UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

ASIGNATURA: Fundamentos de Programación II

Practica 03 - teoría

DOCENTE:

Richart Smith Escobedo Quispe

INTEGRANTES:

- **Companocca Checco Roni**

Arequipa – Perú

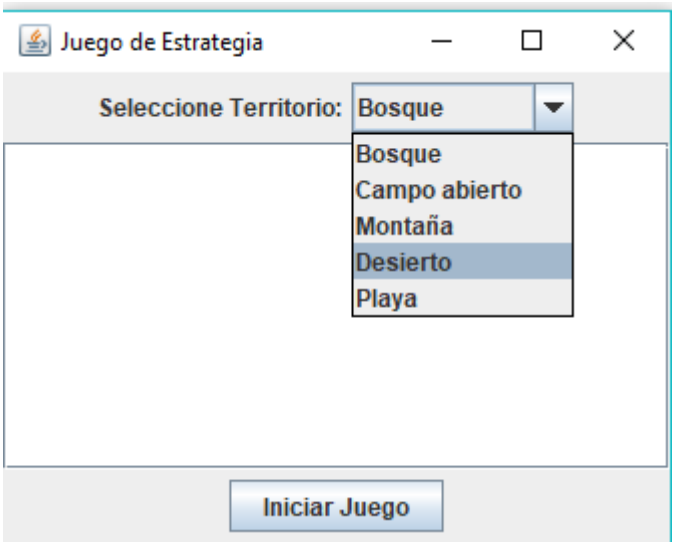
2023

Link REPOSITORIO:

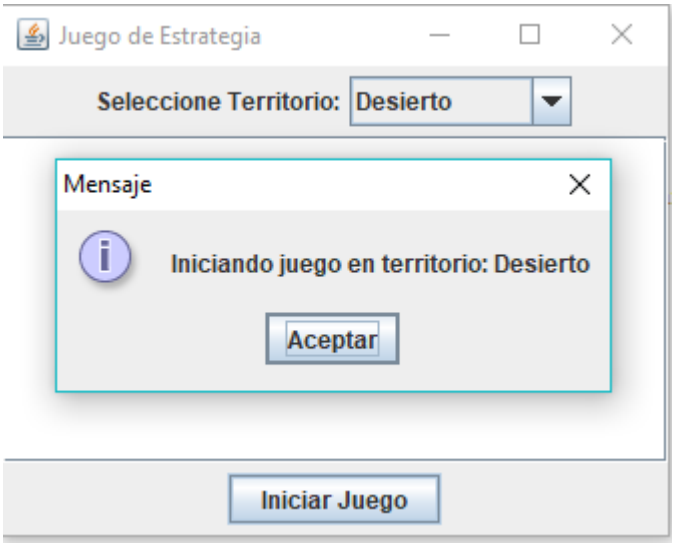
https://github.com/RONI-COMPANOCCA-CHECCO/PRACTICA03-FASE03

Genere un tablero(10x10) para el juego que está desarrollando en laboratorio usando Swing/JavaFX.
- La interfaz generada debe tener las recomendaciones del libro de Jhonson.

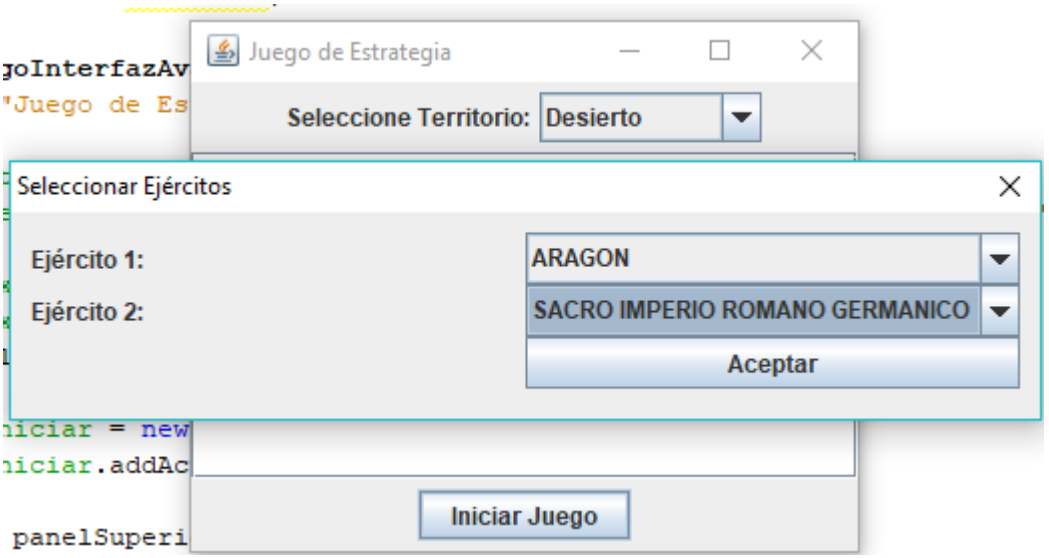
1.



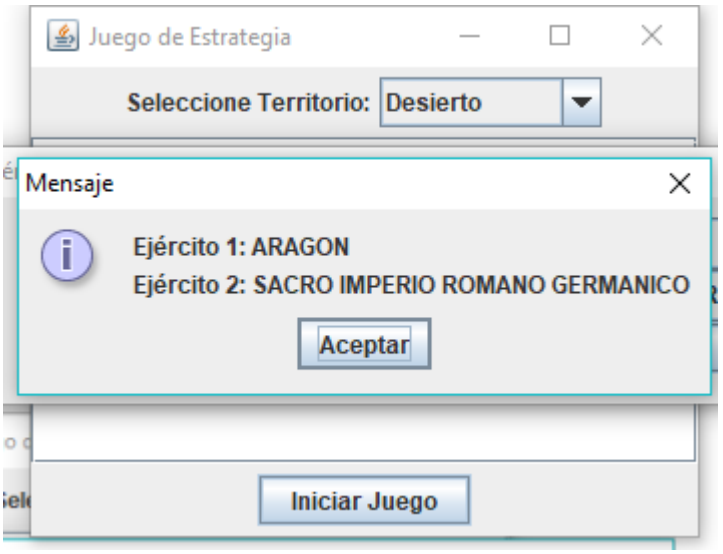
2.



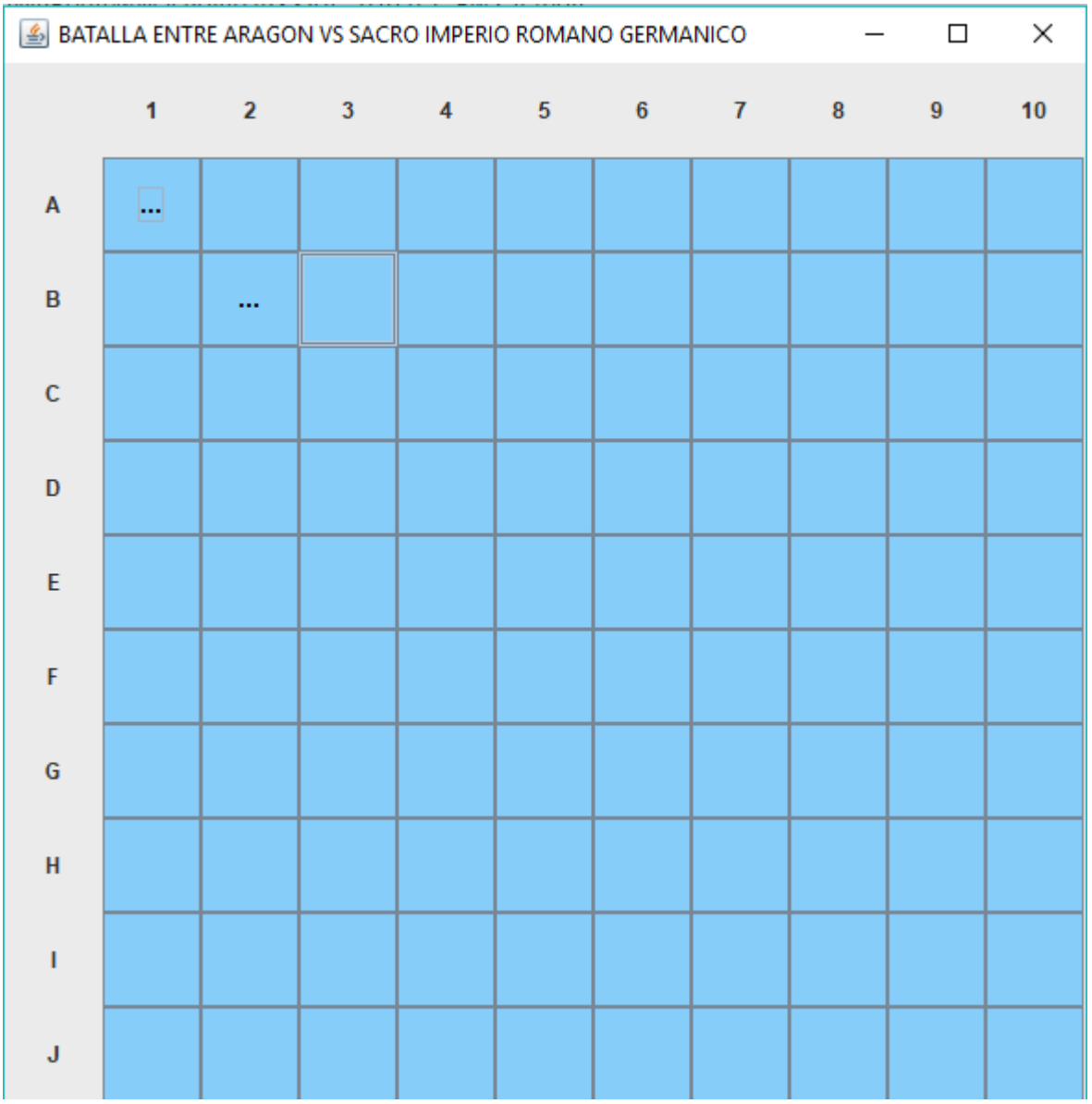
3.


goInterfazAv
"Juego de Es
iciar = new
iciar.addAc
panelSuperi
Juego de Estrategia
Seleccione Territorio:
Desierto
Seleccionar Ejércitos
Ejército 1:
ARAGON
Ejército 2:
SACRO IMPERIO ROMANO GERMANICO
Aceptar
Iniciar Juego

4.


Juego de Estrategia
Seleccione Territorio:
Desierto
Mensaje
Ejército 1: ARAGON
Ejército 2: SACRO IMPERIO ROMANO GERMANICO
Aceptar
Iniciar Juego

5.



- Averiguar en Java cuanto de memoria RAM utilizó para generar su tablero. (Print en consola)

```
// Obtener información sobre la memoria
Runtime runtime = Runtime.getRuntime();
long totalMemory = runtime.totalMemory();
long freeMemory = runtime.freeMemory();
long usedMemory = totalMemory - freeMemory;

System.out.println("Total de memoria: " + totalMemory + " bytes");
System.out.println("Memoria libre: " + freeMemory + " bytes");
System.out.println("Memoria utilizada: " + usedMemory + " bytes");
```

Ejecución:

```
Total de memoria: 128974848 bytes
Memoria libre: 122158344 bytes
Memoria utilizada: 6816504 bytes
```

- Qué patrón de diseño de software utilizará para reducir la cantidad de memoria utilizada.

1. Implementaremos el patrón de diseño uso de recursos gráficos: En la clase JuegoInterfazAvanzada, para intentar liberar recursos gráficos cuando ya no los necesitemos.

```
// Cerrar la ventana de selección de ejércitos
((Window) SwingUtilities.getRoot(botonAceptar)).dispose();
```

```
// Liberar recursos gráficos al cerrar la ventana
@Override
public void dispose() {
    // Realizar cualquier limpieza necesaria antes de cerrar la ventana
    super.dispose();
}
```

2. Para reducir la cantidad de memoria utilizada en nuestra aplicación, podemos considerar utilizar el patrón de diseño Flyweight. Ya que nos permite minimizar la cantidad de objetos o estructuras de datos utilizados en la aplicación, compartiendo eficientemente recursos similares entre varios objetos. En el contexto de nuestra aplicación, podemos aplicar el patrón Flyweight para optimizar el uso de recursos gráficos, especialmente cuando tenemos una gran cantidad de elementos similares, como los botones y etiquetas en tu interfaz gráfica.

Creamos una interfaz Flyweight

```
package paquete;
// ElementoFlyweight.java
public interface ElementoFlyweight {
    void mostrar();
}
```

Seguidamente implementamos una clase concreta Flyweight

```java
package paquete;

// Clase concreta que implementa la interfaz Flyweight
public class EtiquetaTerritorio implements ElementoFlyweight {
    private String contenido;

    public EtiquetaTerritorio(String contenido) {
        this.contenido = contenido;
    }

    @Override
    public void mostrar() {
        // mostramos el elemento
        System.out.println(contenido);
    }
}
```

Creamos una fábrica Flyweight

```java
package paquete;

/**
 *
 * @author Roni
 */
import java.util.HashMap;
import java.util.Map;

public class FabricaElementoFlyweight {
    private Map<String, ElementoFlyweight> elementos = new HashMap<>();

    public ElementoFlyweight getElemento(String contenido) {
        if (!elementos.containsKey(contenido)) {
            elementos.put(contenido, new EtiquetaTerritorio(contenido));
        }
        return elementos.get(contenido);
    }
}
```

utiliza la fábrica y los objetos Flyweight para crear instancias compartidas cuando sea necesario.

```java
// Utilizar Flyweight para crear instancias de etiquetas compartidas
ElementoFlyweight etiquetaTerritorio = fabricaElementoFlyweight.getElemento("Territorio: " + selectedCultura);
etiquetaTerritorio.mostrar();
```

- Adjunte informe de investigación y código fuente.

CLASE INTERFAZ

```java
package paquete;
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;

public class JuegoInterfazAvanzada extends JFrame {

            // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
        private void initComponents() {

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

                                javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
            getContentPane().setLayout(layout);
            layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGap(0, 400, Short.MAX_VALUE)
            );
            layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGap(0, 300, Short.MAX_VALUE)
            );
```

```
            pack();
        }// </editor-fold>//GEN-END:initComponents

        // Variables declaration - do not modify//GEN-BEGIN:variables
        // End of variables declaration//GEN-END:variables

    private JLabel etiquetaTerritorio;
    private JComboBox<String> listaTerritorio;
    private JButton botonIniciar;
    private JTextArea areaTexto;
      private FabricaElementoFlyweight fabricaElementoFlyweight = new FabricaElementoFlyweight();

    public JuegoInterfazAvanzada() {
        super("Juego de Estrategia");

        etiquetaTerritorio = new JLabel("Seleccione Territorio:");
         listaTerritorio = new JComboBox<>(new String[]{"Bosque", "Campo abierto", "Montaña", "Desierto", "Playa"});

        areaTexto = new JTextArea(10, 30);
        areaTexto.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(areaTexto);

        botonIniciar = new JButton("Iniciar Juego");
```

```
        botonIniciar.addActionListener(e -> iniciarJuego());

        JPanel panelSuperior = new JPanel(new FlowLayout());
        panelSuperior.add(etiquetaTerritorio);
        panelSuperior.add(listaTerritorio);

        JPanel panelCentral = new JPanel(new BorderLayout());
        panelCentral.add(panelSuperior, BorderLayout.NORTH);
        panelCentral.add(scrollPane, BorderLayout.CENTER);

        JPanel panelBoton = new JPanel();
        panelBoton.add(botonIniciar);

        setLayout(new BorderLayout());
        add(panelCentral, BorderLayout.CENTER);
        add(panelBoton, BorderLayout.SOUTH);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pack();
        setLocationRelativeTo(null);
        setVisible(true);
    }

    private void iniciarJuego() {
```

```
                                String selectedCultura = (String)
listaTerritorio.getSelectedItem();

                mostrarMensaje("Iniciando juego en territorio: " +
selectedCultura);



            // Utilizar Flyweight para crear instancias de etiquetas
compartidas

                               ElementoFlyweight etiquetaTerritorio =
fabricaElementoFlyweight.getElemento("Territorio: " + selectedCultura);

        etiquetaTerritorio.mostrar();



          // Crear una ventana para seleccionar ejércitos en la misma
ventana

          String[] ejercitos = {"INGLATERRA", "FRANCIA", "SACRO IMPERIO
ROMANO GERMANICO", "ARAGON", "MOROS"};

        JComboBox<String> comboEjercito1 = new JComboBox<>(ejercitos);

        JComboBox<String> comboEjercito2 = new JComboBox<>(ejercitos);



        JButton botonAceptar = new JButton("Aceptar");

        botonAceptar.addActionListener(e -> {

                            String ejercitoSeleccionado1 = (String)
comboEjercito1.getSelectedItem();

                            String ejercitoSeleccionado2 = (String)
comboEjercito2.getSelectedItem();



        // Lógica para iniciar el juego con los ejércitos seleccionados

                mostrarMensaje("Ejército 1: " + ejercitoSeleccionado1 +
"\nEjército 2: " + ejercitoSeleccionado2);
```

```
        // Puedes agregar más lógica según tus necesidades


        // Crear el Tablero con los ejércitos seleccionados
        SwingUtilities.invokeLater(() -> {
                            Tablero tablero = new Tablero(10, 10,
ejercitoSeleccionado1, ejercitoSeleccionado2);
            tablero.setLocationRelativeTo(null); // Centra la ventana
            tablero.setVisible(true);
        });


        // Cerrar la ventana de selección de ejércitos
        ((Window) SwingUtilities.getRoot(botonAceptar)).dispose();
        });


          JPanel panelSeleccionEjercitos = new JPanel(new GridLayout(3,
2));
        panelSeleccionEjercitos.add(new JLabel("Ejército 1:"));
        panelSeleccionEjercitos.add(comboEjercito1);
        panelSeleccionEjercitos.add(new JLabel("Ejército 2:"));
        panelSeleccionEjercitos.add(comboEjercito2);
        panelSeleccionEjercitos.add(new JLabel());
        panelSeleccionEjercitos.add(botonAceptar);


        JOptionPane.showOptionDialog(
            this,
            panelSeleccionEjercitos,
```

```
            "Seleccionar Ejércitos",
            JOptionPane.DEFAULT_OPTION,
            JOptionPane.PLAIN_MESSAGE,
            null,
            new Object[]{},
            null
        );
    }

    private void mostrarMensaje(String mensaje) {
              JOptionPane.showMessageDialog(this, mensaje, "Mensaje",
JOptionPane.INFORMATION_MESSAGE);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new JuegoInterfazAvanzada());
    }

}
```

CLASE TABLERO

```
package paquete;

import javax.swing.*;
```

```
import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class Tablero extends javax.swing.JFrame {

    @SuppressWarnings("unchecked")

          //   <editor-fold   defaultstate="collapsed"   desc="Generated
Code">//GEN-BEGIN:initComponents

    private void initComponents() {



setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);



                            javax.swing.GroupLayout    layout    =    new
javax.swing.GroupLayout(getContentPane());

        getContentPane().setLayout(layout);

        layout.setHorizontalGroup(


layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGap(0, 400, Short.MAX_VALUE)

        );

        layout.setVerticalGroup(


layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGap(0, 300, Short.MAX_VALUE)

        );



        pack();
```

```
    }// </editor-fold>//GEN-END:initComponents


    // Variables declaration - do not modify//GEN-BEGIN:variables
    // End of variables declaration//GEN-END:variables
    private JButton[][] botones;
    private Ejercito e1;
    private Ejercito e2;


      public Tablero(int filas, int columnas, String nombreEjercito1,
String nombreEjercito2) {
              super("BATALLA ENTRE " + nombreEjercito1 + " VS " +
nombreEjercito2);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(600, 600);


        botones = new JButton[filas][columnas];


        // Configura el diseño del contenedor
        setLayout(new GridLayout(filas + 1, columnas + 1));


        // Agrega una etiqueta vacía en la esquina superior izquierda
        add(new JLabel());


        // Agrega etiquetas para las columnas (números)
        for (int i = 1; i <= columnas; i++) {
```

```
                    JLabel label = new JLabel(Integer.toString(i),
JLabel.CENTER);

            add(label);

        }



        // Crear y agregar botones al Tablero con letras para las filas

        for (int i = 1; i <= filas; i++) {

             JLabel label = new JLabel(Character.toString((char) ('A' +
i - 1)), JLabel.CENTER);

            add(label);



            for (int j = 1; j <= columnas; j++) {

                botones[i - 1][j - 1] = new JButton();

                 botones[i - 1][j - 1].setBackground(new Color(135, 206,
250));

                botones[i - 1][j - 1].setForeground(Color.BLACK);

                       botones[i - 1][j - 1].setFont(new Font("Arial",
Font.BOLD, 14));

                           botones[i - 1][j - 1].addActionListener(new
BotonListener());



                add(botones[i - 1][j - 1]);

            }

        }



            // Lógica para la creación de Ejércitos y colocación de
soldados
```

```
        int cant;

            String cultura[] = {"Inglaterra","Francia","Sacro Imperio
Romano Germanico","Aragon","Moros"};



        cant = aleatorio(1, 10);

        e1 = new Ejercito(nombreEjercito1, cant);

        colocarSoldado(0, 0, e1.misSoldados.get(0));



        cant = aleatorio(1, 10);

        e2 = new Ejercito(nombreEjercito2, cant);

        colocarSoldado(1, 1, e2.misSoldados.get(0));



        // Atacar entre soldados

        e1.misSoldados.get(0).atacar(e2.misSoldados.get(0));



          // Mostrar el Tablero (puedes implementar un método mostrar()
en la clase Tablero)

        mostrar();



        setVisible(true);



        // Obtener información sobre la memoria

        Runtime runtime = Runtime.getRuntime();

        long totalMemory = runtime.totalMemory();

        long freeMemory = runtime.freeMemory();

        long usedMemory = totalMemory - freeMemory;
```

```
            System.out.println("Total de memoria: " + totalMemory + "
bytes");
        System.out.println("Memoria libre: " + freeMemory + " bytes");
            System.out.println("Memoria utilizada: " + usedMemory + "
bytes");
    }

    private class BotonListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            JButton boton = (JButton) e.getSource();
            System.out.println("Clic en el botón: " + boton.getText());
        }
    }

     public void colocarSoldado(int fila, int columna, Soldado soldado)
{
        botones[fila][columna].setText(soldado.toString());
        soldado.setPosicion(fila, columna);
    }

    public static int aleatorio(int a, int b) {
        return (int) (Math.random() * (b - a + 1)) + a;
    }
```

```
    public void mostrar() {

        // Lógica para mostrar información de los Ejércitos

        System.out.print("Ejercito 1 " + e1.getCultura());

                System.out.println("\nCantidad total de Soldados: " +
Soldado.cuantos() + "\n"

                 + "Espadachines: " + Espadachin.cuantos() + "\n"

                 + "Arqueros: " + Arquero.cuantos() + "\n"

                 + "Lanceros: " + Lancero.cuantos() + "\n"

                 + "Caballeros: " + Caballero.cuantos() + "\n");

        Ejercito.resetearCantidad();



        System.out.print("Ejercito 2 " + e2.getCultura());

                System.out.println("\nCantidad total de Soldados: " +
Soldado.cuantos() + "\n"

                 + "Espadachines: " + Espadachin.cuantos() + "\n"

                 + "Arqueros: " + Arquero.cuantos() + "\n"

                 + "Lanceros: " + Lancero.cuantos() + "\n"

                 + "Caballeros: " + Caballero.cuantos() + "\n");

        Ejercito.resetearCantidad();

    }



    public static void main(String[] args) {

    SwingUtilities.invokeLater(() -> {

                Tablero tablero = new Tablero(10, 10, "Ejercito1",
"Ejercito2");

        tablero.setLocationRelativeTo(null); // Centra la ventana
```

```
        tablero.setVisible(true);

    });

}

}
```