



David Alejandro Galicia Cárdenas

Programación de Dispositivos Móviles

2025 – 1

Unidad 6 – Actividad 1

6 de Noviembre 2024

Código fuente

Clase Aeropuerto

```
package com.example.u6;

import android.util.Log;

import java.util.HashSet;
import java.util.Set;
import java.util.Stack;

public class Aeropuerto {
    private int filas;
    private int columnas;
    private Avion[][] grid;
    private int colisiones;
    private Stack<Avion[][]> historialMovimientos;
    private int pasos;

    public Aeropuerto(int filas, int columnas) {
        this.filas = filas;
        this.columnas = columnas;
        grid = new Avion[filas][columnas];
        colisiones = 0;
        pasos = 0;
        historialMovimientos = new Stack<>();
    }

    public boolean colocarAvion(Avion avion, int fila, int columna) {
        if (fila >= 0 && fila < filas && columna >= 0 && columna < columnas &&
            grid[fila][columna] == null) {
            grid[fila][columna] = avion;
            return true;
        } else {
            System.out.println("no se puede poner en la posicion (" + fila + ", " + columna
                + ")");
            return false;
        }
    }
}
```

```

public void moverAviones() {
    guardarEstadoActual();
    pasos++;

    Avion[][] nuevoGrid = new Avion[filas][columnas];
    Set<String> posicionesOcupadas = new HashSet<>();

    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
            if (grid[i][j] != null) {
                Avion avion = grid[i][j];
                int[] nuevaPos = calcularNuevaPosicion(avion, i, j);

                String posClave = nuevaPos[0] + "," + nuevaPos[1];

                Log.d("Movimiento", "avion en (" + i + ", " + j + ") que se mueve a (" +
nuevaPos[0] + ", " + nuevaPos[1] + ")");
                Log.d("Posiciones", "revisando posicion clave: " + posClave);

                if (posicionesOcupadas.contains(posClave)) {
                    colisiones++;
                    Log.d("Colision", "colision detectada en (" + nuevaPos[0] + ", " +
nuevaPos[1] + ")");
                    Log.d("Colision", "colision detectada en: " + posClave);
                } else {
                    posicionesOcupadas.add(posClave);
                    nuevoGrid[nuevaPos[0]][nuevaPos[1]] = avion;
                    Log.d("Posiciones", "posicion agregada: " + posClave);
                }
            }
        }
    }

    grid = nuevoGrid;
    mostrarGrid();
}

```

```

private void guardarEstadoActual() {
    Avion[][] copiaGrid = new Avion[filas][columnas];
    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
            copiaGrid[i][j] = grid[i][j];
        }
    }
    historialMovimientos.push(copiaGrid);
}

```

```

public void retrocederPaso() {
    if (!historialMovimientos.isEmpty()) {
        grid = historialMovimientos.pop();
        pasos = Math.max(0, pasos - 1);
        mostrarGrid();
    } else {
        System.out.println("no se puede retroceder :(");
    }
}

```

```

private int[] calcularNuevaPosicion(Avion avion, int x, int y) {
    int nuevoX = x;
    int nuevoY = y;

    switch (avion.getDireccion()) {
        case "Norte":
            if (x > 0) nuevoX = x - 1;
            break;
        case "Sur":
            if (x < filas - 1) nuevoX = x + 1;
            break;
        case "Este":
            if (y < columnas - 1) nuevoY = y + 1;
            break;
        case "Oeste":
            if (y > 0) nuevoY = y - 1;
            break;
    }
}

```

```

    }
    return new int[]{nuevoX, nuevoY};
}

```

```

public void mostrarGrid() {
    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
            if (grid[i][j] == null) {
                System.out.print(". ");
            } else {
                System.out.print("A ");
            }
        }
        System.out.println();
    }
}

```

```

public Avion[][] getGrid() {
    return grid;
}

```

```

public int getColisiones() {
    return colisiones;
}

```

```

public int getPasos() {
    return pasos;
}
}

```

Clase Avion

```

package com.example.u6;

```

```

public class Avion {
    private int id;
    private String direccion;

    public Avion(int id, String direccion) {

```

```

        this.id = id;
        this.direccion = direccion;
    }

    public int getId() {
        return id;
    }

    public String getDireccion() {
        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    public int getImagenSegunDireccion() {
        switch (direccion) {
            case "Norte":
                return R.drawable.arriba;
            case "Sur":
                return R.drawable.abajo;
            case "Este":
                return R.drawable.derecha;
            case "Oeste":
                return R.drawable.izquierda;
            default:
                return R.drawable.arriba;
        }
    }
}

```

Clase Main

```

package com.example.u6;

import android.os.Bundle;
import android.util.Log;
import android.view.animation.Animation;
import android.view.animation.ScaleAnimation;

```



```

import android.widget.Button;
import android.widget.GridLayout;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {
    private Aeropuerto aeropuerto;
    private GridLayout gridLayout;
    private TextView textViewInfo;
    private List<ImageView> celdasGrid;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        gridLayout = findViewById(R.id.gridLayout);
        textViewInfo = findViewById(R.id.textViewInfo);

        aeropuerto = new Aeropuerto(5, 5);
        textViewInfo = findViewById(R.id.textViewInfo);

        celdasGrid = new ArrayList<>();

        int filas = 5;
        int columnas = 5;
        aeropuerto = new Aeropuerto(filas, columnas);

        for (int i = 0; i < filas * columnas; i++) {
            ImageView imageView = new ImageView(this);
            GridLayout.LayoutParams params = new GridLayout.LayoutParams();
            params.width = 0;
            params.height = 0;
            params.columnSpec = GridLayout.spec(GridLayout.UNDEFINED, 1f);
            params.rowSpec = GridLayout.spec(GridLayout.UNDEFINED, 1f);
            params.setMargins(2, 2, 2, 2);

```

```

        imageView.setLayoutParams(params);
        imageView.setBackgroundResource(R.drawable.celda_vacia);
        gridLayout.addView(imageView);
        celdasGrid.add(imageView);
    }

    Avion avion1 = new Avion(1, "Este");
    Avion avion2 = new Avion(2, "Oeste");

    aeropuerto.colocarAvion(avion1, 2, 2); // hay q colocarlo en (2, 2)
    aeropuerto.colocarAvion(avion2, 2, 3); // hay q colocarlo en (3, 3)

    actualizarGridUI();
    actualizarInfo();

    Button buttonNext = findViewById(R.id.buttonSiguiente);
    buttonNext.setOnClickListener(v -> {
        aeropuerto.moverAviones();
        actualizarGridUI();
        actualizarInfo();
    });

    Button buttonPrevious = findViewById(R.id.buttonAnterior);
    buttonPrevious.setOnClickListener(v -> {
        aeropuerto.retrocederPaso();
        actualizarGridUI();
        actualizarInfo();
    });
}

private void actualizarGridUI() {
    for (ImageView celda : celdasGrid) {
        celda.setImageResource(R.drawable.celda_vacia);
    }

    Avion[][] grid = aeropuerto.getGrid();
    for (int i = 0; i < grid.length; i++) {
        for (int j = 0; j < grid[i].length; j++) {
            Avion avion = grid[i][j];
            if (avion != null) {

```

```

        int index = i * grid[i].length + j;
        ImageView celda = celdasGrid.get(index);

        celda.setImageResource(getDrawableForDireccion(avion.getDireccion()));
        animarCelda(celda);
    }
}

}

}

private int getDrawableForDireccion(String direccion) {
    switch (direccion) {
        case "Norte":
            return R.drawable.arriba;
        case "Sur":
            return R.drawable.abajo;
        case "Este":
            return R.drawable.derecha;
        case "Oeste":
            return R.drawable.izquierda;
        default:
            return R.drawable.arriba;
    }
}

private void animarCelda(ImageView celda) {
    Animation scaleAnimation = new ScaleAnimation(
        1.0f, 1.5f,
        1.0f, 1.5f,
        Animation.RELATIVE_TO_SELF, 0.5f,
        Animation.RELATIVE_TO_SELF, 0.5f
    );
    scaleAnimation.setDuration(300);
    scaleAnimation.setRepeatCount(1);
    scaleAnimation.setRepeatMode(Animation.REVERSE);
    celda.startAnimation(scaleAnimation);
}

private void actualizarInfo() {

```

```

int pasos = aeropuerto.getPasos();
int colisiones = aeropuerto.getColisiones();
textViewInfo.setText("pasos: " + pasos + " | colisiones: " + colisiones);
}

```

```

public void mostrarGrid() {
    Log.d("mostrarGrid", "cambio del grid en la interfaz");
    GridLayout gridLayout = findViewById(R.id.gridLayout);
    gridLayout.removeAllViews();

    Avion[][] grid = aeropuerto.getGrid();

    int filas = grid.length;
    int columnas = grid[0].length;

    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
            ImageView imageView = new ImageView(this);

            Avion avion = null;
            if (grid[i][j] != null) {
                avion = grid[i][j];
                imageView.setImageResource(avion.getImagenSegunDireccion());
            } else {
                imageView.setImageResource(R.drawable.celda_vacia);
            }

            GridLayout.LayoutParams params = new GridLayout.LayoutParams();
            params.width = 0;
            params.height = 0;
            params.rowSpec = GridLayout.spec(i, 1f);
            params.columnSpec = GridLayout.spec(j, 1f);

            gridLayout.addView(imageView);

            if (avion != null) {
                Log.d("Avion", "posicion: (" + i + ", " + j + ") direccion: " +
                    avion.getDireccion());
            }
        }
    }
}

```

}

Conclusión

El archivo de GitHub no me permitió descargarlo, quise abrir el ZIP o RAR creo era y no me dejaba, así que en base a lo que se me pidió fue lo que hice. El programa desde sus inicios me dio muchos dolores de cabeza ya que es la primera vez que hago algo así y no sabía por donde empezar. Aunque toma en cuenta los pasos, el programa tiene errores y el único que yo he notado hasta ahora es que no detecta las colisiones y traté de arreglarlo varias veces con mensajes de depuración y demás, pero no supe dar con el error e incluso un cambio mínimo hacia que el programa actuase de manera caótica.