```
public class Carta {
	private String palo;
	private String valor;
	private boolean bocaArriba;

	public Carta(String palo, String valor) {
	this.palo = palo;
	this.valor = valor;
	this.bocaArriba = false;
	}

	public String getPalo() {
	return palo;
	}

	public String getValor() {
	return valor;
	}

	public boolean isBocaArriba() {
	return bocaArriba;
	}

	public void voltear() {
	this.bocaArriba = !this.bocaArriba;
	}

	@Override
	public String toString() {
	return bocaArriba ? valor + " de " + palo : "[X]";
	}
}
```

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class Baraja {
	private List<Carta> cartas;

	public Baraja() {
	cartas = new ArrayList<>();
	String[] palos = {"Oros", "Copas", "Espadas", "Bastos"};
	String[] valores = {"As", "Dos", "Tres", "Cuatro", "Cinco", "Seis", "Siete", "Sota",
"Caballo", "Rey"};

	for (String palo : palos) {
	for (String valor : valores) {
		cartas.add(new Carta(palo, valor));
	}
	}
	}

	public void barajar() {
	Collections.shuffle(cartas);
	}

	public Carta sacarCarta() {
	return cartas.isEmpty() ? null : cartas.remove(0);
	}

	public boolean estaVacia() {
	return cartas.isEmpty();
	}
}
```

```
import java.util.*;

public class Solitario {
	private Baraja baraja;
	private List<Stack<Carta>> pilas;
	private Stack<Carta> descarte;

	public Solitario() {
	baraja = new Baraja();
	pilas = new ArrayList<>();
	descarte = new Stack<>();
	for (int i = 0; i < 7; i++) {
	pilas.add(new Stack<>());
	}
	}

	public void inicializarJuego() {
	baraja.barajar();

	// Distribuir cartas en las pilas iniciales
	for (int i = 0; i < 7; i++) {
	for (int j = 0; j <= i; j++) {
		Carta carta = baraja.sacarCarta();
		if (j == i) {
		carta.voltear(); // Última carta boca arriba
		}
		pilas.get(i).push(carta);
	}
	}
	}

	public void mostrarTablero() {
	System.out.println("\nEstado del tablero:");
	for (int i = 0; i < pilas.size(); i++) {
	System.out.print("Pila " + (i + 1) + ": ");
	if (pilas.get(i).isEmpty()) {
		System.out.println("(vacía)");
	} else {
		for (Carta carta : pilas.get(i)) {
		System.out.print(carta + " ");
		}
		System.out.println();
	}
	}
	System.out.println("Descarte: " + (descarte.isEmpty() ? "(vacío)" : descarte.peek()));
	}

	public void jugar() {
```

```
Scanner scanner = new Scanner(System.in);
inicializarJuego();

while (!baraja.estaVacia() || !juegoTerminado()) {
mostrarTablero();

System.out.println("\nOpciones:");
System.out.println("1. Sacar carta del mazo");
System.out.println("2. Mover carta entre pilas");
System.out.println("3. Salir");
System.out.print("Elige una opción: ");
int opcion = scanner.nextInt();

switch (opcion) {
        case 1:
        sacarCartaDelMazo();
        break;
        case 2:
        moverCartaEntrePilas(scanner);
        break;
        case 3:
        System.out.println("¡Gracias por jugar!");
        return;
        default:
        System.out.println("Opción no válida.");
}
}

if (juegoTerminado()) {
System.out.println("¡Felicidades! Has completado el Solitario.");
} else {
System.out.println("Juego terminado.");
}
scanner.close();
}

private void sacarCartaDelMazo() {
if (!baraja.estaVacia()) {
Carta carta = baraja.sacarCarta();
carta.voltear();
descarte.push(carta);
} else {
System.out.println("El mazo está vacío.");
}
}

private void moverCartaEntrePilas(Scanner scanner) {
System.out.print("Ingresa el número de la pila de origen: ");
```

```
        int origen = scanner.nextInt() - 1;
        System.out.print("Ingresa el número de la pila de destino: ");
        int destino = scanner.nextInt() - 1;

        if (origen >= 0 && origen < pilas.size() && destino >= 0 && destino < pilas.size() &&
!pilas.get(origen).isEmpty()) {
        Carta carta = pilas.get(origen).peek();
        if (puedeMoverCarta(carta, pilas.get(destino))) {
                pilas.get(destino).push(pilas.get(origen).pop());
                if (!pilas.get(origen).isEmpty() && !pilas.get(origen).peek().isBocaArriba()) {
                pilas.get(origen).peek().voltear();
                }
        } else {
                System.out.println("Movimiento no válido.");
        }
        } else {
        System.out.println("Pilas inválidas.");
        }
        }

        private boolean puedeMoverCarta(Carta carta, Stack<Carta> destino) {
        if (destino.isEmpty()) {
        return true; // Permitir mover a una pila vacía.
        }
        Carta cartaSuperior = destino.peek();
        return !carta.getPalo().equals(cartaSuperior.getPalo());
        }

        private boolean juegoTerminado() {
        return pilas.stream().allMatch(Stack::isEmpty);
        }

        public static void main(String[] args) {
        Solitario solitario = new Solitario();
        solitario.jugar();
        }
}
```