Pauta Certamen 3, Programación II

1. *30pts.* De las siguentes afirmaciones, encierre en un círculo la o las alternativas correctas. Pueden ser todas, algunas, una o ninguna [60 minutos].

   i. *3pts.* Respecto a las interfaces gráficas en Java:
   (a) Swing sustituye a AWT.
   (b) AWT sustituye a Swing.
   (c) AWT se apoya en Swing.
   (d) AWT incopora los JComponents.
   (e) Swing proporciona los ActionEvent.

   ii. *4pts.* Referente a JFrame:
   (a) Habitualmente se usa para crear la ventana principal.
   (b) Su método getPaneContent() obtiene el panel principal.
   (c) Su método add() permite agregar componentes al panel.
   (d) Su método size() permite dimensionar la ventana.
   (e) Su método pack() empaqueta las clases.

   iii. *4pts.* Continuando con JFrame:
   (a) Es posible implementarlo.
   (b) Es posible extenderlo.
   (c) Es posible instanciarlo.
   (d) Incluye el atributo estático CLOSE_AND_EXIT.
   (e) Incluye el método setVisible(boolean).

   iv. *4pts.* Para realizar acciones desde un botón, se requiere:
   (a) Crear una clase que implemente un ActionEvent.
   (b) Crear una clase que implemente un ActionList.
   (c) Sobreescribir el método actionList(ActionPerformance)
   (d) Sobreescribir el método actionArrayList(ActionEvent)
   (e) Agregar la instancia que implementa un ActionListener.

   v. *3pts.* Algunos JComponents :
   (a) JPane, JScrollPane, JDialog.
   (b) JPanel, JScrollPanel, JDialogPane.
   (c) JFileChooser, JScrollPane, JLabel.
   (d) JCheckRadio, JTextArea, JText.
   (e) JDialogPane, JTextField, JTextArea.

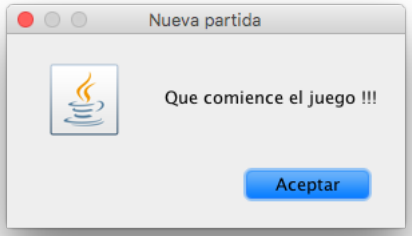
   vi. *4pts.* Sobre los BorderLayout:
   (a) Cada región se ordena GridLayout.
   (b) Cada región es obligatoria.
   (c) Su constructor recibe argumentos de margen.
   (d) Agrega un panel en cada región.
   (e) setFlowLayout(boolean) fija una nueva dispoción.

   vii. *4pts.* Sobre los JCheckbox:
   (a) Al pulsarse, cambia de estado.
   (b) Es una especialización del JButton.
   (c) Al ser un botón, tambien puede escuchar acciones.
   (d) El método isSelected() entrega su estado.
   (e) Sólo se puede seleccionar uno de un grupo.

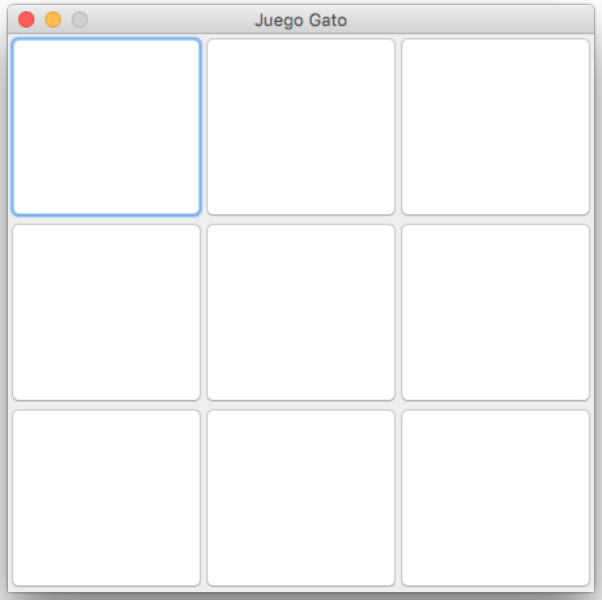
   viii. *4pts.* Sobre los JLabel:
   (a) Su constructor recibe texto.
   (b) Su constructor recibe imágenes.
   (c) Su constructor recibe texto e imágenes, al mismo tiempo.
   (d) Su constructor recibe el tamaño del texto a mostrar.
   (e) No pueden ser modificados en ejecución.

2. *70pts.* Desarrolle el juego "El Gato". Se pide lo siguiente:

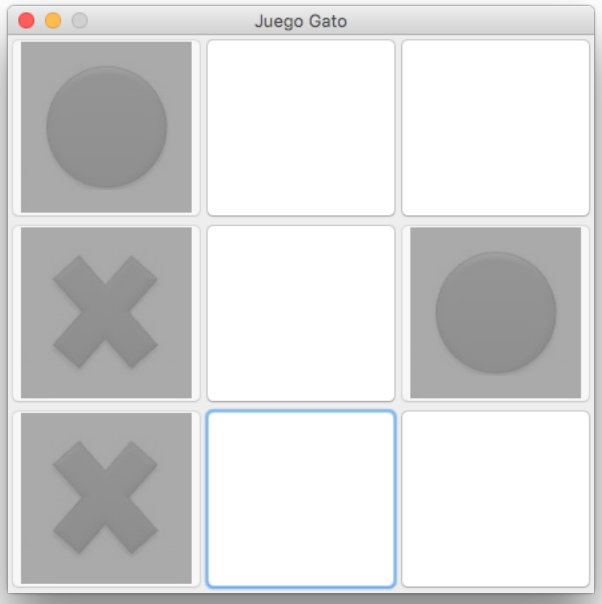
   - Al iniciar el sistema, se debe deplegar el siguiente mensaje:



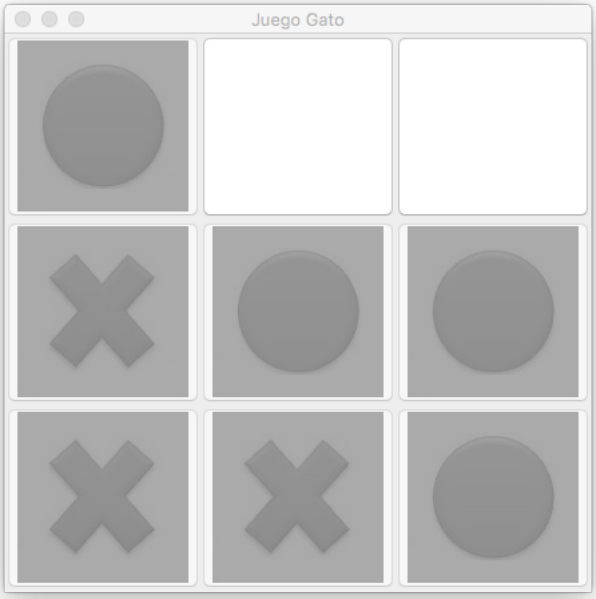
   - Al presionar aceptar, se despliega el tablero vacío:

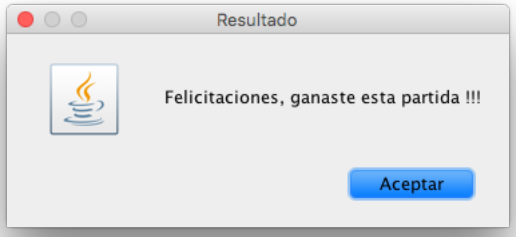


   - El jugardor siempre comienza jugando.
   - La selección de **cruz** y **círculo** debe ser al azar, para el jugador y la máquina.
   - Al presionar en una casilla, ésta se selecciona y se bloquea. En ese instante, la máquina selecciona al **azar** una casilla libre (no bloqueada):

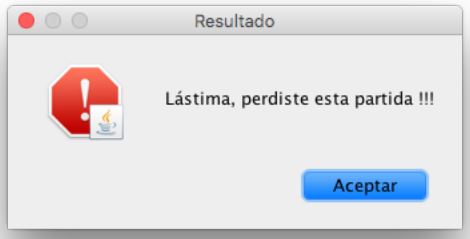


- Si el jugador obtiene 3 selecciones en línea (horizontal, recta, diagonal derecha o diagonal izquierda), gana. Se debe informar que el jugador ha ganado la partida.

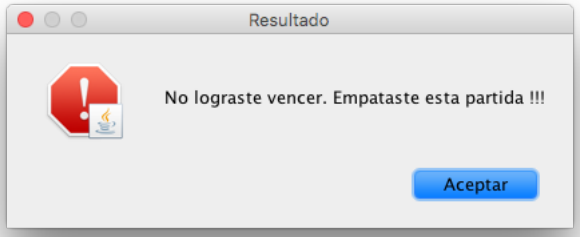




- Si la máquina obtiene 3 selecciones en línea (horizontal, recta, diagonal derecha o diagonal izquierda), ud pierde. Se debe informar que la máquina ha ganado la partida.



- Si hay empate, el sistema debe desplegar el siguiente mensaje:



```
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.Serializable;
import java.util.Random;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class JuegoGato extends JFrame {

    private final JButton[][] casillas = new JButton[3][3];
    private Juego juego;

    public JuegoGato() {
        super("Juego Gato");
        iniciarPartida();
        setLayout(new GridLayout(3, 3));
        setSize(440, 440);
        setResizable(false);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    private void iniciarPartida() {
        juego = new Juego();
        for (int i = 0; i < casillas.length; i++) {
            for (int j = 0; j < casillas[i].length; j++) {
                casillas[i][j] = new JButton();
                casillas[i][j].setBackground(Color.BLACK);
                casillas[i][j].setIcon(new ImageIcon("images/blanco.png"));
                casillas[i][j].setName("[" + i + "," + j + "]");
                casillas[i][j].addActionListener(new OyenteBotonJuego());
                add(casillas[i][j]);
            }
        }
        JOptionPane.showMessageDialog(this, "Que comience el juego !!!", "Nueva partida
    ",
                JOptionPane.INFORMATION_MESSAGE);
    }

    public static void main(String[] args) {
        new JuegoGato();
    }

    class OyenteBotonJuego implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            switch (juego.jugar(((JButton) e.getSource()))) {
                case Juego.GANA:
                    JOptionPane.showMessageDialog(null, "Felicitaciones, ganaste esta
    partida !!!", "Resultado",
                            JOptionPane.INFORMATION_MESSAGE);
                    System.exit(JFrame.EXIT_ON_CLOSE);
                case Juego.PIERDE:
                    JOptionPane.showMessageDialog(null, "L stima, perdiste esta
    partida !!!", "Resultado",
                            JOptionPane.ERROR_MESSAGE);

                    System.exit(JFrame.EXIT_ON_CLOSE);
                case Juego.EMPATE:
                    JOptionPane.showMessageDialog(null, "No lograste vencer. Empataste
    esta partida !!!", "Resultado",
                            JOptionPane.ERROR_MESSAGE);
                    System.exit(JFrame.EXIT_ON_CLOSE);
```

```
            case Juego.SIGUE:
                break;
        }
    }
}

class Juego {

    private String jugador, maquina;
    private final String path = "imagenes/", extensionImagen = ".png";
    protected static final int GANA = 1, PIERDE = -1, EMPATE = 0, SIGUE = 2;
    private int jugada;
    private final Random r;

    public Juego() {
        jugada = 0;
        r = new Random();
    }

    private void validarPirmeraJugada(String botonPresionado) {
        if (jugada == 0) {
            if (r.nextBoolean()) {
                jugador = "cruz";
                maquina = "circulo";
            } else {
                jugador = "circulo";
                maquina = "cruz";
            }
        }
    }

    private void juegaJugador(JButton botonPresionado) {
        validarPirmeraJugada(botonPresionado.getName());
        if (botonPresionado.isEnabled()) {
            botonPresionado.setIcon(new ImageIcon(path + jugador + extensionImagen)
);
            botonPresionado.setName(jugador);
            botonPresionado.setEnabled(false);
            jugada++;
        }
    }

    private void juegaMaquina() {
        int i, j;
        do {
            i = r.nextInt(3);
            j = r.nextInt(3);
        } while (!casillas[i][j].isEnabled());
        casillas[i][j].setIcon(new ImageIcon(path + maquina + extensionImagen));
        casillas[i][j].setName(maquina);
        casillas[i][j].setEnabled(false);
        jugada++;
    }

    private boolean validarJuego(String player) {
        if ((casillas[0][0].getName().equals(player)) && (casillas[0][1].getName().
equals(player))
                && (casillas[0][2].getName().equals(player))) {
            return true;
        }
        if ((casillas[1][0].getName().equals(player)) && (casillas[1][1].getName().
equals(player))
                && (casillas[1][2].getName().equals(player))) {
            return true;
        }
        if ((casillas[2][0].getName().equals(player)) && (casillas[2][1].getName().
equals(player))
                && (casillas[2][2].getName().equals(player))) {
```

```
                return true;
            }
            if ((casillas[0][0].getName().equals(player)) && (casillas[1][0].getName().
    equals(player))
                    && (casillas[2][0].getName().equals(player))) {
                return true;
            }
            if ((casillas[0][1].getName().equals(player)) && (casillas[1][1].getName().
    equals(player))
                    && (casillas[2][1].getName().equals(player))) {
                return true;
            }
            if ((casillas[0][2].getName().equals(player)) && (casillas[1][2].getName().
    equals(player))
                    && (casillas[2][2].getName().equals(player))) {
                return true;
            }
            if ((casillas[0][0].getName().equals(player)) && (casillas[1][1].getName().
    equals(player))
                    && (casillas[2][2].getName().equals(player))) {
                return true;
            }
            if ((casillas[0][2].getName().equals(player)) && (casillas[1][1].getName().
    equals(player))
                    && (casillas[2][0].getName().equals(player))) {
                return true;
            }
            return false;
        }

        public int jugar(JButton botonPresionado) {

            juegaJugador(botonPresionado);

            if (validarJuego(jugador)) {
                return GANA;
            } else if (jugada < 9) {
                juegaMaquina();
                if (validarJuego(maquina)) {
                    return PIERDE;
                } else {
                    return SIGUE;
                }
            } else {
                return EMPATE;
            }
        }
    }
}
```