**B1) v15-part7.js → mortgage(tileIndex) → rama Estado: checks**

js

CopiarEditar

// Hipoteca especial del Estado

if (t.owner === 'E') {

if (t.mortgaged) { log('Ya está hipotecada.'); return; }

if (t.houses > 0 || t.hotel) { log('No puedes hipotecar con casas/hotel.'); return; }

**B2) v15-part7.js → mortgage(tileIndex) → rama Estado: efecto**

js

CopiarEditar

const mv = Math.max(0, Math.round((t.price || 0) \* 0.5));

t.mortgaged = true;

Estado.money = Math.round((Estado.money || 0) + mv); // “imprimir dinero”

log(`🏦 Estado hipoteca ${t.name} e imprime ${fmtMoney(mv)}. Mientras esté hipotecada, no cobra alquiler.`);

BoardUI.refreshTiles?.(); renderPlayers?.();

return;

}

**C1) v15-part7.js → unmortgage(tileIndex) → rama Estado: checks y cálculos**

js

CopiarEditar

// Deshipoteca especial del Estado

if (t.owner === 'E') {

if (!t.mortgaged) { log('No está hipotecada.'); return; }

const mv = Math.max(0, Math.round((t.price || 0) \* 0.5));

const fee = Math.max(0, Math.round(mv \* 0.10)); // 10% a repartir

if ((Estado.money || 0) < fee) {

log(`El Estado no tiene saldo para la deshipoteca (faltan ${fmtMoney(fee - (Estado.money || 0))}).`);

return;

}

**C2) v15-part7.js → unmortgage(tileIndex) → repartir y cerrar**

js

CopiarEditar

Estado.money = Math.max(0, Math.round((Estado.money || 0) - fee));

const vivos = (state.players || []).filter(p => p.alive);

if (vivos.length) {

const base = Math.floor(fee / vivos.length);

let resto = fee - base \* vivos.length;

vivos.forEach((p,i) => {

const extra = i < resto ? 1 : 0;

const amt = base + extra;

if (amt > 0) giveMoney(p, amt, { taxable:false, reason:`Reparto por deshipoteca del Estado: ${t.name}` });

});

}

t.mortgaged = false;

log(`🏛️ Estado deshipoteca ${t.name}. Reparte ${fmtMoney(fee)} entre los jugadores.`);

BoardUI.refreshTiles?.(); renderPlayers?.();

return;

}