco.cobro \* totaldados));

JOptionPane.showMessageDialog(null, "Se acaban de pagar "+ (banesco.cobro \* totaldados) + " al propietario", "Cobro", JOptionPane.INFORMATION\_MESSAGE);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Metodo validarBancarrota(jugadores[i]);

Fin caso;

Caso12:

JOptionPane.showMessageDialog(null, "Caiste en Merida", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Si (merida.adquirida == Verdadero){

jugadores[i].setDinero(jugadores[i].getDinero() - merida.alquilerInicial);

JOptionPane.showMessageDialog(null, "Se acaban de pagar "+ merida.alquilerInicial + " al propietario", "Cobro", JOptionPane.INFORMATION\_MESSAGE);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Metodo validarBancarrota(jugadores[i]);

Fin si;

Caso 14:

JOptionPane.showMessageDialog(null, "Caiste en Maturin", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Si (maturin.adquirida == Verdadero){

jugadores[i].setDinero(jugadores[i].getDinero() - maturin.alquilerInicial);

JOptionPane.showMessageDialog(null, "Se acaban de pagar "+ maturin.alquilerInicial + " al propietario", "Cobro", JOptionPane.INFORMATION\_MESSAGE);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Metodo validarBancarrota(jugadores[i]);

Fin caso;

Caso15:

JOptionPane.showMessageDialog(null, "Caiste en Hidrocaribe", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Si (hidrocaribe.adquirida == true){

jugadores[i].setDinero(jugadores[i].getDinero() - (hidrocaribe.cobro \* totaldados));

JOptionPane.showMessageDialog(null, "Se acaban de pagar "+ (hidrocaribe.cobro \* totaldados) + " al propietario", "Cobro", JOptionPane.INFORMATION\_MESSAGE);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Metodo validarBancarrota(jugadores[i]);

Fin caso;

Caso 17:

JOptionPane.showMessageDialog(null, "Caiste en Monagas", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Si (monagas.adquirida ==Verdadero){

jugadores[i].setDinero(jugadores[i].getDinero() - monagas.alquilerInicial);

JOptionPane.showMessageDialog(null, "Se acaban de pagar "+ monagas.alquilerInicial + " al propietario", "Cobro", JOptionPane.INFORMATION\_MESSAGE);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Metodo validarBancarrota(jugadores[i]);

Fin caso;

Caso 18:

JOptionPane.showMessageDialog(null, "Caiste en Barinas", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Si (barinas.adquirida == Verdadero){

jugadores[i].setDinero(jugadores[i].getDinero() - barinas.alquilerInicial);

JOptionPane.showMessageDialog(null, "Se acaban de pagar "+ barinas.alquilerInicial + " al propietario", "Cobro", JOptionPane.INFORMATION\_MESSAGE);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Metodo validarBancarrota(jugadores[i]);

Fin caso;

Caso 20:

JOptionPane.showMessageDialog(null, "Caiste en Anzoategui", "Casilla", JOptionPane.INFORMATION\_MESSAGE);

Si (anzoategui.adquirida == Verdadero){

jugadores[i].setDinero(jugadores[i].getDinero() - anzoategui.alquilerInicial);

JOptionPane.showMessageDialog(null, "Se acaban de pagar "+ anzoategui.alquilerInicial + " al propietario", "Cobro", JOptionPane.INFORMATION\_MESSAGE);

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Metodo validarBancarrota(jugadores[i]);

Fin caso;

}

opcionesGenerales();

}

}

}

}

Metodo opcionesGenerales(){

// Mensaje y título del cuadro de diálogo

int opcionSeleccionada = JOptionPane.showOptionDialog(null, "¿Qué acción deseas realizar?", "Acciones", JOptionPane.DEFAULT\_OPTION, JOptionPane.QUESTION\_MESSAGE, null, opciones, null);

// Comprobar la opción seleccionada y ejecutar la línea de código correspondiente

Caso de (opcionSeleccionada) {

Caso 0: // Pasar turno

// Línea de código para pasar turno

Imprime ("Se pasa al siguiente turno");

Fin caso;

Caso 1: // Comprar propiedad

// Línea de código para comprar una propiedad

Metodo comprarPropiedad();

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

Fin Caso ;

Caso 2: // Edificar

// Línea de código para edificar

JOptionPane.showMessageDialog(null, "No puedes edificar", "Error", JOptionPane.ERROR\_MESSAGE);

Fin Caso;

default:

// No se seleccionó ninguna opción válida

Fin caso;

}

}

Metodo comprarPropiedad(){

Para(int i=0; i<3; i++){

Si (jugadores[i].turnoJugador == 3){

Casos de (jugadores[i].posicionJugador){

Caso 2:

Si (puntoFijo.adquirida == falso){

JOptionPane.showMessageDialog(null, "Acabas de comprar la propiedad 'Punto Fijo'\nPor "+ puntoFijo.precioPropiedad, "Compra", JOptionPane.INFORMATION\_MESSAGE);

jugadores[i].dinero = jugadores[i].dinero - puntoFijo.precioPropiedad;

puntoFijo.adquirida = Verdadero;

jugadores[i].cantidadPropiedades= jugadores[i].cantidadPropiedades+1;

} sino {JOptionPane.showMessageDialog(null, "Esta propiedad no puede ser comprada", "Error", JOptionPane.ERROR\_MESSAGE); }

Fin si

Fin si

Fin para

Fin caso;

Caso 3:

Si (busMargarita.adquirida == falso){

JOptionPane.showMessageDialog(null, "Acabas de comprar la compañia 'Bus Margarita'\nPor "+ busMargarita.precioCompañia, "Compra", JOptionPane.INFORMATION\_MESSAGE);

jugadores[i].dinero = jugadores[i].dinero - busMargarita.precioCompañia;

busMargarita.adquirida = Verdadero;

jugadores[i].cantidadPropiedades= jugadores[i].cantidadPropiedades+1;

} Sino {JOptionPane.showMessageDialog(null, "Esta propiedad no puede ser comprada", "Error", JOptionPane.ERROR\_MESSAGE); }

Fin si

break;

Caso 4:

Si (caracas.adquirida == falso){

JOptionPane.showMessageDialog(null, "Acabas de comprar la propiedad 'Caracas'\nPor "+ caracas.precioPropiedad, "Compra", JOptionPane.INFORMATION\_MESSAGE);

jugadores[i].dinero = jugadores[i].dinero - caracas.precioPropiedad;

caracas.adquirida = Verdadero;

jugadores[i].cantidadPropiedades= jugadores[i].cantidadPropiedades+1;

} Sino {JOptionPane.showMessageDialog(null, "Esta propiedad no puede ser comprada", "Error", JOptionPane.ERROR\_MESSAGE); }

Fin si

Fin si;

Caso 6:

Si (nuevaEsparta.adquirida == falso){

JOptionPane.showMessageDialog(null, "Acabas de comprar la propiedad 'Nueva Esparta'\nPor "+ nuevaEsparta.precioPropiedad, "Compra", JOptionPane.INFORMATION\_MESSAGE);

jugadores[i].dinero = jugadores[i].dinero - nuevaEsparta.precioPropiedad;

nuevaEsparta.adquirida = Verdadero;

jugadores[i].cantidadPropiedades= jugadores[i].cantidadPropiedades+1;

} Sino{JOptionPane.showMessageDialog(null, "Esta propiedad no puede ser comprada", "Error", JOptionPane.ERROR\_MESSAGE); }

Fin si

Fin caso;

Caso 7:

Si (corpoelec.adquirida == falso){

JOptionPane.showMessageDialog(null, "Acabas de comprar la compañia 'Corpoelec'\nPor "+ corpoelec.precioCompañia, "Compra", JOptionPane.INFORMATION\_MESSAGE);

jugadores[i].dinero = jugadores[i].dinero - corpoelec.precioCompañia;

corpoelec.adquirida = Verdadero;

jugadores[i].cantidadPropiedades= jugadores[i].cantidadPropiedades+1;

} Sino{JOptionPane.showMessageDialog(null, "Esta propiedad no puede ser comprada", "Error", JOptionPane.ERROR\_MESSAGE); }

Fin si

Fin Caso;

Caso 9:

Si (valencia.adquirida == falso){

JOptionPane.showMessageDialog(null, "Acabas de comprar la propiedad 'Valencia'\nPor "+ valencia.precioPropiedad, "Compra", JOptionPane.INFORMATION\_MESSAGE);

jugadores[i].dinero = jugadores[i].dinero - valencia.precioPropiedad;

valencia.adquirida = Verdadero;

jugadores[i].cantidadPropiedades= jugadores[i].cantidadPropiedades+1;

} Sino{JOptionPane.showMessageDialog(null, "Esta propiedad no puede ser comprada", "Error", JOptionPane.ERROR\_MESSAGE); }

Fin si

Fin caso;

caso 11:

Si (banesco.adquirida == falso){

JOptionPane.showMessageDialog(null, "Acabas de comprar la compañia 'Banesco'\nPor "+ banesco.precioCompañia, "Compra", JOptionPane.INFORMATION\_MESSAGE);

jugadores[i].dinero = jugadores[i].dinero - banesco.precioCompañia;

banesco.adquirida = Verdadero;

jugadores[i].cantidadPropiedades= jugadores[i].cantidadPropiedades+1;

} Sino{JOptionPane.showMessageDialog(null, "Esta propiedad no puede ser comprada", "Error", JOptionPane.ERROR\_MESSAGE); }

Fin si

Fin caso;

Caso 12:

If (merida.adquirida == falso){

JOptionPane.showMessageDialog(null, "Acabas de comprar la propiedad 'Merida'\nPor "+ merida.precioPropiedad, "Compra", JOptionPane.INFORMATION\_MESSAGE);

jugadores[i].dinero = jugadores[i].dinero - merida.precioPropiedad;

merida.adquirida = Verdadero;

jugadores[i].cantidadPropiedades= jugadores[i].cantidadPropiedades+1;

} Sino{JOptionPane.showMessageDialog(null, "Esta propiedad no puede ser comprada", "Error", JOptionPane.ERROR\_MESSAGE); }

Fin si

Fin caso;

Caso 14:

Si (maturin.adquirida == falso){

JOptionPane.showMessageDialog(null, "Acabas de comprar la propiedad 'Maturin'\nPor "+ maturin.precioPropiedad, "Compra", JOptionPane.INFORMATION\_MESSAGE);

jugadores[i].dinero = jugadores[i].dinero - maturin.precioPropiedad;

maturin.adquirida = Verdadero;

jugadores[i].cantidadPropiedades= jugadores[i].cantidadPropiedades+1;

} Sino{JOptionPane.showMessageDialog(null, "Esta propiedad no puede ser comprada", "Error", JOptionPane.ERROR\_MESSAGE); }

Fin si

break;

Caso 15:

Si (hidrocaribe.adquirida == Falso){

JOptionPane.showMessageDialog(null, "Acabas de comprar la compañia 'Hidrocaribe'\nPor "+ hidrocaribe.precioCompañia, "Compra", JOptionPane.INFORMATION\_MESSAGE);

jugadores[i].dinero = jugadores[i].dinero - hidrocaribe.precioCompañia;

hidrocaribe.adquirida = Verdadero;

jugadores[i].cantidadPropiedades= jugadores[i].cantidadPropiedades+1;

} Sino {JOptionPane.showMessageDialog(null, "Esta propiedad no puede ser comprada", "Error", JOptionPane.ERROR\_MESSAGE); }

Fin si

Fin caso;

caso 17:

Si (monagas.adquirida == falso){

JOptionPane.showMessageDialog(null, "Acabas de comprar la propiedad 'Monagas'\nPor "+ monagas.precioPropiedad, "Compra", JOptionPane.INFORMATION\_MESSAGE);

jugadores[i].dinero = jugadores[i].dinero - monagas.precioPropiedad;

monagas.adquirida = Verdadero;

jugadores[i].cantidadPropiedades= jugadores[i].cantidadPropiedades+1;

} Sino{JOptionPane.showMessageDialog(null, "Esta propiedad no puede ser comprada", "Error", JOptionPane.ERROR\_MESSAGE); }

Fin si

Fin caso;

Caso 18:

Si (barinas.adquirida == falso){

JOptionPane.showMessageDialog(null, "Acabas de comprar la propiedad 'Barinas'\nPor "+ barinas.precioPropiedad, "Compra", JOptionPane.INFORMATION\_MESSAGE);

jugadores[i].dinero = jugadores[i].dinero - barinas.precioPropiedad;

barinas.adquirida = Verdadero;

jugadores[i].cantidadPropiedades= jugadores[i].cantidadPropiedades+1;

} Sino{JOptionPane.showMessageDialog(null, "Esta propiedad no puede ser comprada", "Error", JOptionPane.ERROR\_MESSAGE); }

Fin si

Fin caso;

caso 20:

if (anzoategui.adquirida == falso){

JOptionPane.showMessageDialog(null, "Acabas de comprar la propiedad 'Anzoategui'\nPor "+ anzoategui.precioPropiedad, "Compra", JOptionPane.INFORMATION\_MESSAGE);

jugadores[i].dinero = jugadores[i].dinero - anzoategui.precioPropiedad;

anzoategui.adquirida = Verdadero;

jugadores[i].cantidadPropiedades= jugadores[i].cantidadPropiedades+1;

} Sino{JOptionPane.showMessageDialog(null, "Esta propiedad no puede ser comprada", "Error", JOptionPane.ERROR\_MESSAGE); }

Fin si

Fin caso;

}

}

Metodo validarBancarrota(jugadores[i]);

}

}

Metodo opcionesCarcel(){

// Mensaje y título del cuadro de diálogo

int opcionSeleccionada = JOptionPane.showOptionDialog(null, "¿Qué acción deseas realizar?", "Acciones en carcel", JOptionPane.DEFAULT\_OPTION, JOptionPane.QUESTION\_MESSAGE, null, opcionesCarcel, null);

// Comprobar la opción seleccionada y ejecutar la línea de código correspondiente

Caso de (opcionSeleccionada) {

caso 0:

// Línea de código para pasar turno

Imprime ("Se pasa al siguiente turno");

Fin caso;

caso 1:

// Línea de código para pagar fianza

Imprime("Se paga la fianza");

Para (int i = 0; i<3 ; i++){

Si (jugadores[i].turnoJugador==3) {

jugadores[i].setDinero(jugadores[i].getDinero() - 45000);

jugadores[i].carcel = falso;

jugadores[i].turnosCarcel = 0;

pagoFianza = Verdadero;

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Fin si

}

Fin para

Fin caso;

}

}

Metodo mainLoop(){

Mientras (bancasrrotas < 2){

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

Metodo mover();

Metodo validarCasilla();

Si (J1.turnoJugador == 3){

Si (turnoDoble==false | turnoDobleN >= 2){

J1.turnoJugador = 0;

turnoDobleN = 0;

}

Fin si

}

Fin si

Si(J2.turnoJugador == 3){

Si (turnoDoble==false | turnoDobleN >= 2){

J2.turnoJugador = 0;

turnoDobleN = 0;

}

Fin si

}

Fin si

Si(J3.turnoJugador == 3){

Si (turnoDoble==false | turnoDobleN >= 2){

J3.turnoJugador = 0;

turnoDobleN = 0;

}

Fin si

}

Fin si

Si (turnoDoble==false | turnoDobleN >= 2){

J1.turnoJugador = J1.turnoJugador + 1;

J2.turnoJugador = J2.turnoJugador + 1;

J3.turnoJugador = J3.turnoJugador + 1;

Para (int i=0; i<3; i++){

if (jugadores[i].turnoJugador == 3) JOptionPane.showMessageDialog(null, "es el turno de " + jugadores[i].getNombre(), "Siguiente turno", JOptionPane.INFORMATION\_MESSAGE);

}

Fin para

} Sino{

JOptionPane.showMessageDialog(null, "Se repite el turno", "¡Salieron dobles!", JOptionPane.INFORMATION\_MESSAGE);

}

Fin si

totaldados=0;

Imprime ("Jugador: "+J1.nombre +"\nDinero: "+J1.dinero);

Imprime ("Jugador: "+J2.nombre +"\nDinero: "+J2.dinero);

Imprime ("Jugador: "+J3.nombre +"\nDinero: "+J3.dinero);

}

Fin si

Para (int i = 0; i<3 ; i++){

Si (jugadores[i].bancarrota== false) JOptionPane.showMessageDialog(null, "El ganador ha sido el jugador "+jugadores[i].nombre, "¡Felicidades!", JOptionPane.INFORMATION\_MESSAGE);

}

Fin si

}

Fin para

Metodo lanzarDados(){

dado1 = random.nextInt(6) + 1;

dado2 = random.nextInt(6) + 1;

JOptionPane.showMessageDialog(null, "El dado 1 ha dado: " + dado1 + " | El dado 2 ha dado: "+dado2, "Resultado Dados", JOptionPane.INFORMATION\_MESSAGE);

totaldados = dado1 + dado2;

}

Metodo validarBancarrota(Jugador jugador){

Si (jugador.dinero <= 0 && jugador.turnoJugador == 3) {

jugador.bancarrota= Verdadero;

JOptionPane.showMessageDialog(null, "El Jugador " + jugador.nombre + " ha caido en bancarrota", "Lamentablemente", JOptionPane.INFORMATION\_MESSAGE);

bancasrrotas= bancasrrotas + 1;

jugador.turnoJugador = 4;

}

Fin si

}

Metodo mover(){

Para (int i = 0 ; i<3 ; i++){

Si (jugadores[i].turnoJugador == 3 && jugadores[i].carcel == true && jugadores[i].turnosCarcel<4){

Metodo opcionesCarcel();

Si (jugadores[i].turnoJugador == 3 && jugadores[i].carcel == true && jugadores[i].turnosCarcel<4 && pagoFianza == false){

jugadores[i].turnoJugador = 0;

jugadores[i].turnosCarcel = jugadores[i].turnosCarcel + 1;

J1.turnoJugador = J1.turnoJugador + 1;

J2.turnoJugador = J2.turnoJugador + 1;

J3.turnoJugador = J3.turnoJugador + 1;

JOptionPane.showMessageDialog(null, "Se encuentra en la carcel", "Turno perdido para el jugador " + jugadores[i].getNombre(), JOptionPane.INFORMATION\_MESSAGE);

} Sino entonces (jugadores[i].turnoJugador == 3 && jugadores[i].carcel == true && jugadores[i].turnosCarcel>4){

jugadores[i].carcel = Falso;

jugadores[i].turnosCarcel = 0;

}

Fin si

}

Fin si

pagoFianza = falso;

}

Fin para

JOptionPane.showMessageDialog(null, "Se lanzaran los dados", "Turno de " + nombreJugadorTurno(), JOptionPane.INFORMATION\_MESSAGE);

Metodo lanzarDados();

Si (dado1 == dado2){

turnoDoble = Verdadero;

turnoDobleN = turnoDobleN + 1;

JOptionPane.showMessageDialog(null, "¡Salieron dobles!", "Resultado Dados", JOptionPane.INFORMATION\_MESSAGE);

} Sino{

turnoDoble = falso;

turnoDobleN = 0;

}

Fin si

Imprime ("Total dados: " + totaldados);

Si(J1.turnoJugador==3) {

x= jLabel1.getX();

y= jLabel1.getY();

Para (int movFal = 0; movFal < totaldados; movFal++){

J1.setPosicionJugador((J1.getPosicionJugador() + 1));

Si(J1.getPosicionJugador()>20){

J1.setPosicionJugador(1);

}

Fin si

xn = lista.buscarPosicionX(J1.getPosicionJugador(), xn, J1, J2, J3);

yn = lista.buscarPosicionY(J1.getPosicionJugador(), yn, J1, J2, J3);

}

Fin para

Imprime (J1.getPosicionJugador());

Metodo moverJugador(x, y, xn, yn);

}

Fin si

Si (J2.turnoJugador==3) {

x = jLabel2.getX();

y = jLabel2.getY();

Para (int movFal = 0; movFal < totaldados; movFal++){

J2.setPosicionJugador((J2.getPosicionJugador() + 1));

Si(J2.getPosicionJugador()>20){

J2.setPosicionJugador(1);

}

Fin si

xn= lista.buscarPosicionX(J2.getPosicionJugador(), xn, J1, J2, J3);

yn= lista.buscarPosicionY(J2.getPosicionJugador(), yn, J1, J2, J3);

}

Fin para

Imprime(J2.getPosicionJugador());

Metodo moverJugador(x, y, xn, yn);

}

Fin si

SI (J3.turnoJugador==3) {

x= jLabel4.getX();

y= jLabel4.getY();

Para (int movFal = 0; movFal < totaldados; movFal++){

J3.setPosicionJugador((J3.getPosicionJugador() + 1));

Si(J3.getPosicionJugador()>20){

J3.setPosicionJugador(1);

}

Fin si

xn= lista.buscarPosicionX(J3.getPosicionJugador(), xn, J1, J2, J3);

yn= lista.buscarPosicionY(J3.getPosicionJugador(), yn, J1, J2, J3);

}

Fin para

Fin Para(J3.getPosicionJugador());

Metodo moverJugador(x, y, xn, yn);

}

Fin si

}

Metodo moverJugador(int x, int y, int xn, int yn){

Si(J1.turnoJugador == 3){

jLabel1.setLocation(xn, yn);

}

Fin si

SI(J2.turnoJugador == 3){

jLabel2.setLocation(xn, yn);

}

Fin si

Si(J3.turnoJugador == 3){

jLabel4.setLocation(xn, yn);

}

Fin si

}

Metodo String nombreJugadorTurno(){

Para (int i=0; i<3; i++){

Si (jugadores[i].turnoJugador==3) {

nombreJT = jugadores[i].nombre;

}

Fin si

}

Fin para

retorna nombreJT;

}

Metodo main(String args[]) {

Intenta {

Para (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

Si ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

Fin caso;

}

Fin caso

}

Fin para

} Atrapa (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Tablero.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} Atrapa (InstantiationException ex) {

java.util.logging.Logger.getLogger(Tablero.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} Atrapa (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Tablero.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} Atrapa (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Tablero.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

java.awt.EventQueue.invokeLater(new Runnable() {

Metodo run() {

nuevo Tablero().setVisible(Visible);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton2;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel10;

public javax.swing.JLabel jLabel11;

private javax.swing.JLabel jLabel12;

private javax.swing.JLabel jLabel14;

public javax.swing.JLabel jLabel15;

private javax.swing.JLabel jLabel17;

public javax.swing.JLabel jLabel19;

public javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel6;

private javax.swing.JLabel jLabel7;

private javax.swing.JLabel jLabel8;

private javax.swing.JLabel jLabel9;

private javax.swing.JPanel jPanel2;

// End of variables declaration

// Mis declaraciones

int x;

int xn;

int y;

int yn;

Jugador J1 = nuevo Jugador("Ganso", 1);

Jugador J2 = nuevo Jugador("Oso", 2);

Jugador J3 = nuevo Jugador("Vaca", 3);

Jugador[] jugadores = nuevo Jugador[3];

boolean turnoDoble = falso;

boolean pagoFianza = falso;

int turnoDobleN = 0;

int bancasrrotas;

int dado1;

int dado2;

int totaldados;

String[] opciones = {"Pasar turno", "Comprar propiedad", "Edificar"};

String[] opcionesCarcel = {"Pasar turno", "Pagar fianza"};

String nombreJT;

Random random = nuevo Random();

int dJ1, dJ2, dJ3;

Scanner inFo = Nuevo Scanner(System.in);

//Declaracion de las casillas

Evento salida = nuevo Evento(1, "Salida", "Negro", 640, 560, 570, 560, 600, 640);

Propiedad puntoFijo = nuevo Propiedad(2, "Punto Fijo", "Morado",460, 560, 510, 590, 470, 650, 15000, 1500, 3500, 4000);

Compañia busMargarita = nuevo Compañia(3, "Bus Margarita", "Azul claro",360, 560, 410, 600, 360, 650, 38000, 450);

PropiedadExtendida caracas = nuevo PropiedadExtendida(4, "Caracas", "Amarillo", 260, 560, 300, 600, 250, 650, 25000, 2500, 3500, 5500, 4000, 6000, 9000);

Evento unimar = nuevo Evento(5, "Unimar", "Blanco",150, 560, 190, 600, 140, 650);

PropiedadExtendida nuevaEsparta = nuevo PropiedadExtendida(6, "Nueva Esparta", "Amarillo",0, 560, 70, 600,0, 650, 20000, 2000, 3000, 5000, 3500, 5000, 8000);

Compañia corpoelec = nuevo Compañia(7, "Corpoelec", "Azul claro", 0, 460, 80, 450, 40, 500, 35000, 500);

Evento carcel = nuevo Evento(8, "Carcel", "Gris", 0, 350, 90, 350, 40, 390);

PropiedadExtendida valencia = nuevo PropiedadExtendida(9, "Valencia", "Naranja", 0, 250, 100, 250, 40, 290, 18000, 1000, 1500, 2000, 2500, 3000, 3000);

Evento avanceDinamico = nuevo Evento(10, "Avance Dinamico", "Negro", 0, 150, 90, 150, 40, 190);

Compañia banesco = nuevo Compañia(11, "Banesco", "Azul claro", 0, 0, 80, 20, 10, 80, 40000, 600);

PropiedadExtendida merida = nuevo PropiedadExtendida(12, "Merida", "Naranja", 150, 0, 200, 30, 150, 90, 20000, 1300, 1800, 2200, 2900, 3500, 3500);

Evento impuestoLujo = nuevo Evento(13, "Impuesto al Lujo", "Rojo", 250, 0, 300, 30, 250, 90);

PropiedadExtendida maturin = new PropiedadExtendida(14, "Maturin", "Verde",360, 0, 400, 30, 360, 90, 23000, 2000, 3250, 4500, 3850, 5500, 7500);

Compañia hidrocaribe = new Compañia(15, "Hidrocaribe", "Azul claro",460, 0, 500, 30, 460, 90, 30000, 300);

Evento policia = nuevo Evento(16, "Policia", "Azul oscuro",560, 0, 650, 30, 570, 80);

PropiedadExtendida monagas = nuevo PropiedadExtendida(17, "Monagas", "Verde",560, 150, 660, 150, 610, 190, 23000, 2000, 3250, 4500, 3850, 5500, 7500);

Propiedad barinas = nuevo Propiedad(18, "Barinas", "Morado",560, 250, 660, 260, 610, 290, 17000, 1800, 3800, 4800);

Evento retrocesoDinamico = nuevo Evento(19, "Retroceso Dinamico", "Gris",560, 360, 660, 360, 600, 400);

Propiedad anzoategui = nuevo Propiedad(20, "Anzoategui", "Amarillo",560, 460, 660, 460, 600, 500, 25000, 2500, 4500, 7000);

//Instanciacion de la lista

Lista lista = nuevo Lista();

}

***Clase Nodo***

package monopolio;

Clase publica Nodo{

Metodo Casilla casillaNodo;

Metodo Nodo siguiente;

Metodo Nodo anterior;

}

***Clase Lista***

Clase Lista {

Metodo Nodo cabeza;

Metodo Nodo cola;

Metodo Lista() {

cabeza = null;

cola = null;

}

Metodo boolean estaVacia() {

retorna cabeza == null;

}

Metodo insertar(Casilla nuevaCasilla) {

Nodo nuevoNodo = nuevo Nodo();

nuevoNodo.casillaNodo = nuevaCasilla;

Si (estaVacia()) {

cabeza = nuevoNodo;

cola = nuevoNodo;

cabeza.siguiente = cola;

cola.anterior = cabeza;

} sino {

nuevoNodo.anterior = cola;

cola.siguiente = nuevoNodo;

cola = nuevoNodo;

cola.siguiente = cabeza;

cabeza.anterior = cola;

}

Fin si

}

Método eliminar(Casilla casillaBorrar) {

Si (!estaVacia()) {

Nodo nodoActual = cabeza;

Mientras (nodoActual != cola && nodoActual.casillaNodo != casillaBorrar) {

nodoActual = nodoActual.siguiente;

}

Fin mientras

si (nodoActual.casillaNodo == casillaBorrar) {

si (nodoActual == cabeza) {

cabeza = cabeza.siguiente;

cabeza.anterior = cola;

cola.siguiente = cabeza;

} sino entonces (nodoActual == cola) {

cola = cola.anterior;

cola.siguiente = cabeza;

cabeza.anterior = cola;

} sino {

nodoActual.anterior.siguiente = nodoActual.siguiente;

nodoActual.siguiente.anterior = nodoActual.anterior;

}

Fin si

}

Fin si

}

Fin si

}

Metodo imprimir() {

Si (!estaVacia()) {

Nodo nodoActual = cabeza;

haz {

imprime(nodoActual.casillaNodo.getAtributos());

nodoActual = nodoActual.siguiente;

} mientras (nodoActual != cabeza);

}Fin si

}

Metodo buscarPosicionX(int posicion, int xn, Jugador J1, Jugador J2, Jugador J3){

Nodo actual = nuevo Nodo();

actual = cola;

boolean encontrado = falso;

haz{

if(actual.casillaNodo.getPosicionCasilla() == posicion){

encontrado = Verdadero;

si(J2.turnoJugador == 3) xn = actual.casillaNodo.poj1x;

si(J1.turnoJugador == 3) xn = actual.casillaNodo.poj2x;

si(J3.turnoJugador == 3) xn = actual.casillaNodo.poj3x;

}

actual=actual.anterior;

} mientras (actual!=cola);

Si(encontrado==Verdadero){

//System.out.println("nodo encontrado");

} sino{

//System.out.println("no encontrado");

}

retorna xn;

}

Metodo buscarPosicionY(int posicion, int yn, Jugador J1, Jugador J2, Jugador J3){

Nodo actual = nuevo Nodo();

actual = cola;

boolean encontrado = falso;

Haz{

Si(actual.casillaNodo.getPosicionCasilla() == posicion){

encontrado = Verdadero;

Si(J2.turnoJugador == 3) yn = actual.casillaNodo.poj1y;

Si(J1.turnoJugador == 3) yn = actual.casillaNodo.poj2y;

Si(J3.turnoJugador == 3) yn = actual.casillaNodo.poj3y;

}

actual=actual.anterior;

} mientras (actual!=cola);

si(encontrado==true){

//System.out.println("nodo encontrado");

} sino{

//System.out.println("no encontrado");

}

retorna yn;

}

***Clase Jugador***

package monopolio;

Clase publica Jugador {

String nombre;

int id;

int vecesUnimar=0;

int dinero = 100000;

int posicionJugador=1;

int turnoJugador=0;

int cantidadPropiedades=0;

boolean bancarrota = falso;

boolean carcel = falso;

int turnosCarcel=0;

método Jugador(String nombre, int id){

this.nombre = nombre;

this.id = id;

}

Metodo setDinero(int dineroN){

this.dinero = dineroN;

}

Metodo setPosicionJugador(int posicionJugador){

this.posicionJugador = posicionJugador;

}

Metodo setCantidadPropiedades(int cantidadPropiedadesN){

this.cantidadPropiedades = cantidadPropiedadesN;

}

Metodo setBancarrota(boolean bancarrotaN){

this.bancarrota = bancarrotaN;

}

Metodo setCarcel(boolean carcelN){

this.carcel = carcelN;

}

Metodo setNombre(String nombreN){

this.nombre = nombreN;

}

Metodo String getNombre(){

Retorna this.nombre;

}

Metodo int getDinero(){

retorna this.dinero;

}

Metodo int getCantidadPropiedades(){

Retorna this.cantidadPropiedades;

}

Metodo boolean getBancarota(){

Retorna this.bancarrota;

}

Método int getPosicionJugador(){

retorna this.posicionJugador;

}

}

***Clase Casilla***

package monopolio;

Clase publica Casilla {

int posicionCasilla;

String nombreCasilla;

String colorCasilla;

int poj1x;

int poj1y;

int poj2x;

int poj2y;

int poj3x;

int poj3y;

metodo Casilla(int posicionCasilla, String nombreCasilla, String colorCasilla, int poj1x, int poj1y, int poj2x, int poj2y, int poj3x, int poj3y){

this.posicionCasilla = posicionCasilla;

this.nombreCasilla = nombreCasilla;

this.colorCasilla = colorCasilla;

this.poj1x = poj1x;

this.poj1y = poj1y;

this.poj2x = poj2x;

this.poj2y = poj2y;

this.poj3x = poj3x;

this.poj3y = poj3y;

}

Metodo String getNombreCasilla(){

retorna this.nombreCasilla;

}

Método int getPosicionCasilla(){

Retorna this.posicionCasilla;

}

Metodo String getColorCasilla() {

Retorna colorCasilla;

}

Método String getAtributos(){

Retorna "" +getNombreCasilla() + " " + getPosicionCasilla() + " " + getColorCasilla() +"/n";

}

}

***Clase Propiedad***

package monopolio;

Clase publica Propiedad extiende Casilla {

int precioPropiedad;

int alquilerInicial;

int alquiler1Casa;

int precioCasa;

boolean adquirida = falso;

metodo Propiedad (int posicionCasilla, String nombreCasilla, String colorCasilla, int poj1x, int poj1y,int poj2x, int poj2y, int poj3x, int poj3y, int precioPropiedad, int alquilerInicial, int alquiler1Casa, int precioCasa){

super (posicionCasilla, nombreCasilla, colorCasilla, poj1x, poj1y, poj2x, poj2y, poj3x, poj3y);

this.precioPropiedad = precioPropiedad;

this.alquilerInicial = alquilerInicial;

this.alquiler1Casa = alquiler1Casa;

this.precioCasa = precioCasa;

}

Método setAdquirida(boolean adquiridaN){

this.adquirida = adquiridaN;

}

Metodo int getPrecioPropiedad(){

retorna this.precioPropiedad;

}

Metodo int getAlquilerInicial(){

Retorna this.alquilerInicial;

}

Método getAlquiler1Casa(){

retorna this.alquiler1Casa;

}

metodo int getPrecioCasa(){

retorna this.precioCasa;

}

Metodo boolean getAdquirida(){

retorna this.adquirida;

}

}

***Clase PropiedadExtendida***

package monopolio;

Clase publica PropiedadExtendida extends Propiedad{

int alquiler2Casa;

int alquiler1Hotel;

int precioHotel;

metodo PropiedadExtendida(int posicionCasilla, String nombreCasilla, String colorCasilla, int poj1x, int poj1y, int poj2x, int poj2y, int poj3x, int poj3y,int precioPropiedad, int alquilerInicial, int alquiler1Casa, int precioCasa, int alquiler2Casa, int alquiler1Hotel, int precioHotel){

super(posicionCasilla, nombreCasilla, colorCasilla,poj1x, poj1y, poj2x, poj2y, poj3x, poj3y, precioPropiedad, alquilerInicial, alquiler1Casa, precioCasa);

this.alquiler2Casa = alquiler2Casa;

this.alquiler1Hotel = alquiler1Hotel;

this.precioHotel = precioHotel;

}

Metodo int getAlquiler2Casa(){

Retorna this.alquiler2Casa;

}

Metodo getAlquiler1Hotel(){

retorna this.alquiler1Hotel;

}

Metodo int getPrecioHotel(){

retorna this.precioHotel;

}

}

***Clase Compañia***

package monopolio;

Clase publica Compañia extiende Casilla{

int precioCompañia;

int cobro;

boolean adquirida = falso;

metodo Compañia(int posicionCasilla, String nombreCasilla, String colorCasilla,int poj1x, int poj1y, int poj2x, int poj2y, int poj3x, int poj3y, int precioCompañia, int cobro){

super(posicionCasilla, nombreCasilla, colorCasilla, poj1x, poj1y, poj2x, poj2y, poj3x, poj3y);

this.precioCompañia = precioCompañia;

this.cobro = cobro;

}

metodo int getPrecioCompañia(){

retorna this.precioCompañia;

}

Metodo int getCobro(){

retorna this.cobro;

}

Metodo boolean getAdquirida(){

retorna this.adquirida;

}

}

***Clase Evento***

package monopolio;

Clase publica Compañia extiende Casilla{

int precioCompañia;

int cobro;

boolean adquirida = falso;

Metodo Compañia(int posicionCasilla, String nombreCasilla, String colorCasilla,int poj1x, int poj1y, int poj2x, int poj2y, int poj3x, int poj3y, int precioCompañia, int cobro){

super(posicionCasilla, nombreCasilla, colorCasilla, poj1x, poj1y, poj2x, poj2y, poj3x, poj3y);

this.precioCompañia = precioCompañia;

this.cobro = cobro;

}

Metodo int getPrecioCompañia(){

retorna this.precioCompañia;

}

Metodo int getCobro(){

retorna this.cobro;

}

Metodo boolean getAdquirida(){

retorna this.adquirida;

}

}