```
public class Celula {
	private boolean viva;

	// Constructor
	public Celula() {
	this.viva = false;  // Por defecto, la célula está muerta
	}

	// Método para cambiar el estado de la célula
	public void setViva(boolean viva) {
	this.viva = viva;
	}

	// Método para verificar si la célula está viva
	public boolean esViva() {
	return viva;
	}

	// Método para mostrar el estado de la célula (vivida o muerta)
	public String mostrar() {
	return esViva() ? "1" : "0";  // "1" para viva, "0" para muerta
	}
}
```

```
public class Tablero {
	private Celula[][] tablero;
	private int filas;
	private int columnas;

	// Constructor
	public Tablero(int filas, int columnas) {
	this.filas = filas;
	this.columnas = columnas;
	this.tablero = new Celula[filas][columnas];

	// Inicializar todas las células en el tablero como muertas
	for (int i = 0; i < filas; i++) {
	for (int j = 0; j < columnas; j++) {
		tablero[i][j] = new Celula();
	}
	}
	}

	// Método para establecer el estado de una célula en el tablero
	public void setCelula(int fila, int columna, boolean viva) {
	if (fila >= 0 && fila < filas && columna >= 0 && columna < columnas) {
	tablero[fila][columna].setViva(viva);
	}
	}

	// Método para obtener el estado de una célula
	public Celula getCelula(int fila, int columna) {
	if (fila >= 0 && fila < filas && columna >= 0 && columna < columnas) {
	return tablero[fila][columna];
	}
	return null;  // Retorna null si la posición está fuera de los límites
	}

	// Método para mostrar el tablero
	public void mostrar() {
	for (int i = 0; i < filas; i++) {
	for (int j = 0; j < columnas; j++) {
		System.out.print(tablero[i][j].mostrar() + " ");
	}
	System.out.println();
	}
	}

	// Método para obtener el número de vecinos vivos alrededor de una célula
	public int contarVecinosVivos(int fila, int columna) {
	int vecinosVivos = 0;
	// Recorrer las 8 posibles posiciones alrededor de la célula
```

```
        for (int i = -1; i <= 1; i++) {
        for (int j = -1; j <= 1; j++) {
                if (i == 0 && j == 0) continue;  // Ignorar la célula misma
                int nuevaFila = fila + i;
                int nuevaColumna = columna + j;

                if (nuevaFila >= 0 && nuevaFila < filas && nuevaColumna >= 0 &&
nuevaColumna < columnas) {
                if (getCelula(nuevaFila, nuevaColumna).esViva()) {
                vecinosVivos++;
                }
                }
        }
        }
        return vecinosVivos;
        }

        // Método para actualizar el tablero según las reglas del juego
        public void siguienteGeneracion() {
        Celula[][] nuevoTablero = new Celula[filas][columnas];

        // Copiar las referencias del tablero viejo
        for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
                nuevoTablero[i][j] = new Celula();
        }
        }

        // Aplicar las reglas de Conway a cada célula del tablero
        for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
                Celula celula = getCelula(i, j);
                int vecinosVivos = contarVecinosVivos(i, j);

                // Si la célula está viva
                if (celula.esViva()) {
                if (vecinosVivos < 2 || vecinosVivos > 3) {
                nuevoTablero[i][j].setViva(false);  // La célula muere
                } else {
                nuevoTablero[i][j].setViva(true);  // La célula sigue viva
                }
                }
                // Si la célula está muerta
                else {
                if (vecinosVivos == 3) {
                nuevoTablero[i][j].s
```

```
import java.util.Scanner;

public class JuegoDeLaVida {
	private Tablero tablero;
	private int generaciones;

	// Constructor
	public JuegoDeLaVida(int filas, int columnas, int generaciones) {
	this.tablero = new Tablero(filas, columnas);
	this.generaciones = generaciones;
	}

	// Método para configurar algunas células vivas al principio
	public void configurarInicial() {
	// Configuramos algunas células vivas al inicio
	tablero.setCelula(1, 2, true);
	tablero.setCelula(2, 2, true);
	tablero.setCelula(3, 2, true);
	tablero.setCelula(2, 1, true);
	}

	// Método para ejecutar el juego
	public void ejecutar() {
	Scanner scanner = new Scanner(System.in);

	// Configurar el tablero inicial
	configurarInicial();

	// Mostrar las generaciones del juego
	for (int i = 0; i < generaciones; i++) {
	System.out.println("Generación " + (i + 1));
	tablero.mostrar();
	System.out.println();

	// Avanzar a la siguiente generación
	tablero.siguienteGeneracion();

	// Pausar entre generaciones
	if (i < generaciones - 1) {
		System.out.println("Presione Enter para continuar...");
		scanner.nextLine();
	}
	}

	scanner.close();
	}

	public static void main(String[] args) {
```

```
        // Crear el juego con 5 filas, 5 columnas y 10 generaciones
        JuegoDeLaVida juego = new JuegoDeLaVida(5, 5, 10);
        juego.ejecutar();
        }
}
```