Práctica 2 PDO | David Rodríguez Aparicio

En esta segunda práctica de PDO vamos a seguir desarrollando el juego Civitas, basado en el monopoly que todos conocemos.

Iremos comentando e incluyendo el código de las distintas clases que hemos creado en esta práctica.

Además de ello, aunque en esta documentación no lo encontremos se ha completado el diagrama UML, creando todas las clases, métodos y relaciones que aparecian en este.

En esta práctica no se ha llegado a programar un test de estas clases, pero para los siguientes avances haremos un testeo completo y en condiciones de todo.

Los códigos no tienen muchos comentarios, ya que la programación orientada a objetos de este tipo de programas suele ser muy clara y no requiere de ellos generalmente, aunque comentaremos las cosas más importante antes de mostrar estos códigos.

Si el profesor lo necesita, el progreso de la práctica está recopilado en mi GitHub, puede pedirlo si es necesario.

TituloPropiedad

Tiene un único constructor, el cuál es llamado cuando es necesario en la creación del tablero.

Maneja algunas de las funcionalidades de las Casillas que son propiedads (casas,hoteles,ventas,etc.).

```
package civitas;

public class TituloPropiedad {

    private float alquilerBase;
    private static final float factorInteresesHipoteca = (float) 1.1; /*  ?¿¿?¿?¿
*/
    private float factorRevalorizacion;
    private float hipotecaBase;
    private boolean hipotecado;
    private String nombre;
    private int numCasas;
    private int numHoteles;
    private float precioCompra;
    private float precioEdificar;

    private Jugador propietario;


    /* ---------------- CONSTRUCTOR ---------------- */

    TituloPropiedad(String nombre, float baseAlquiler, float
factorRevalorizacion,float baseHipoteca, float precioCompra, float precioEdificar)
```

```
{
        this.nombre=nombre;
        this.alquilerBase = baseAlquiler;
        this.factorRevalorizacion = factorRevalorizacion;
        this.hipotecaBase = baseHipoteca;
        this.precioCompra = precioCompra;
        this.precioEdificar = precioEdificar;

        this.propietario = null;
        this.numCasas = 0;
        this.numHoteles = 0;
        this.hipotecado = false;

    }



    /* ---------------- METODOS ---------------- */


    void actualizaPropietarioPorConversion(Jugador jugador) {
        this.propietario = jugador;
    }

    boolean cancelarHipoteca(Jugador jugador) {

        if(this.hipotecado && esEsteElPropietario(jugador) ){

            this.hipotecado = false;
            jugador.paga( getImporteCancelarHipoteca() );

            return true;
        }else{

            return false;
        }
    }

    int cantidadCasasHoteles() {
        return this.numCasas + this.numHoteles;
    }

    boolean comprar(Jugador jugador) {

        if(!this.tienePropietario() && jugador.getSaldo() >= this.precioEdificar){
            jugador.paga(this.precioCompra);
            this.propietario = jugador;

            return true;
        }

        return false;

    }
```

```
    boolean construirCasa(Jugador jugador) {

        boolean puedeConstruir = false;

        if(esEsteElPropietario(jugador) && jugador.getSaldo() >=
this.precioEdificar ){
            jugador.paga(this.precioEdificar);
            this.numCasas++;

            puedeConstruir=true;

            return puedeConstruir;
        }
        return puedeConstruir;

    }

    boolean construirHotel(Jugador jugador) {

        boolean puedeConstruir = false;

        if(esEsteElPropietario(jugador) && jugador.getSaldo() >=
this.precioEdificar){
            jugador.paga(this.precioEdificar);
            this.numHoteles++;

            puedeConstruir=true;

            return puedeConstruir;
        }
        return puedeConstruir;

    }

    boolean derruirCasas(int n, Jugador jugador) {

        if (esEsteElPropietario(jugador) && this.numCasas >= n) {
            this.numCasas = this.numCasas - n;
            return true;
        }

        return false;
    }

    private boolean esEsteElPropietario(Jugador jugador) {
        return (this.propietario == jugador ) ? true : false;
    }

    boolean hipotecar(Jugador jugador) {

        if( !this.hipotecado && esEsteElPropietario(jugador) ){

            this.hipotecado = true;
```

```
            jugador.recibe(this.hipotecaBase);

            return true;
        }else{

            return false;
        }

    }

    private boolean propietarioEncarcelado() {

        return ( this.propietario != null || this.propietario.encarcelado ) ? true
: false;

    }

    boolean tienePropietario() {
        return (this.propietario != null ) ? true : false;
    }

    public String toString() {
        String representacion;

        representacion =
                "Titulo Propiedad:\n" +
                "\t- Nombre = " + nombre + "\n" +
                "\t- Alquiler base = " + alquilerBase + "\n" +
                "\t- Factor de revalorización = " + factorRevalorizacion + "\n" +
                "\t- Hipoteca base = " + hipotecaBase + "\n" +
                "\t- Precio de compra = " + precioCompra + "\n" +
                "\t- Precio de edificar = " + precioEdificar + "\n" +
                "\t- Número de casas = " + numCasas + "\n" +
                "\t- Número de hoteles = " + numHoteles + "\n";

        if (propietario == null)
            representacion += "\t- Propietario = sin propietario\n";
        else
            representacion += "\t- Propietario = " + propietario.getNombre();

        return representacion;
    }

    void tramitarAlquiler(Jugador jugador) {

        if(!this.hipotecado && !esEsteElPropietario(jugador)){
            jugador.pagaAlquiler(this.alquilerBase);
            this.propietario.recibe(this.alquilerBase);
        }

    }

    boolean vender(Jugador jugador) {
```

```
        if(this.tienePropietario()){
            float devolver = this.precioCompra + ( (this.numCasas *
factorRevalorizacion * precioEdificar) + (this.numHoteles * factorRevalorizacion *
precioEdificar) );

            this.numCasas = 0;
            this.numHoteles = 0;

            jugador.recibe(devolver);
            this.propietario = null;

            return true;
        }

        return false;
    }

    /* ---------------- SETTERS / GETTERS ---------------- */

    public boolean getHipotecado() {
        return this.hipotecado;
    }

    float getImporteCancelarHipoteca() {

        float total = this.hipotecaBase * ( 1f + ( this.numCasas * 0.5f ) + (
this.numHoteles * 2.5f ) ) ;
        return  total * factorInteresesHipoteca;

    }

    private float getImporteHipoteca() {
        return this.hipotecaBase;
    }

    String getNombre() {
        return this.nombre;
    }

    int getNumCasas() {
        return this.numCasas;
    }

    int getNumHoteles() {
        return  this.numHoteles;
    }

    private float getPrecioAlquiler() {

        if( this.hipotecado || this.propietarioEncarcelado() ){
            return (float) 0;
        }else{

            float precioReal = this.alquilerBase * ( 1f + ( this.numCasas * 0.5f )
```

```
+ ( this.numHoteles * 2.5f ) );

            return precioReal;
        }

    }

    float getPrecioCompra() {
        return this.precioCompra;
    }

    float getPrecioEdificar() {
        return this.precioEdificar;
    }

    private float getPrecioVenta() {

        float total = this.precioCompra + this.cantidadCasasHoteles() *
precioEdificar;

        return total;
    }

    Jugador getPropietario() {
        return this.propietario;
    }
}
```

Sorpresa

Tiene varios constructores, que se usan dependiendo de donde se llama y de lo que se necesite al momento de usarlo.

Funciona mediante un método (aplicarJugador) con un switch que decide el método a usar en cada caso, también incluye métodos para verificar si el jugador es correcto, salir del mazo y si una carta sorpresa ha sido usada.

```
package civitas;


class Sorpresa{
    private String texto;
    private int valor;

    TipoSorpresa tipo;
    MazoSorpresa mazo;
    Tablero tablero;
```

```
    /* ----------------- CONSTRUCTORES ----------------- */

    Sorpresa(TipoSorpresa tipo, Tablero tablero) {
        this.init();
        this.tipo = tipo;
        this.tablero = tablero;

        this.texto = "";
        this.valor = -1;
        this.mazo = null;
    }

    Sorpresa(TipoSorpresa tipo, Tablero tablero, int valor, String texto) {
        this.init();
        this.tipo = tipo;
        this.tablero = tablero;
        this.valor = valor;
        this.texto = texto;

        this.mazo = null;
    }

    Sorpresa(TipoSorpresa tipo, int valor, String texto) {
        this.init();
        this.tipo = tipo;
        this.valor = valor;
        this.texto = texto;

        this.tablero = null;
        this.mazo = null;
    }

    Sorpresa(TipoSorpresa tipo, MazoSorpresa mazo) {
        this.init();
        this.tipo = tipo;
        this.mazo = mazo;

        this.texto = "";
        this.valor = -1;
        this.tablero = null;
    }



    /* ----------------- METODOS ----------------- */

    void aplicarJugador(int actual, Jugador[] Jugadores){

        /* Hay 6 valores posibles, Casilla | Carcel | PagarCobrar | PorCasaHotel |
PorJugador | Salir Carcel */
        switch (this.tipo) {
            case IRCARCEL -> this.aplicarJugador_irCarcel(actual, Jugadores);
```

```
            case IRCASILLA -> this.aplicarJugador_irCasilla(actual, Jugadores);
            case PAGARCOBRAR -> this.aplicarJugador_pagarCobrar(actual,
Jugadores);
            case PORCASAHOTEL -> this.aplicarJugador_porCasaHotel(actual,
Jugadores);
            case PORJUGADOR -> this.aplicarJugador_porJugador(actual, Jugadores);
            case SALIRCARCEL -> this.aplicarJugador_salirCarcel(actual,
Jugadores);
        }
    }

    private void aplicarJugador_irCasilla(int actual, Jugador[] Jugadores){
        if(jugadorCorrecto(actual, Jugadores)){
            informe(actual, Jugadores);
            Diario.getInstance().ocurreEvento("Se ha usado irCasilla en: " +
Jugadores[actual].getNombre() );


            int casillaActual = Jugadores[actual].getNumCasillaActual();

            int tirada = tablero.calcularTirada(casillaActual, this.valor); /*
Falta añadir el numero del dado. ?¿ */

            int nuevaPosicion = tablero.nuevaPosicion(casillaActual, tirada);

            Jugadores[actual].moverACasilla(nuevaPosicion);

            tablero.getCasilla(nuevaPosicion).recibeJugador(actual, Jugadores);
        }
    }

    private void aplicarJugador_irCarcel(int actual, Jugador[] Jugadores){
        if(jugadorCorrecto(actual, Jugadores)){
            Jugadores[actual].encarcelar( tablero.getCarcel() );/*   Casilla de la
carcel  */
            Diario.getInstance().ocurreEvento("Se ha encarcelado al jugador: " +
Jugadores[actual].getNombre() );
        }
    }

    private void aplicarJugador_pagarCobrar(int actual, Jugador[] Jugadores){
        if(jugadorCorrecto(actual, Jugadores)){
            informe(actual, Jugadores);
            Diario.getInstance().ocurreEvento("Se ha usado pagarCobrar en: " +
Jugadores[actual].getNombre() );
            Jugadores[actual].modificarSaldo(this.valor);

        }
    }

    private void aplicarJugador_porCasaHotel(int actual, Jugador[] Jugadores){

        if (jugadorCorrecto(actual, Jugadores)) {
            informe(actual, Jugadores);
```

```
            Diario.getInstance().ocurreEvento("Se ha usado porCasaHotel en: " +
Jugadores[actual].getNombre() );

            int numCasasHoteles = Jugadores[actual].cantidadCasasHoteles();

            int cantidad = valor * numCasasHoteles;

            Jugadores[actual].modificarSaldo(cantidad);
    }
}

    private void aplicarJugador_porJugador(int actual, Jugador[] Jugadores){

        if (jugadorCorrecto(actual, Jugadores)) {

            informe(actual, Jugadores);
            Diario.getInstance().ocurreEvento("Se ha usado porJugador en: " +
Jugadores[actual].getNombre() );


            int numJugadores = Jugadores.length;

            Sorpresa paga = new Sorpresa(TipoSorpresa.PAGARCOBRAR, -valor, "Paga
al jugador " + Jugadores[actual].getNombre());

            for (int i = 0; i < numJugadores; i++) {
                if ( i != actual ) {
                    paga.aplicarJugador(i, Jugadores);
                }
            }

            Sorpresa cobra = new Sorpresa(TipoSorpresa.PAGARCOBRAR,valor *
(numJugadores - 1),"Cobra del resto de Jugadores");

            cobra.aplicarJugador(actual, Jugadores);
        }
    }


    private void aplicarJugador_salirCarcel(int actual, Jugador[] Jugadores){
        if (jugadorCorrecto(actual, Jugadores)) {

            informe(actual, Jugadores);
            Diario.getInstance().ocurreEvento("Se ha usado salirCarcel en: " +
Jugadores[actual].getNombre() );


            boolean alguienTieneSalvoconducto = false;

            for (int i = 0; i < Jugadores.length && !alguienTieneSalvoconducto;
i++) {
                if (Jugadores[i].tieneSalvoconducto()) {
                    alguienTieneSalvoconducto = true;
                }
```

```
            }

            if (!alguienTieneSalvoconducto) {
                Jugadores[actual].obtenerSalvoconducto(this);
                salirDelMazo();
            }
        }
    }

    private void informe(int actual, Jugador[] Jugadores){
        Diario.getInstance().ocurreEvento( "Se le está aplicando una sorpresa a: "
+ Jugadores[actual] );
    }

    public boolean jugadorCorrecto(int actual, Jugador[] Jugadores){
        return (actual >= 0 && actual < Jugadores.length);
    }

    void salirDelMazo(){
        if (tipo == TipoSorpresa.SALIRCARCEL && mazo != null) {
            mazo.inhabilitarCartaEspecial(this);
        }
    }

    @Override
    public String toString() {
        return this.texto;
    }

    void usada(){
        if (tipo == TipoSorpresa.SALIRCARCEL && mazo != null) {
                mazo.habilitarCartaEspecial(this);
        }
    }

    private void init(){
        this.valor = -1;
    }



}
```

Casilla

Sus distintos constructores se usan en base a que tipo de casilla sea al crear el tablero (ya sea Juez,Impuesto,Propiedad,etc.).

Tiene los métodos a usar cuando un jugador caiga en esta casilla.

```java
package civitas;

public class Casilla {


    private int carcel;                     /* DUDA: se podría poner como tipo
casilla ??  */
    private float importe;
    private String nombre;

    TipoCasilla tipo;
    TituloPropiedad tituloPropiedad;    /*  tituloPropiedad tipo="CALLE"  ?¿?¿? */
    Sorpresa sorpresa;                  /* tipo = "SORPRESA" */
    MazoSorpresa mazo;                  /* tipo = "SORPRESA" */



    /* ----------------- Constructores ----------------- */

    /* REVISAR CONSTRUCTORES */

    Casilla(String nombre) {
        this.init();
        this.nombre = nombre;

    }

    Casilla(TituloPropiedad titulo){
        this.init();

        this.tituloPropiedad = titulo;
        this.nombre = titulo.getNombre();
        this.tipo = TipoCasilla.CALLE;

    }

    Casilla(float cantidad,String nombre){
        this.init();

        this.importe = cantidad;
        this.nombre = nombre;
        this.tipo = TipoCasilla.IMPUESTO;

    }

    Casilla(int numCasillaCarcel, String nombre){
        this.init();

        this.carcel = numCasillaCarcel;
        this.nombre = nombre;
        this.tipo = TipoCasilla.JUEZ;
```

```
    }

    Casilla(MazoSorpresa mazo, String nombre){
        this.init();

        this.mazo = mazo;
        this.nombre = nombre;
        this.tipo = TipoCasilla.SORPRESA;
        // this.sorpresa = mazo.siguiente();


    }









    /* ----------------- METODOS ----------------- */

    private void informe(int iActual, Jugador[] Jugadores){
        Diario.getInstance().ocurreEvento("El jugador " +
Jugadores[iActual].getNombre() + " ha caído en la casilla: " + toString());
    }


    private void init(){
        this.nombre = "";
        this.importe = 0.0f;
        this.carcel = -1;
        this.tipo = TipoCasilla.DESCANSO;
        this.tituloPropiedad = null;
        this.sorpresa = null;
        this.mazo = null;
    }

    public boolean jugadorCorrecto(int iActual,  Jugador[] Jugadores){
        return iActual >= 0 && iActual < Jugadores.length;  /* Ya existe una de
estas en:  Sopresa | ?¿?¿ */
    }

    /* Siguiente Práctica */
    void recibeJugador(int iActual, Jugador[] Jugadores){

        /* Se debería poner en la siguiente práctica  */
        /* switch (this.tipo) {

            case CALLE -> this.recibeJugador_calle(iActual, Jugadores);
            case IMPUESTO -> this.recibeJugador_impuesto(iActual, Jugadores);
            case JUEZ -> this.recibeJugador_juez(iActual, Jugadores);
            case SORPRESA -> this.recibeJugador_sorpresa(iActual, Jugadores);
            case DESCANSO -> {
```

```
                if (jugadorCorrecto(iActual, Jugadores))
                    informe(iActual, Jugadores);
            }
        } */
    }

    /* Siguiente Práctica */
    private void recibeJugador_calle(int iActual, Jugador[] Jugadores){
        /* if (jugadorCorrecto(iActual, Jugadores)) {
            informe(iActual, Jugadores);
            Diario.getInstance().ocurreEvento("El jugador " +
Jugadores[iActual].getNombre() + " ha recibido: " +
this.tituloPropiedad.getNombre()  );
            tituloPropiedad.tramitarAlquiler(Jugadores[iActual]);
        } */
    }

    private void recibeJugador_impuesto(int iActual, Jugador[] Jugadores){
        if (jugadorCorrecto(iActual, Jugadores)) {
            informe(iActual, Jugadores);
            Diario.getInstance().ocurreEvento("El jugador " +
Jugadores[iActual].getNombre() + " ha caido en: " + this.nombre );

            Jugadores[iActual].pagaImpuesto(importe);
        }
    }

    private void recibeJugador_juez(int iActual, Jugador[] Jugadores){
        if (jugadorCorrecto(iActual, Jugadores)) {
            informe(iActual, Jugadores);
            Jugadores[iActual].encarcelar(carcel);
        }
    }

    /* Siguiente Práctica */
    private void recibeJugador_sorpresa(int iActual, Jugador[] Jugadores){
        /* if (jugadorCorrecto(iActual, Jugadores)) {
            informe(iActual, Jugadores);
            sorpresa = mazo.siguiente();
            sorpresa.aplicarJugador(iActual, Jugadores);
        } */
    }

    @Override
    public String toString(){
        String info = "Casilla{" +
                      "nombre='" + this.nombre + '\'' +
                      ", tipo=" + this.tipo;

            switch (this.tipo) {
                case CALLE -> info += ", titulo=" + tituloPropiedad.getNombre();
                case IMPUESTO -> info += ", importe=" + this.importe;
                case JUEZ -> info += ", carcel=" + this.carcel;
                case SORPRESA -> info += ", mazoSorpresas";
```

```
                case DESCANSO -> info += ", casilla normal";
            }

            info += "}";
            return info;
    }


    /* ----------------- GETTERS / SETTERS ----------------- */

    public String getNombre() {
        return this.nombre;
    }

    TituloPropiedad getTituloPropiedad(){
        return this.tituloPropiedad;
    }

}
```

Jugador

Tiene un constructor que se autoinicia solo con el nombre, y otro constructor de copia.

Contiene todos los métodos relacionados con el jugador,dinero de este y la gestión de sus propiedades.

```
package civitas;

import java.util.ArrayList;

@SuppressWarnings("all")

public class Jugador implements Comparable<Jugador>{

    protected static final int CasasMax = 4;
    protected static final int CasasPorHotel = 4;
    protected static final int HotelesMax = 4;

    protected static final float PasoPorSalida = 1000;
    protected static final float PrecioLibertad = 200;
    private static final float SaldoInicial = 7500;

    protected boolean encarcelado;

    private String nombre;
    private int numCasillaActual;
```

```
    private boolean puedeComprar;
    private float saldo;



    private ArrayList<TituloPropiedad> propiedades;
    private Sorpresa salvoconducto; /* Tipo = SalirCarcel */


    /* ---------------- CONSTRUCTORES ---------------- */

    Jugador(String nombre) {
        this.encarcelado = false;
        this.nombre = nombre;
        this.numCasillaActual = 0;
        this.saldo = SaldoInicial;
        this.salvoconducto = null;
        this.propiedades = null;
    }

    protected Jugador(Jugador otro) {
        this.encarcelado = otro.encarcelado;
        this.nombre = otro.nombre;
        this.numCasillaActual = otro.numCasillaActual;
        this.saldo = otro.saldo;
        this.salvoconducto = otro.salvoconducto;
        this.propiedades = otro.propiedades;
    }


    /* ---------------- METODOS ---------------- */

    /* Siguiente Práctica */
    boolean cancelarHipoteca(int ip) {

        return  true;
    }

    int cantidadCasasHoteles() {

        int total = 0;

        for (TituloPropiedad propiedad : propiedades) {
            total += propiedad.getNumCasas() + propiedad.getNumHoteles();
        }

        return total;
    }

    @Override
    public int compareTo(Jugador otro) {
        return Float.compare(this.saldo, otro.saldo);
    }
```

```
    /* Próxima Práctica */
    /* boolean comprar(TituloPropiedad titulo) {

    } */

    /* Próxima Práctica */
    boolean construirCasa(int ip) {

        return true;
    }

    /* Próxima Práctica */
    boolean construirHotel(int ip) {

        return true;
    }

    protected boolean debeSerEncarcelado() {
        if ( !this.encarcelado) {
            Diario.getInstance().ocurreEvento("El jugador " + this.nombre + "ha
caído en la carcel.");
            return true;
        }else if( this.salvoconducto == null ){
            perderSalvoconducto();
            Diario.getInstance().ocurreEvento("El jugador " + this.nombre + "se ha
salvado de la carcel por salvaconducto.");
            return false;
        }
        return false;
    }

    boolean enBancarrota() {
        return this.saldo < 0;
    }

    boolean encarcelar(int numCasillaCarcel) {
        if( this.debeSerEncarcelado() ){
            this.moverACasilla(numCasillaCarcel);
            this.encarcelado = true;
            Diario.getInstance().ocurreEvento("El jugador " + this.nombre + "ha
sido encarcelado");
        }
        return this.encarcelado;
    }

    private boolean existeLaPropiedad(int ip) {
        return (this.propiedades.get(ip) != null);
    }

    /* Próxima Práctica */
    boolean hipotecar(int ip) {

        return true;
```

```
    }

    boolean modificarSaldo(float cantidad) {
        this.saldo = this.saldo + cantidad;
        Diario.getInstance().ocurreEvento("Se le han aplicado a " + this.nombre +
", " + cantidad + " a su saldo.");
        return true;
    }

    boolean moverACasilla(int numCasilla) {

        if (this.encarcelado){
            return false;
        }

        this.numCasillaActual = numCasilla;
        this.puedeComprar = false;
        Diario.getInstance().ocurreEvento( "El jugador " + this.nombre + " se
mueve a la casilla " + this.numCasillaActual);

        return true;
    }

    boolean obtenerSalvoconducto(Sorpresa sorpresa) {
        if(!this.encarcelado){
            this.salvoconducto = sorpresa ; /* Hacer referencia a una sorpresa
tipo SalirCarcel */
            return true;
        }
        return false;
    }

    boolean paga(float cantidad) {
        return this.modificarSaldo(-1 * cantidad);
    }

    boolean pagaAlquiler(float cantidad) {
        if (this.encarcelado){
            return false;
        }
        return paga(cantidad);
    }

    boolean pagaImpuesto(float cantidad) {
        if(this.encarcelado){
            return false;
        }else{
            return this.paga(cantidad);
        }
    }

    boolean pasaPorSalida() {
        modificarSaldo(PasoPorSalida);
```

```
        Diario.getInstance().ocurreEvento("El jugador " + this.nombre + " ha
pasado por la salida y recibe " + PasoPorSalida );

        return true;
    }

    private void perderSalvoconducto() {
        this.salvoconducto.usada();
        this.salvoconducto = null;
    }

    boolean puedeComprarCasilla() {
        this.puedeComprar = !this.encarcelado;
        return this.puedeComprar;
    }

    private boolean puedeSalirCarcelPagando() {
        return this.saldo >= PrecioLibertad;
    }

    private boolean puedeEdificarCasa(TituloPropiedad propiedad) {
        return !(this.encarcelado || !puedoGastar(propiedad.getPrecioEdificar())
|| propiedad.getNumCasas() >= CasasMax);

    }

    private boolean puedeEdificarHotel(TituloPropiedad propiedad) {
        return !(this.encarcelado || !puedoGastar(propiedad.getPrecioEdificar())
|| propiedad.getNumCasas() < CasasPorHotel || propiedad.getNumHoteles() >=
HotelesMax );
    }

    private boolean puedoGastar(float precio) {
        if (this.encarcelado){
            return false;
        }
        return this.saldo >= precio;
    }

    boolean recibe(float cantidad) {
        if (this.encarcelado){
            return false;
        }

        return modificarSaldo(cantidad);
    }

    boolean salirCarcelPagando() {
        if (!this.encarcelado) return false;
        if (!puedeSalirCarcelPagando()) return false;

        paga(PrecioLibertad);
        this.encarcelado = false;
```

```
        Diario.getInstance().ocurreEvento("El jugador " + this.nombre + " ha
salido de la cárcel pagando " + PrecioLibertad);

        return true;
    }

    boolean salirCarcelTirando() {
        if (!this.encarcelado) return false;

        boolean sale = Dado.getInstance().salgoDeLaCarcel();

        if (sale) {
            this.encarcelado = false;

            Diario.getInstance().ocurreEvento("El jugador " + this.nombre + " ha
salido de la cárcel tirando el dado y sacando un " +
Dado.getInstance().getNumSalirCarcel());
        }

        return sale;
    }

    boolean tieneAlgoQueGestionar() {
        return !propiedades.isEmpty();
    }

    boolean tieneSalvoconducto() {
        return this.salvoconducto != null;
    }


    @Override
    public String toString() {

        String estado = this.encarcelado ? "ENCARCELADO" : "LIBRE";

        String texto =
            "Jugador: " + this.nombre +
            "\n  Estado: " + estado +
            "\n  Casilla actual: " + this.numCasillaActual +
            "\n  Saldo: " + this.saldo +
            "\n  Puede comprar: " + this.puedeComprar +
            "\n  Salvoconducto: " + (this.salvoconducto != null ? "SI" : "NO") +
            "\n  Propiedades:";

        for (TituloPropiedad propiedad : propiedades) {
            texto += "\n    - " + propiedad.getNombre() +
                    " (Casas: " + propiedad.getNumCasas() +
                    ", Hoteles: " + propiedad.getNumHoteles() + ")";
        }

        return texto;
    }
```

```
    boolean vender(int ip) {
        if (this.encarcelado){
            return false;
        }else if (!existeLaPropiedad(ip)){
            return false;
        }else{
            TituloPropiedad propiedad = propiedades.get(ip);

            boolean vendido = propiedad.vender(this);

            if (vendido) {
                propiedades.remove(ip);
                Diario.getInstance().ocurreEvento("El jugador " + nombre + " ha
vendido la propiedad " + propiedad.getNombre());
            }

            return vendido;
        }
    }


    /* ---------------- SETTERS / GETTERS ---------------- */

    private int getCasasMax() {
        return CasasMax;
    }

    int getCasasPorHotel() {
        return CasasPorHotel;
    }

    private int getHotelesMax() {
        return HotelesMax;
    }

    protected String getNombre() {
        return this.nombre;
    }

    int getNumCasillaActual() {
        return this.numCasillaActual;
    }

    private float getPrecioLibertad() {
        return PrecioLibertad;
    }

    private float getPremioPasoSalida() {
        return PasoPorSalida;
    }

    protected ArrayList<TituloPropiedad> getPropiedades() {
        return this.propiedades;
```

```
    }

    boolean getPuedeComprar() {
        return this.puedeComprar;
    }

    protected float getSaldo() {
        return this.saldo;
    }

    public boolean isEncarcelado() {
        return this.encarcelado;
    }

}
```

CivitasJuego

Tiene un constructor que se inicia con la lista de nombres de jugadores.

Contiene la base del juego, métodos sobre gestión de propiedades, que se ayuda de otras clases, avances de jugador, ranking, turnos, información y algunos de los más importantes, creacion de Tablero y creación de Mazo Sorpresa.

```
package civitas;

import java.util.ArrayList;
import java.util.Arrays;
// import java.util.Collections;

public class CivitasJuego {

    private int indiceJugadorActual;

    private Jugador jugadores[]; /* Tiene que ser un array con todos los jugadores
*/
    private EstadosJuego estado;
    private GestorEstados gestorEstados;
    private Tablero tablero;
    OperacionesJuego tipoOperacion;
    private static final Dado dado = Dado.getInstance();
    private MazoSorpresa mazo;


    /* ---------------- CONSTRUCTOR ---------------- */

    CivitasJuego(ArrayList<String> nombres) {
```

```
        this.jugadores = new Jugador[nombres.size()];

        for (int i = 0; i < nombres.size(); i++) {
            this.jugadores[i] = new Jugador(nombres.get(i));
        }

        this.gestorEstados = new GestorEstados();
        this.estado = gestorEstados.estadoInicial();

        this.indiceJugadorActual = dado.quienEmpieza(this.jugadores.length);

        this.mazo = new MazoSorpresa();

        this.inicializarTablero(this.mazo); /* Ahora mismo tablero es null */

        this.inicializarMazoSorpresa(this.tablero);
    }




    /* ---------------- METODOS ---------------- */

    void actualizarInfo() {
        Jugador jugadorActual = jugadores[indiceJugadorActual];

        System.out.println("=== INFORMACIÓN DEL JUGADOR ACTUAL ===");
        System.out.println(jugadorActual.toString());

        System.out.println("\n=== CASILLA ACTUAL ===");
        int pos = jugadorActual.getNumCasillaActual();
        System.out.println(tablero.getCasilla(pos).toString());

        if (jugadorActual.enBancarrota()) {
            System.out.println("\n*** EL JUGADOR HA CAÍDO EN BANCARROTA ***");
            System.out.println("=== RANKING FINAL ===");
            System.out.println(ranking());   // lo implementas más adelante
        }
    }


    /* SIGUIENTE PRACTICA */
    private void avanzaJugador(){

    }

    public boolean hipotecar(int ip){
        Jugador jugadorActual = this.jugadores[indiceJugadorActual];
        return jugadorActual.hipotecar(ip);
    }

    public boolean cancelarHipoteca(int ip){
        Jugador jugadorActual = this.jugadores[indiceJugadorActual];
```

```
        return jugadorActual.cancelarHipoteca(ip);
    }

    /* SIGUIENTE PRACTICA */
    public boolean comprar(){

        return true;
    }

    public boolean comprarCasa(int ip){
        Jugador jugadorActual = this.jugadores[indiceJugadorActual];

        return jugadorActual.construirCasa(ip);
    }

    public boolean comprarHotel(int ip){
        Jugador jugadorActual = this.jugadores[indiceJugadorActual];

        return jugadorActual.construirHotel(ip);
    }

    private void contabilizarPasosPorSalida(Jugador jugadorActual){
        while (this.tablero.getPorSalida() > 0) {
            jugadorActual.pasaPorSalida();
        }
    }

    public boolean finalDelJuego(){
        for (Jugador jugador : this.jugadores) {
                if (jugador.enBancarrota()) {
                    return true;
                }
            }
        return false;
    }

    public String infoJugadorTexto(){
        return this.jugadores[indiceJugadorActual].toString();
    }

    /* REVISAR + Ver Reglas */
    private void inicializarMazoSorpresa(Tablero tablero) {

        if (this.mazo == null) this.mazo = new MazoSorpresa();

        this.mazo.alMazo(new
Sorpresa(TipoSorpresa.IRCARCEL,tablero,tablero.getCarcel(),"Vas directo a la
cárcel"));
        this.mazo.alMazo(new Sorpresa(TipoSorpresa.IRCASILLA,tablero,3,"Te mueves
a la casilla 3"));

        this.mazo.alMazo(new Sorpresa(TipoSorpresa.PAGARCOBRAR,-200,"Pagas 200"));
        this.mazo.alMazo(new Sorpresa(TipoSorpresa.PAGARCOBRAR,200,"Cobras 200"));
```

```
        this.mazo.alMazo(new Sorpresa(TipoSorpresa.PORCASAHOTEL,50,"Cobras 50 por
cada casa y hotel"));
        this.mazo.alMazo(new Sorpresa(TipoSorpresa.PORJUGADOR,100,"Cada jugador te
paga 100"));

        this.mazo.alMazo(new Sorpresa(TipoSorpresa.SALIRCARCEL,this.mazo));
    }

    private void inicializarTablero(MazoSorpresa mazo){
        int numCasillaCarcel = 5;
        this.tablero = new Tablero(numCasillaCarcel);


        /* TABLERO PROVISIONAL */

        /* 1 */tablero.añadeCasilla(new Casilla("SALIDA"));
        /* 2 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle1",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 3 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle2",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 4 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle3",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 5 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle4",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 6 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle5",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 7 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle6",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 8 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle7",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 9 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle8",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 10 */tablero.añadeCasilla(new Casilla(15,"JUEZ"));
        /* 11 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle11",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 12 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle12",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 13 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle13",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 14 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle14",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 15 */tablero.añadeCasilla(new Casilla("CARCEL"));
        /* 16 */tablero.añadeCasilla(new Casilla(new MazoSorpresa(),"SORPRESA"));
        /* 17 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle17",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 18 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle18",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 19 */tablero.añadeCasilla(new Casilla(new TituloPropiedad("calle19",
10.00f,  1.1f, 100.00f,  500.00f,  250.00f )));
        /* 20 */tablero.añadeCasilla(new Casilla((float) 250.00,"IMPUESTO"));


    }
```

```
    private void pasarTurno(){
        this.indiceJugadorActual = (this.indiceJugadorActual + 1) %
jugadores.length;
    }

    private Jugador[] ranking() {

        Jugador[] copiaJugadores = jugadores.clone(); /* Con .clone() se copia el
array. */

        Arrays.sort(copiaJugadores); /* Se supone que ordena por saldo gracias a
compareTo, ya que sobreescribe un método existente. */

        return copiaJugadores;
    }

    public boolean salirCarcelPagando(){
        Jugador jugadorActual = this.jugadores[indiceJugadorActual];
        return jugadorActual.salirCarcelPagando();
    }

    public boolean salirCarcelTirando(){
        Jugador jugadorActual = this.jugadores[indiceJugadorActual];
        return jugadorActual.salirCarcelTirando();
    }

    /* SIGUIENTE PRACTICA */
    /* public OperacionesJuego siguientePaso(){

    } */

    public void siguientePasoCompletado(OperacionesJuego operacion){
        Jugador jugadorActual = this.jugadores[indiceJugadorActual];

        this.estado = gestorEstados.siguienteEstado(jugadorActual, this.estado,
operacion);
    }

    public boolean vender(int ip){
        Jugador jugadorActual = this.jugadores[indiceJugadorActual];
        return jugadorActual.vender(ip);
    }




    /* ---------------- SETTERS / GETTERS ---------------- */

    public Casilla getCasillaActual(){
        return this.tablero.getCasilla(this.indiceJugadorActual);
    }

    public Jugador getJugadorActual(){
```

```
            return this.jugadores[this.indiceJugadorActual];
        }

    }
```