# UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS


## Facultad de Ingeniería Electrónica, Eléctrica y Telecomunicaciones

PROGRAMACIÓN ORIENTA A OBJETOS

Integrantes:

- Galvan Salvador Camila Lizbeth 18190328 (5pts)
- Gaspar Gabriel Jorge Victor 18190329 (5pts)
- Avila Huatuco Rihana Luz 18190338 (5pts)
- Ramirez Chavez Angela Camila 18190316 (0pts)
- Avalos Chino Maricielo 18190085 (5pts)
- Jhenry Rover Rosales Zuasnabar 18190343 (5pts)

Profesor: Herminio Paucar

2019

# Contenido

# JUEGO SIMPLE USANDO COMUNICACIÓN VÍA RED USANDO SOCKETS DE (Batalla Naval)

## Introducción

El siguiente proyecto nos hablara de un popular juego, que en sus inicios fue un juego de mesa, el cual combina conceptos de razonamiento lógicos y matemáticos, además de una estrategia naval. Hoy existen en el mercado diferentes versiones del juego, debido al avance de la tecnología, pero todas basadas en el mismo concepto. Nos referimos al juego de Batalla naval, el cual su objetivo principal es hundir la flota enemiga tratando de adivinar su ubicación en el tablero.

Lo novedoso de este juego en la actualidad, es la implementación de sockets, estos permiten a dos o más clientes/usuarios conectarse con un servidor que servirá de puente para la comunicación entre estos clientes.

Para esto, nuestro grupo trató de emular este juego, aplicando todos los métodos aprendidos en clase y de nuestra propia investigación.

## Historia

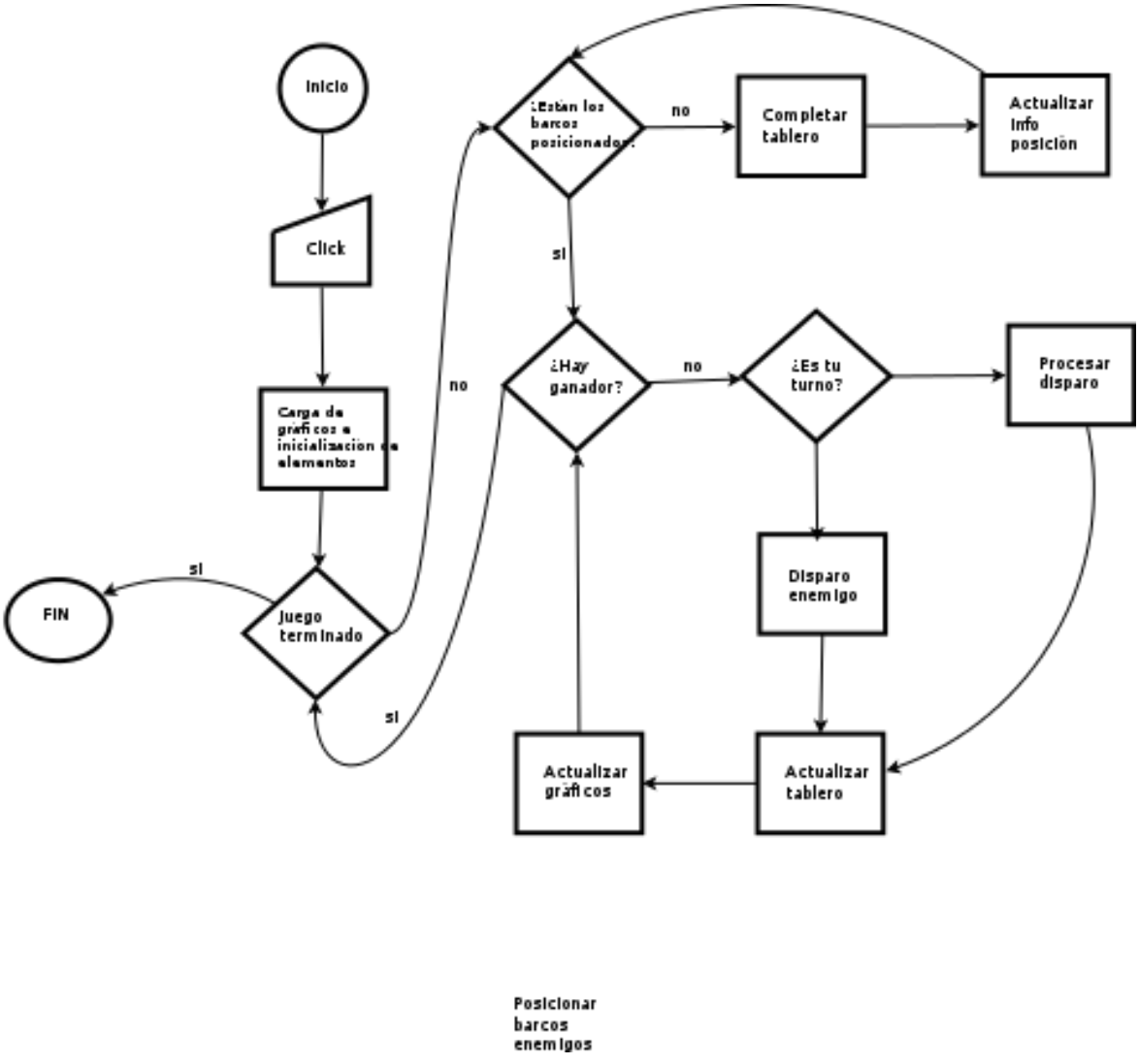
Este juego tiene sus orígenes en uno en lápiz y que se jugaba para los años 1900. Principalmente entre los soldados de la Primera Guerra Mundial. Para ese tiempo dibujaban una cuadrícula sobre un papel y ubicaban unos barcos sobre ella. Más tarde se creó un papel que salió al mercado para jugar un juego parecido al actual Battleship. En 1943 Milton Bradley lanzó el juego en una versión de lápiz y papel. Pero no fue hasta el 1967 que lo publicó tal y como lo conocemos. Fue recibido muy bien pues ya muchos conocían el concepto y verlo en una forma tridimensional fue toda una novedad. En 1977 salió al mercado una versión electrónica. Actualmente el derecho le pertenece a Hasbro.

## Requerimientos

| Type | Requirement | Priority |
|---|---|---|
| Funcional | El usuario se conecta al servidor mediante un IP o un puerto | Núcleo |
| Funcional | El juego permitirá a un solo jugador (por partida) cuyo contrincante será la maquina | Núcleo |
| Funcional | El juego permitirá al jugador insertar su nick (nombre opcional como desea identificarse) | Esencial |
| Funcional | El jugador tendrá la opción de iniciar o salir del juego | Esencial |
| Funcional | El juego mostrará dos tableros, el del izquierdo pertenece al jugador y el de lado derecho será de la máquina | Esencial |
| Funcional | El juego permitirá que el jugador ubique 5 barcos en el tablero | Esencial |
| Funcional | El juego permitirá al usuario posicionar a los barcos de forma horizontal haciendo 1 click para insertarlos | Núcleo |
| Funcional | El juego permitirá al usuario cambiar de una posición a otra empleando click derecho, siendo horizontal y vertical las únicas posiciones que podrán emplear | Esencial |
| Funcional | El tamaño de los barcos del jugador disminuirán de forma decreciente desde 5 casillas hasta 1 mientras los ubicamos en el tablero | Esencial |
| Funcional | El juego ubicará de forma aleatoria los barcos del rival del jugador siguiendo los mismos criterios | Esencial |
| Funcional | El juego permitirá al usuario escoger quien empieza la partida | Esencial |
| Funcional | Los barcos del rival no serán visibles para el jugador y viceversa | Deseado |
| Funcional | El juego permitirá al jugador un solo disparo por cada turno | Esencial |
| Funcional | El usuario hará clic en cualquier casilla del tablero contrario para atacar al rival y esperara a que el rival realice su ataque para volver a disparar. | Esencial |
| Funcional | La casilla seleccionada para el ataque se tornará de color amarillo si el ataque es asertivo (hirió a un barco del contrincante) | Esencial |
| Funcional | Si la casilla seleccionada resultase vacía se tornará de color celeste | Esencial |
| Funcional | En el momento en el que se hayan seleccionado todas las casillas de un barco rival, estas se tornarán de color rojo indicando de este modo que este barco se ha sido hundido | Esencial |
| Funcional | El juego terminará cuando todos los barcos ya sea del usuario o de la máquina se hayan hundido. | Esencial |
| Funcional | Al terminar la partida el juego mostrará un mensaje declarando si ganaste o perdiste. | Esencial |
| Funcional | El juego tendrá la opción de cancelar el juego a media partida | Esencial |

| | | |
|---|---|---|
| Funcional | Una vez de haber terminado el juego tendrás la opción de salir o iniciar otra partida | Esencial |
| Operacional | El juego mantendrá la relación cliente - servidor, este deberá conocer el IP y el puerto por el cual se comunicará | Núcleo |
| Técnico | Los archivos de portada y tablero deben estar en el almacenamiento, los cuales serán importados en el código | Núcleo |

## Diagrama de flujo



## Proceso del juego

- Combate naval es un juego para 2, cuyo objetivo consiste en destruir la flota de barcos de sus enemigos antes que él destruya todos los tuyos. En el diseño del juego se observa una ventana que muestra la imagen inicial del juego, luego pasa a mostrar, dentro de la misma ventana, dos tableros que son donde el servidor y el cliente ubicaran los barcos.

- El cliente ubicara 5 barcos de distintos tamaños en el tablero que le corresponde, y estos serán invisibles para el servidor (máquina), así mismo el servidor ubicara sus barcos de forma aleatoria y no serán visibles para el cliente.

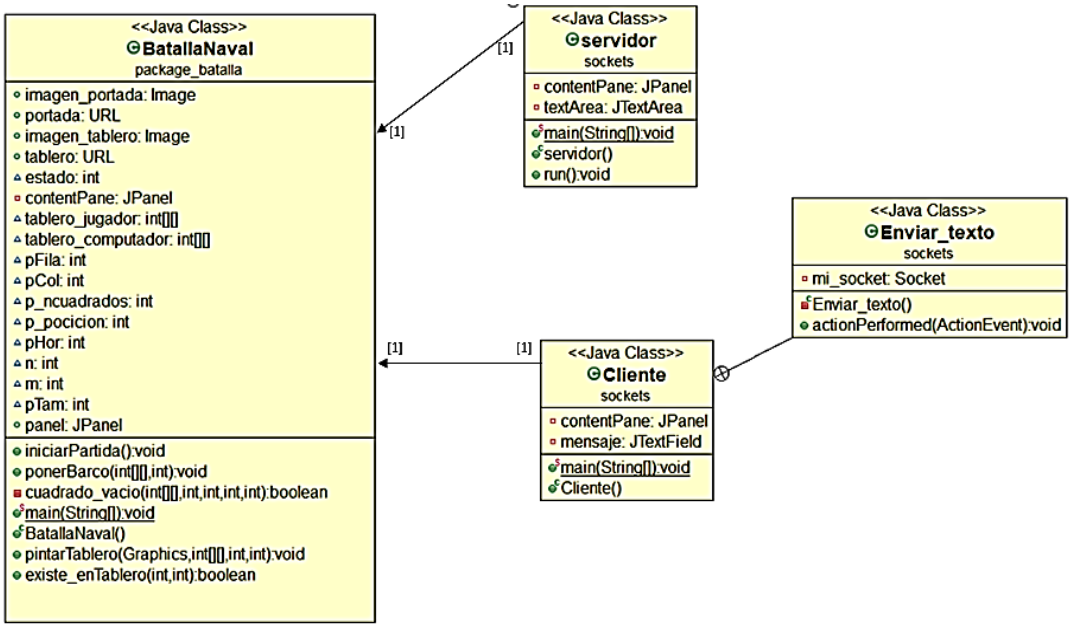
- Una vez que has posicionado todos tus barcos, el juego se desarrollará por turnos en donde debes disparar a las casillas de la zona de juego de tu oponente, con la finalidad de encontrar sus barcos escondidos.

- Si aciertas a una posición en donde hay un barco enemigo, la zona de juego rival deberá mostrar la casilla de color amarillo, esto nos dice que lograste disparar en una de las casillas de su barco. Sin embargo, si no logras disparar algún barco de tu oponente la casilla se mostrará de color celeste (que simula que disparate al agua). Finalmente, cuando logras dar a todas las casillas del barco de tu oponente estas se mostrarán de color rojo, que significa que lograste derribar el barco.

- El juego se repite mientras no se hayan hundido todos los barcos de alguno de los jugadores.

- El juego se finaliza cuando exista un ganador.

## Diagrama de clase




## Código Fuente

- **Clase BatallaNaval**

```
package batalla;

import java.awt.Color;
import java.awt.Container;
import java.awt.EventQueue;
import java.awt.Graphics;
import java.awt.Image;

import javax.swing.JFrame;
import javax.swing.JPanel;

import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import java.net.URL;

import javax.swing.ImageIcon;

@SuppressWarnings("serial")
public class BatallaNaval extends JFrame {

        public Image imagen_portada;
        public URL portada;
        public Image imagen_tablero;
```

```
public URL tablero;
int estado=0;
@SuppressWarnings("unused")
private JPanel contentPane;
int tablero_jugador[][]=new int[8][8];
int tablero_computador[][]=new int[8][8];

int pFila=0;
int pCol=0;
int p_ncuadrados=5;
int p_pocicion=0;
int pHor;
int n;
int m;
int pTam;

public void iniciarPartida(){
	for (int i=0; i<8;i++){
		for (int j=0; j<8;j++){
			tablero_jugador[i][j]=0;
			tablero_computador[i][j]=0;
		}
	}
	for (int tam=1; tam<=5;tam++){
		ponerBarco(tablero_computador, tam);
	}
	p_ncuadrados=5;
}

public void ponerBarco(int[][] tablero, int tam) {
	int i,j,hor;

	do{
		i=(int)Math.random()*8;

		j=(int)Math.random()*8;

		hor=(int)Math.random()*2;

	}while(!cuadrado_vacio(tablero,tam,i,j,hor));
	int x=0,y=0;
	if(hor==1){
		x=1;
	}else{
		y=1;
	}
	if(x==1){
		for(int i2=i;i2<=i+tam;i2++){
```

```
                                        tablero[i2][y]=tam;
                                }
                        }else{
                                for(int j2=j;j2<=j+tam;j2++){
                                        tablero[x][j2]=tam;
                                }
                        }
                }

                private boolean cuadrado_vacio(int[][] tablero, int tam, int i, int j, int hor) {
                        int x=0,y=0;
                        if(hor==1){
                                x=1;
                        }else{
                                y=1;
                        }
                        if(x==1){
                                for(int i2=i;i2<=i+tam;i2++){
                                        if(!existe_enTablero(i2,j)){
                                                return false;
                                        }
                                }
                        }else{
                                for(int j2=j;j2<=j+tam;j2++){
                                        if(!existe_enTablero(i,j2)){
                                                return false;
                                        }
                                }
                        }

                        for(int i2=i-1;i2<=i+1+x*tam;i2++){
                                for(int j2=j-1;j2<j+1+y*tam;j2++){
                                        if(existe_enTablero(i2,j2)& tablero[i2][j2]!=0){
                                                return false;
                                        }
                                }
                        }
                        return true;
                }

                public static void main(String[] args) {
                        EventQueue.invokeLater(new Runnable() {
                                public void run() {
                                        try {
                                                BatallaNaval frame = new BatallaNaval();
                                                frame.setVisible(true);
                                        } catch (Exception e) {
                                                e.printStackTrace();
```

```
                        }
                    }
                });
        }

        public BatallaNaval() {

                this.setBounds(100, 100, 900, 600);
                this.setTitle("Batalla Naval");
                this.setVisible(true);
                this.setLocationRelativeTo(null);

                portada=this.getClass().getResource("/batalla/portada.png");
                imagen_portada=new ImageIcon(portada).getImage();
                tablero=this.getClass().getResource("/batalla/tablero.jpg");
                imagen_tablero=new ImageIcon(tablero).getImage();

                Container contenedor=getContentPane();
                contenedor.add(panel);
                panel.setLayout(null);

                addMouseListener(
                        new MouseAdapter(){
                                public void mouseClicked(MouseEvent e){
                                        /**Al hacer click a la pantalla, la imagen cambia
de
                                   portada a tablero con un poco de retraso
                                   **/
                                        if (estado==0){
                                                estado=1;
                                                iniciarPartida();
                                                repaint();
                                        }
                                        if (estado==1){
                                                int pDF=0;
                                                int pDC=0;
                                                if (pHor==1){
                                                        pDF=1;
                                                }else{
                                                        pDC=1;
                                                }
                                                for(int n=pFila;n<=pFila+pTam*pDF;n++){
                                                        for (int
m=pCol;m<=pCol+pTam*pDC;m++){

        tablero_jugador[n][m]=pTam;
                                                        }
                                                }
```

```
                                        }

                                }
                        }
                );
                //el mouse no salga fuera del tablero
                addMouseMotionListener(
                        new MouseMotionAdapter(){
                                /**El tablero siente la presencia del mouse
                                 *
                                 */
                                @Override
                                public void mouseMoved(MouseEvent e){
                                        int x=e.getX();
                                        int y=e.getY();
                                        if(x>=115 && y>=250 && x<115+30*8 &&
y<250+30*8){
                                                int f=(y-200)/30;
                                                int c=(x-100)/30;
                                                if(f!=pFila || c!=pCol){
                                                        pFila=f;
                                                        pCol=c;
                                                        int pDF=0;
                                                        int pDC=0;
                                                        if (pHor==1){
                                                                pDF=1;
                                                        }else{
                                                                pDC=1;
                                                        }
                                                        if(pFila+pTam*pDF>=8){
                                                                pFila=8-pFila+pTam*pDC;
                                                        }if(pCol+pTam*pDF>=8){
                                                                pCol=8-pCol+pTam*pDC;
                                                        }
                                                        repaint();
                                                }
                                        }
                                }
                        }
                );
        }

        public JPanel panel=new JPanel(){
                public void paint(Graphics g){

                        if(estado==0){
                        g.drawImage(imagen_portada, 0,0, getWidth(), getHeight(), this);
                        }else{
```

```
                                g.drawImage(imagen_tablero, 0,0, getWidth(),
getHeight(), this);
                                pintarTablero(g, tablero_jugador,115,250);
                                pintarTablero(g, tablero_computador,525,250);

                        }
                }
        };

        public void pintarTablero(Graphics g, int[][] tablero, int i, int j) {
                for (int n=0; n<8;n++){
                        for (int m=0; m<8;m++){
                                if (tablero[n][m]>0){
                                        g.setColor(Color.RED);
                                        g.fillRect(i+n*30, j+m*30, 30, 30);
                                }
                                g.setColor(Color.BLACK);
                                g.drawRect(i+n*30, j+m*30, 30, 30);
                                }

                        if(estado==1){
                                int pDF=0;
                                int pDC=0;
                                if (pHor==1){
                                        pDF=1;
                                }else{
                                        pDC=1;
                                }

                                if (n>=pFila && m>=pCol &&n<=pFila + pTam*pDF &&
m<=pCol+pTam*pDC){
                                        g.setColor(Color.green);
                                        g.fillRect(i+n*30, j+m*30, 30, 30);
                                }
                        }
                        }
        }

        public boolean existe_enTablero(int i,int j){
                if (i<0) return false;
                if (j<0) return false;
                if (i>=8) return false;
                if (j>=8) return false;
        return true;
        }

}
```

- **Clase Cliente**

```
package batalla;

import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JButton;
import javax.swing.JTextField;

@SuppressWarnings("serial")
public class Cliente extends JFrame {

	private JPanel contentPane;
	private JTextField mensaje;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Cliente frame = new Cliente();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Cliente() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 450, 300);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);
```

```
                JButton btnNewButton = new JButton("Enviar");
                btnNewButton.setBounds(161, 79, 97, 25);
                Enviar_texto evento=new Enviar_texto();
                btnNewButton.addActionListener(evento);
                contentPane.add(btnNewButton);

                mensaje = new JTextField();
                mensaje.setBounds(152, 44, 116, 22);
                contentPane.add(mensaje);
                mensaje.setColumns(10);
        }

        private class Enviar_texto implements ActionListener{

                private Socket mi_socket;

                @Override
                public void actionPerformed(ActionEvent e) {
                        try {
                                mi_socket = new Socket("172.16.223.96",6666);

                                DataOutputStream flujo_salida = new
DataOutputStream(mi_socket.getOutputStream());

                                flujo_salida.writeUTF(mensaje.getText());
                                flujo_salida.close();


                        } catch (UnknownHostException e1) {
                                e1.printStackTrace();
                        } catch (IOException e1) {
                                //nos lanza un mensaje si falla la conexion
                                System.out.print(e1.getMessage());
                        }
                        System.out.println(mensaje.getText());

                }

        }

}
```

- **Clase Servidor**

```
package batalla;

import java.awt.EventQueue;
import java.io.DataInputStream;
import java.io.IOException;
```

```
import java.net.ServerSocket;
import java.net.Socket;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JTextArea;

@SuppressWarnings("serial")
public class Servidor extends JFrame implements Runnable {

	private JPanel contentPane;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Servidor frame = new Servidor();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Servidor() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 450, 300);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);

		textArea.setBounds(55, 13, 315, 227);
		contentPane.add(textArea);

		/**Creacion de un thread para que el socket reciba texto
		en el JTextArea,
		permanecer a la escucha y tener el puerto abierto**/
		Thread mi_hilo=new Thread(this);
		mi_hilo.start();
	}

	private JTextArea textArea=new JTextArea();

	@Override
	public void run() {
		try {
			@SuppressWarnings("resource")
			ServerSocket servidor =new ServerSocket(6666);
			while(true){
			Socket mi_socket=servidor.accept();
			DataInputStream flujo_entrada=new
DataInputStream(mi_socket.getInputStream());
```

```
                    String mensaje_entrada=flujo_entrada.readUTF();
                    textArea.append("\n"+mensaje_entrada);
                    mi_socket.close();
                    }
              } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              }

        }

}
```