Logotipo

Descripción generada automáticamenteUn dibujo de una persona

Descripción generada automáticamente con confianza bajaInstituto Tecnológico Superior de Calkiní en el Estado de Campeche

SCD-1027 Tópicos avanzados de programación

|| PRÁCTICA 4 Programación concurrente para conexión a un origen de datos ||

**Patrón de fondo

Descripción generada automáticamente**Ingeniería en sistemas computacionales

**Ing. José Luis**

**Lira Turriza**

**Diego Manuel Chi Tzab |7262|**

**4° Semestre Grupo A**

# PRÁCTICA No. 4.- Programación concurrente para conexión a un origen de datos.

#### INTRODUCCIÓN

La creación de hilos junto con las conexiones remotas permite interactuar a más de un usuario dentro del mismo sistema.

Un hilo es un hilo de ejecución en un programa. La JVM permite a una aplicación tener múltiples hilos de ejecución corriendo concurrentemente.

Cada hilo tiene una prioridad. Los Hilos con una prioridad alta son ejecutados sobre los hilos con baja prioridad. Cada hilo puede o puede que no también ser marcado como un “daemon”. Cuando el código corre algún hilo crea un nuevo objeto Thread, el nuevo hilo tiene su prioridad inicialmente igual a la prioridad del hilo que lo crea, y es un hilo “daemon” si y solo si el hilo que lo crea lo es.

Cuando una JVM comienza, hay usualmente un hilo simple “no-daemon” (el cual típicamente llama al método llamado main de alguna clase diseñada). La JVM continua ejecutando hilos hasta algo de lo siguiente ocurre:

El método exit de la clase Runtime es llamado y el administrador de seguridad ha permitido la operación de salida.

Todos los hilos que no son “daemon” han muerto, ya sea por regresando al método que llamó al método run o lanzando una excepción que se propaga más allá del método run.

Hay dos maneras para crear un nuevo hilo de ejecución. Uno es declarar una clase a ser una subclase de Thread. Esta subclase debería sobrescribir el método run de dicha clase. Una instancia de la subclase puede entonces ser localizada y comenzar.

La otra manera para crear un hilo es declarar una clase que implementa la interface Runnable. Esta clase implementa el método run. Una instancia de la clase puede entonces ser localizada, pasada como un argumento cuando se crea un objeto Thread e iniciar.

#### -OBJETIVO

Al finalizar la práctica el alumno será capaz de implementar hilos para realizar conexiones de red para una aplicación.

#### LUGAR

AULA

#### -SEMANA DE EJECUCIÓN

Semana NUEVE (Parcial 2)

#### MATERIAL Y EQUIPO

* + Sistema Operativo
  + Procesador de Textos
  + Software para el desarrollo de aplicaciones “eclipse”.
  + Cañón
  + Plumones
  + Pizarrón.
  + Equipos de cómputo para todos los estudiantes de la asignatura.

#### DESARROLLO DE LA PRÁCTICA

Battleship o Batalla Naval es un juego de lápiz y papel tradicional de adivinanza que involucra a dos participantes.

Se compone de dos tableros por jugador, dividido cada uno en cuadrículas. Los tableros típicos son cuadrados de 10 por 10 casillas, y cada posición se identifica con números para las columnas (de 1 a 10) y con letras para las filas (de la A a la J). En uno de los tableros el jugador coloca sus barcos y registra los tiros del oponente. En el otro, se registran los tiros propios. Antes de comenzar, cada jugador posiciona los barcos de forma secreta o invisible al oponente, generalmente con el tablero en posición vertical como pizarra. Cada uno ocupa, según su modelo, una cierta cantidad de posiciones ya sea horizontal o verticalmente. De esta forma, no se permiten lugares solapados, ya que cada uno ocupa posiciones únicas. Ambos participantes poseen y deben ubicar igual número de naves.

Una vez todas las naves han sido posicionadas, se inicia una serie de rondas. En cada ronda, cada jugador en su turno indica una posición del tablero de su oponente. Si esa posición es ocupada por una parte de un barco, el oponente indica averiado (toque o tocado) y el atacante marca con rojo esa posición, con un pin. Cuando todas las posiciones de un mismo barco han sido dañadas debe indicarse hundido dando a conocer tal circunstancia que indicará al atacante la importancia de la nave destruida. Ahora bien, si la posición indicada, efectivamente, no posee un barco alojado, se indica con agua, y será marcada con un pin blanco.

Quien descubra primero todas las naves será el vencedor, pero en caso de que el participante que comenzó la partida hunda el barco en su última jugada, el otro participante tiene una última posibilidad para alcanzar el empate.

1. Crea un paquete que contenga las clases necesarias para el control de la comunicación en la aplicación.
2. Adicionalmente crea un paquete que implemente la interfaz de usuario para la aplicación propuesta.
3. También usa un paquete que implemente la lógica del juego.
4. Ten en cuenta implementar un algoritmo para la versión automática de adversario.
5. Documenta toda la aplicación.

#### EVALUACIÓN Y RESULTADOS

#### La realización de está practica será por medio de la aplicación visual Studio Code para el código en java y Visual Studio para el lenguaje de c#

#### Este es un juego demasiado común en prácticamente todo el mundo, lo hayas jugado en la primaria o inclusive en la preparatoria, esto con ayuda de un pedazo de papel y bolígrafo o cualquier otro material que te ayudará a pintar, pero que pasa cuando una persona se cansa del método común y decide salirse de la rutina, por ello en está practica llevaremos este juego a un nivel superior esperando tener buenos resultados.

#### Código en Visual Studio Code : Lenguaje JAVA:

#### Primeramente debemos crear una clase principal, por así decir la base de todo el programa donde este se encargará de unir a todas las clases requeridas al igual de poder ejecutar el programa desde esta ventana sin tener que recorrer todo el código

|  |
| --- |
| import javax.swing.\*;  import java.awt.event.KeyEvent;  import java.awt.event.KeyListener;  public class Game implements KeyListener {        public static void main(String[] args) {          Game game = new Game();      }      private GamePanel gamePanel;      public Game() {          // Choose the AI Difficulty          String[] options = new String[] {"Fácil", "Medio", "Díficil"};          String message = "Fácil hará movimientos completamente al azar.\nMedio se centrará en las áreas donde encuentra barcos,"                  + "\ny Díficil tomará decisiones más inteligentes sobre Medio.";          int difficultyChoice = JOptionPane.showOptionDialog(null, message,                  "Elige una dificultad de IA",                  JOptionPane.DEFAULT\_OPTION, JOptionPane.PLAIN\_MESSAGE,                  null, options, options[0]);          JFrame frame = new JFrame("BATTLESHIP");          frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);          frame.setResizable(false);          gamePanel = new GamePanel(difficultyChoice);          frame.getContentPane().add(gamePanel);          frame.addKeyListener(this);          frame.pack();          frame.setVisible(true);      }      public void keyPressed(KeyEvent e) {          gamePanel.handleInput(e.getKeyCode());      }      public void keyTyped(KeyEvent e) {}      public void keyReleased(KeyEvent e) {}  } |

#### Para lo siguiente crearemos el panel del juego el cual nos ayudará a dar la base del juego (Tablero).

|  |
| --- |
| import javax.swing.\*;  import java.awt.\*;  import java.awt.event.KeyEvent;  import java.awt.event.MouseEvent;  import java.awt.event.MouseListener;  import java.awt.event.MouseMotionListener;  public class GamePanel extends JPanel implements MouseListener, MouseMotionListener {      public enum GameState { PlacingShips, FiringShots, GameOver }      private StatusPanel statusPanel;      private SelectionGrid computer;      private SelectionGrid player;      private BattleshipAI aiController;      private Ship placingShip;      private Position tempPlacingPosition;      private int placingShipIndex;      private GameState gameState;      public static boolean debugModeActive;      public GamePanel(int aiChoice) {          computer = new SelectionGrid(0,0);          player = new SelectionGrid(0,computer.getHeight()+50);          setBackground(new Color(139, 185, 221));          setPreferredSize(new Dimension(computer.getWidth(), player.getPosition().y + player.getHeight()));          addMouseListener(this);          addMouseMotionListener(this);          if(aiChoice == 0) aiController = new SimpleRandomAI(player);          else aiController = new SmarterAI(player,aiChoice == 2,aiChoice == 2);          statusPanel = new StatusPanel(new Position(0,computer.getHeight()+1),computer.getWidth(),49);          restart();      }      public void paint(Graphics g) {          super.paint(g);          computer.paint(g);          player.paint(g);          if(gameState == GameState.PlacingShips) {              placingShip.paint(g);          }          statusPanel.paint(g);      }      public void handleInput(int keyCode) {          if(keyCode == KeyEvent.VK\_ESCAPE) {              System.exit(1);          } else if(keyCode == KeyEvent.VK\_R) {              restart();          } else if(gameState == GameState.PlacingShips && keyCode == KeyEvent.VK\_Z) {              placingShip.toggleSideways();              updateShipPlacement(tempPlacingPosition);          } else if(keyCode == KeyEvent.VK\_D) {              debugModeActive = !debugModeActive;          }          repaint();      }      public void restart() {          computer.reset();          player.reset();          player.setShowShips(true);          aiController.reset();          tempPlacingPosition = new Position(0,0);          placingShip = new Ship(new Position(0,0),                                 new Position(player.getPosition().x,player.getPosition().y),                                 SelectionGrid.BOAT\_SIZES[0], true);          placingShipIndex = 0;          updateShipPlacement(tempPlacingPosition);          computer.populateShips();          debugModeActive = false;          statusPanel.reset();          gameState = GameState.PlacingShips;      }      private void tryPlaceShip(Position mousePosition) {          Position targetPosition = player.getPositionInGrid(mousePosition.x, mousePosition.y);          updateShipPlacement(targetPosition);          if(player.canPlaceShipAt(targetPosition.x, targetPosition.y,                  SelectionGrid.BOAT\_SIZES[placingShipIndex],placingShip.isSideways())) {              placeShip(targetPosition);          }      }      private void placeShip(Position targetPosition) {          placingShip.setShipPlacementColour(Ship.ShipPlacementColour.Placed);          player.placeShip(placingShip,tempPlacingPosition.x,tempPlacingPosition.y);          placingShipIndex++;          if(placingShipIndex < SelectionGrid.BOAT\_SIZES.length) {              placingShip = new Ship(new Position(targetPosition.x, targetPosition.y),                            new Position(player.getPosition().x + targetPosition.x \* SelectionGrid.CELL\_SIZE,                         player.getPosition().y + targetPosition.y \* SelectionGrid.CELL\_SIZE),                            SelectionGrid.BOAT\_SIZES[placingShipIndex], true);              updateShipPlacement(tempPlacingPosition);          } else {              gameState = GameState.FiringShots;              statusPanel.setTopLine("Ataca a la computadora!");              statusPanel.setBottomLine("Destruye todos los barcos para ganar!");          }      }      private void tryFireAtComputer(Position mousePosition) {          Position targetPosition = computer.getPositionInGrid(mousePosition.x,mousePosition.y);          if(!computer.isPositionMarked(targetPosition)) {              doPlayerTurn(targetPosition);              if(!computer.areAllShipsDestroyed()) {                  doAITurn();              }          }      }      private void doPlayerTurn(Position targetPosition) {          boolean hit = computer.markPosition(targetPosition);          String hitMiss = hit ? "IMPACTO" : "FALLÓ";          String destroyed = "";          if(hit && computer.getMarkerAtPosition(targetPosition).getAssociatedShip().isDestroyed()) {              destroyed = "(Barco Destruido)";          }          statusPanel.setTopLine("JUGADOR " + hitMiss + " " + targetPosition + destroyed);          if(computer.areAllShipsDestroyed()) {              // Player wins!              gameState = GameState.GameOver;              statusPanel.showGameOver(true);          }      }      private void doAITurn() {          Position aiMove = aiController.selectMove();          boolean hit = player.markPosition(aiMove);          String hitMiss = hit ? "IMPACTO" : "FALLÓ";          String destroyed = "";          if(hit && player.getMarkerAtPosition(aiMove).getAssociatedShip().isDestroyed()) {              destroyed = "(Barco Destruido)";          }          statusPanel.setBottomLine("COMPUTADORA " + hitMiss + " " + aiMove + destroyed);          if(player.areAllShipsDestroyed()) {              gameState = GameState.GameOver;              statusPanel.showGameOver(false);          }      }      private void tryMovePlacingShip(Position mousePosition) {          if(player.isPositionInside(mousePosition)) {              Position targetPos = player.getPositionInGrid(mousePosition.x, mousePosition.y);              updateShipPlacement(targetPos);          }      }      private void updateShipPlacement(Position targetPos) {            if(placingShip.isSideways()) {              targetPos.x = Math.min(targetPos.x, SelectionGrid.GRID\_WIDTH - SelectionGrid.BOAT\_SIZES[placingShipIndex]);          } else {              targetPos.y = Math.min(targetPos.y, SelectionGrid.GRID\_HEIGHT - SelectionGrid.BOAT\_SIZES[placingShipIndex]);          }          placingShip.setDrawPosition(new Position(targetPos),                                      new Position(player.getPosition().x + targetPos.x \* SelectionGrid.CELL\_SIZE,                                   player.getPosition().y + targetPos.y \* SelectionGrid.CELL\_SIZE));          tempPlacingPosition = targetPos;          if(player.canPlaceShipAt(tempPlacingPosition.x, tempPlacingPosition.y,                  SelectionGrid.BOAT\_SIZES[placingShipIndex],placingShip.isSideways())) {              placingShip.setShipPlacementColour(Ship.ShipPlacementColour.Valid);          } else {              placingShip.setShipPlacementColour(Ship.ShipPlacementColour.Invalid);          }      }      public void mouseReleased(MouseEvent e) {          Position mousePosition = new Position(e.getX(), e.getY());          if(gameState == GameState.PlacingShips && player.isPositionInside(mousePosition)) {              tryPlaceShip(mousePosition);          } else if(gameState == GameState.FiringShots && computer.isPositionInside(mousePosition)) {              tryFireAtComputer(mousePosition);          }          repaint();      }      public void mouseMoved(MouseEvent e) {          if(gameState != GameState.PlacingShips) return;          tryMovePlacingShip(new Position(e.getX(), e.getY()));          repaint();      }      public void mouseClicked(MouseEvent e) {}      public void mousePressed(MouseEvent e) {}      public void mouseEntered(MouseEvent e) {}      public void mouseExited(MouseEvent e) {}      public void mouseDragged(MouseEvent e) {}  } |

#### Seguido crearemos la forma que tendrá el juego, en nuestro caso será un Rectángulo, ya que nos permitirá un mejor aprovechamiento a la hora del juego

|  |
| --- |
| public class Rectangle {      protected Position position;      protected int width;        protected int height;      public Rectangle(Position position, int width, int height) {          this.position = position;          this.width = width;          this.height = height;      }      public Rectangle(int x, int y, int width, int height) {          this(new Position(x,y),width,height);      }      public int getHeight() {          return height;      }      public int getWidth() {          return width;      }      public Position getPosition() {          return position;      }      public boolean isPositionInside(Position targetPosition) {          return targetPosition.x >= position.x && targetPosition.y >= position.y                  && targetPosition.x < position.x + width && targetPosition.y < position.y + height;      }  } |

#### Seguidamente viene la parte más interesante, lo que es la creación del barco, porque no puede ser un juego Battleship sin la característica principal del juego que son los barcos.

|  |
| --- |
| import java.awt.\*;  import java.util.ArrayList;  import java.util.List;  public class Ship {      public enum ShipPlacementColour {Valid, Invalid, Placed}      private Position gridPosition;        private Position drawPosition;        private int segments;        private boolean isSideways;        private int destroyedSections;        private ShipPlacementColour shipPlacementColour;      public Ship(Position gridPosition, Position drawPosition, int segments, boolean isSideways) {          this.gridPosition = gridPosition;          this.drawPosition = drawPosition;          this.segments = segments;          this.isSideways = isSideways;          destroyedSections = 0;          shipPlacementColour = ShipPlacementColour.Placed;      }      public void paint(Graphics g) {          if(shipPlacementColour == ShipPlacementColour.Placed) {              g.setColor(destroyedSections >= segments ? Color.RED : Color.DARK\_GRAY);          } else {              g.setColor(shipPlacementColour == ShipPlacementColour.Valid ? Color.GREEN : Color.RED);          }          if(isSideways) paintHorizontal(g);          else paintVertical(g);      }      public void setShipPlacementColour(ShipPlacementColour shipPlacementColour) {          this.shipPlacementColour = shipPlacementColour;      }      public void toggleSideways() {          isSideways = !isSideways;      }      public void destroySection() {          destroyedSections++;      }      public boolean isDestroyed() { return destroyedSections >= segments; }      public void setDrawPosition(Position gridPosition, Position drawPosition) {          this.drawPosition = drawPosition;          this.gridPosition = gridPosition;      }      public boolean isSideways() {          return isSideways;      }      public int getSegments() {          return segments;      }      public List<Position> getOccupiedCoordinates() {          List<Position> result = new ArrayList<>();          if(isSideways) {              for(int x = 0; x < segments; x++) {                  result.add(new Position(gridPosition.x+x, gridPosition.y));              }          } else {              for(int y = 0; y < segments; y++) {                  result.add(new Position(gridPosition.x, gridPosition.y+y));              }          }          return result;      }      public void paintVertical(Graphics g) {          int boatWidth = (int)(SelectionGrid.CELL\_SIZE \* 0.8);          int boatLeftX = drawPosition.x + SelectionGrid.CELL\_SIZE / 2 - boatWidth / 2;          g.fillPolygon(new int[]{drawPosition.x+SelectionGrid.CELL\_SIZE/2,boatLeftX,boatLeftX+boatWidth},                  new int[]{drawPosition.y+SelectionGrid.CELL\_SIZE/4,drawPosition.y+SelectionGrid.CELL\_SIZE,drawPosition.y+SelectionGrid.CELL\_SIZE},3);          g.fillRect(boatLeftX,drawPosition.y+SelectionGrid.CELL\_SIZE, boatWidth,                  (int)(SelectionGrid.CELL\_SIZE \* (segments-1.2)));      }      public void paintHorizontal(Graphics g) {          int boatWidth = (int)(SelectionGrid.CELL\_SIZE \* 0.8);          int boatTopY = drawPosition.y + SelectionGrid.CELL\_SIZE / 2 - boatWidth / 2;          g.fillPolygon(new int[]{drawPosition.x+SelectionGrid.CELL\_SIZE/4,drawPosition.x+SelectionGrid.CELL\_SIZE,drawPosition.x+SelectionGrid.CELL\_SIZE},                        new int[]{drawPosition.y+SelectionGrid.CELL\_SIZE/2,boatTopY,boatTopY+boatWidth},3);          g.fillRect(drawPosition.x+SelectionGrid.CELL\_SIZE,boatTopY,                  (int)(SelectionGrid.CELL\_SIZE \* (segments-1.2)), boatWidth);      }  } |

#### Una vez obtenido el barco procederemos a ubicarlo dentro de alguna de nuestras casillas del juego.

|  |
| --- |
| public class Position {      public static final Position DOWN = new Position(0,1);        public static final Position UP = new Position(0,-1);        public static final Position LEFT = new Position(-1,0);        public static final Position RIGHT = new Position(1,0);      public static final Position ZERO = new Position(0,0);      public int x;      public int y;      public Position(int x, int y) {          this.x = x;          this.y = y;      }      public Position(Position positionToCopy) {          this.x = positionToCopy.x;          this.y = positionToCopy.y;      }      public void setPosition(int x, int y) {          this.x = x;          this.y = y;      }      public void add(Position otherPosition) {          this.x += otherPosition.x;          this.y += otherPosition.y;      }      public double distanceTo(Position otherPosition) {          return Math.sqrt(Math.pow(x-otherPosition.x,2)+Math.pow(y-otherPosition.y,2));      }      public void multiply(int amount) {          x \*= amount;          y \*= amount;      }      public void subtract(Position otherPosition) {          this.x -= otherPosition.x;          this.y -= otherPosition.y;      }      public boolean equals(Object o) {          if (this == o) return true;          if (o == null || getClass() != o.getClass()) return false;          Position position = (Position) o;          return x == position.x && y == position.y;      }      public String toString() {          return "(" + x + ", " + y + ")";      }  } |

#### Para lo siguiente teniendo en cuenta que ya tenemos todos los recursos casi listos es la parte de la selección del barco enemigo, porque claro no sería algo justo que no podamos atacar, el juego no tendría algún sentido si fuera así, por lo cual usaremos esta clase con su respectiva función.

|  |
| --- |
| import java.awt.\*;  import java.util.ArrayList;  import java.util.List;  import java.util.Random;  public class SelectionGrid extends Rectangle {      public static final int CELL\_SIZE = 30;        public static final int GRID\_WIDTH = 10;        public static final int GRID\_HEIGHT = 10;        public static final int[] BOAT\_SIZES = {5,4,3,3,2};      private Marker[][] markers = new Marker[GRID\_WIDTH][GRID\_HEIGHT];        private List<Ship> ships;        private Random rand;      private boolean showShips;        private boolean allShipsDestroyed;      public SelectionGrid(int x, int y) {          super(x, y, CELL\_SIZE\*GRID\_WIDTH, CELL\_SIZE\*GRID\_HEIGHT);          createMarkerGrid();          ships = new ArrayList<>();          rand = new Random();          showShips = false;      }      public void paint(Graphics g) {          for(Ship ship : ships) {              if(showShips || GamePanel.debugModeActive || ship.isDestroyed()) {                  ship.paint(g);              }          }          drawMarkers(g);          drawGrid(g);      }      public void setShowShips(boolean showShips) {          this.showShips = showShips;      }      public void reset() {          for(int x = 0; x < GRID\_WIDTH; x++) {              for(int y = 0; y < GRID\_HEIGHT; y++) {                  markers[x][y].reset();              }          }          ships.clear();          showShips = false;          allShipsDestroyed = false;      }      public boolean markPosition(Position posToMark) {          markers[posToMark.x][posToMark.y].mark();          allShipsDestroyed = true;          for(Ship ship : ships) {              if(!ship.isDestroyed()) {                  allShipsDestroyed = false;                  break;              }          }          return markers[posToMark.x][posToMark.y].isShip();      }      public boolean areAllShipsDestroyed() {          return allShipsDestroyed;      }      public boolean isPositionMarked(Position posToTest) {          return markers[posToTest.x][posToTest.y].isMarked();      }      public Marker getMarkerAtPosition(Position posToSelect) {          return markers[posToSelect.x][posToSelect.y];      }      public Position getPositionInGrid(int mouseX, int mouseY) {          if(!isPositionInside(new Position(mouseX,mouseY))) return new Position(-1,-1);          return new Position((mouseX - position.x)/CELL\_SIZE, (mouseY - position.y)/CELL\_SIZE);      }      public boolean canPlaceShipAt(int gridX, int gridY, int segments, boolean sideways) {          if(gridX < 0 || gridY < 0) return false;          if(sideways) { // handle the case when horizontal              if(gridY > GRID\_HEIGHT || gridX + segments > GRID\_WIDTH) return false;              for(int x = 0; x < segments; x++) {                  if(markers[gridX+x][gridY].isShip()) return false;              }          } else { // handle the case when vertical              if(gridY + segments > GRID\_HEIGHT || gridX > GRID\_WIDTH) return false;              for(int y = 0; y < segments; y++) {                  if(markers[gridX][gridY+y].isShip()) return false;              }          }          return true;      }      private void drawGrid(Graphics g) {          g.setColor(Color.BLACK);          int y2 = position.y;          int y1 = position.y+height;          for(int x = 0; x <= GRID\_WIDTH; x++)              g.drawLine(position.x+x \* CELL\_SIZE, y1, position.x+x \* CELL\_SIZE, y2);          int x2 = position.x;          int x1 = position.x+width;          for(int y = 0; y <= GRID\_HEIGHT; y++)              g.drawLine(x1, position.y+y \* CELL\_SIZE, x2, position.y+y \* CELL\_SIZE);      }      private void drawMarkers(Graphics g) {          for(int x = 0; x < GRID\_WIDTH; x++) {              for(int y = 0; y < GRID\_HEIGHT; y++) {                  markers[x][y].paint(g);              }          }      }      private void createMarkerGrid() {          for(int x = 0; x < GRID\_WIDTH; x++) {              for (int y = 0; y < GRID\_HEIGHT; y++) {                  markers[x][y] = new Marker(position.x+x\*CELL\_SIZE, position.y + y\*CELL\_SIZE, CELL\_SIZE, CELL\_SIZE);              }          }      }      public void populateShips() {          ships.clear();          for(int i = 0; i < BOAT\_SIZES.length; i++) {              boolean sideways = rand.nextBoolean();              int gridX,gridY;              do {                  gridX = rand.nextInt(sideways?GRID\_WIDTH-BOAT\_SIZES[i]:GRID\_WIDTH);                  gridY = rand.nextInt(sideways?GRID\_HEIGHT:GRID\_HEIGHT-BOAT\_SIZES[i]);              } while(!canPlaceShipAt(gridX,gridY,BOAT\_SIZES[i],sideways));              placeShip(gridX, gridY, BOAT\_SIZES[i], sideways);          }      }      public void placeShip(int gridX, int gridY, int segments, boolean sideways) {          placeShip(new Ship(new Position(gridX, gridY),                             new Position(position.x+gridX\*CELL\_SIZE, position.y+gridY\*CELL\_SIZE),                              segments, sideways), gridX, gridY);      }      public void placeShip(Ship ship, int gridX, int gridY) {          ships.add(ship);          if(ship.isSideways()) {              for(int x = 0; x < ship.getSegments(); x++) {                  markers[gridX+x][gridY].setAsShip(ships.get(ships.size()-1));              }          } else {              for(int y = 0; y < ship.getSegments(); y++) {                  markers[gridX][gridY+y].setAsShip(ships.get(ships.size()-1));              }          }      }  } |

#### Una vez tenido la selección, proseguiremos a un identificador que nos diga si le dimos al barco o no, por lo cual usaremos lo siguiente.

|  |
| --- |
| import java.awt.\*;  public class Marker extends Rectangle {      private final Color HIT\_COLOUR = new Color(219, 23, 23, 180);      private final Color MISS\_COLOUR = new Color(26, 26, 97, 180);      private final int PADDING = 3;      private boolean showMarker;      private Ship shipAtMarker;      public Marker(int x, int y, int width, int height) {          super(x, y, width, height);          reset();      }      public void reset() {          shipAtMarker = null;          showMarker = false;      }      public void mark() {          if(!showMarker && isShip()) {              shipAtMarker.destroySection();          }          showMarker = true;      }      public boolean isMarked() {          return showMarker;      }      public void setAsShip(Ship ship) {          this.shipAtMarker = ship;      }      public boolean isShip() {          return shipAtMarker != null;      }      public Ship getAssociatedShip() {          return shipAtMarker;      }      public void paint(Graphics g) {          if(!showMarker) return;          g.setColor(isShip() ? HIT\_COLOUR : MISS\_COLOUR);          g.fillRect(position.x+PADDING+1, position.y+PADDING+1, width-PADDING\*2, height-PADDING\*2);      }  } |

#### Para lo siguiente necesitamos una clase que nos diga el status del panel, si todo, donde nos diga como estamos, si ya ganamos o simplemente si le atinamos al barco rival

|  |
| --- |
| import java.awt.\*;  public class StatusPanel extends Rectangle{      private final Font font = new Font("Arial", Font.BOLD, 15);      private final String placingShipLine1 = "¡Coloca tus barcos!";      private final String placingShipLine2 = "Z para rotar el barco.";      private final String gameOverLossLine = "¡Juego terminado! Perdiste :(";      private final String gameOverWinLine = "¡Ganaste! Bien hecho!";        private final String gameOverBottomLine = "Presiona R para reiniar el juego.";      private String topLine;      private String bottomLine;      public StatusPanel(Position position, int width, int height) {          super(position, width, height);          reset();      }      public void reset() {          topLine = placingShipLine1;          bottomLine = placingShipLine2;      }      public void showGameOver(boolean playerWon) {          topLine = (playerWon) ? gameOverWinLine : gameOverLossLine;          bottomLine = gameOverBottomLine;      }      public void setTopLine(String message) {          topLine = message;      }      public void setBottomLine(String message) {          bottomLine = message;      }      public void paint(Graphics g) {          g.setColor(Color.DARK\_GRAY);          g.fillRect(position.x, position.y, width, height);          g.setColor(Color.WHITE);          g.setFont(font);          int strWidth = g.getFontMetrics().stringWidth(topLine);          g.drawString(topLine, position.x+width/2-strWidth/2, position.y+20);          strWidth = g.getFontMetrics().stringWidth(bottomLine);          g.drawString(bottomLine, position.x+width/2-strWidth/2, position.y+40);      }  } |

#### Considerando que este juego estará en tu computadora y simplemente no tienes amigos para jugar, hemos decidido hacer que la computadora juegue contigo para que no estes tan solo.

|  |
| --- |
| import java.util.ArrayList;  import java.util.List;  public class BattleshipAI {      protected SelectionGrid playerGrid;        protected List<Position> validMoves;      public BattleshipAI(SelectionGrid playerGrid) {          this.playerGrid = playerGrid;          createValidMoveList();      }      public Position selectMove() {          return Position.ZERO;      }      public void reset() {          createValidMoveList();      }      private void createValidMoveList() {          validMoves = new ArrayList<>();          for(int x = 0; x < SelectionGrid.GRID\_WIDTH; x++) {              for(int y = 0; y < SelectionGrid.GRID\_HEIGHT; y++) {                  validMoves.add(new Position(x,y));              }          }      }  } |

#### Lo siguiente será hacer que la computadora haga movimientos para que el juego sea entretenido y no solo te haga compañía, si no que te haga frente

|  |
| --- |
| import java.util.ArrayList;  import java.util.Collections;  import java.util.List;  public class SmarterAI extends BattleshipAI {      private List<Position> shipHits;        private final boolean debugAI = false;        private boolean preferMovesFormingLine;      private boolean maximiseAdjacentRandomisation;      public SmarterAI(SelectionGrid playerGrid, boolean preferMovesFormingLine, boolean maximiseAdjacentRandomisation) {          super(playerGrid);          shipHits = new ArrayList<>();          this.preferMovesFormingLine = preferMovesFormingLine;          this.maximiseAdjacentRandomisation = maximiseAdjacentRandomisation;          Collections.shuffle(validMoves);      }      public void reset() {          super.reset();          shipHits.clear();          Collections.shuffle(validMoves);      }      public Position selectMove() {          if(debugAI) System.out.println("\nEMPEZAR TURNO===========");          Position selectedMove;          if(shipHits.size() > 0) {              if(preferMovesFormingLine) {                  selectedMove = getSmarterAttack();              } else {                  selectedMove = getSmartAttack();              }          } else {              if(maximiseAdjacentRandomisation) {                  selectedMove = findMostOpenPosition();              } else {                  selectedMove = validMoves.get(0);              }          }          updateShipHits(selectedMove);          validMoves.remove(selectedMove);          if(debugAI) {              System.out.println("Selected Move: " + selectedMove);              System.out.println("TURNO FINALIZADO===========");          }          return selectedMove;      }      private Position getSmartAttack() {          List<Position> suggestedMoves = getAdjacentSmartMoves();          Collections.shuffle(suggestedMoves);          return  suggestedMoves.get(0);      }      private Position getSmarterAttack() {          List<Position> suggestedMoves = getAdjacentSmartMoves();          for(Position possibleOptimalMove : suggestedMoves) {              if(atLeastTwoHitsInDirection(possibleOptimalMove,Position.LEFT)) return possibleOptimalMove;              if(atLeastTwoHitsInDirection(possibleOptimalMove,Position.RIGHT)) return possibleOptimalMove;              if(atLeastTwoHitsInDirection(possibleOptimalMove,Position.DOWN)) return possibleOptimalMove;              if(atLeastTwoHitsInDirection(possibleOptimalMove,Position.UP)) return possibleOptimalMove;          }            Collections.shuffle(suggestedMoves);          return  suggestedMoves.get(0);      }      private Position findMostOpenPosition() {          Position position = validMoves.get(0);;          int highestNotAttacked = -1;          for(int i = 0; i < validMoves.size(); i++) {              int testCount = getAdjacentNotAttackedCount(validMoves.get(i));              if(testCount == 4) {                  return validMoves.get(i);              } else if(testCount > highestNotAttacked) {                  highestNotAttacked = testCount;                  position = validMoves.get(i);              }          }          return position;      }      private int getAdjacentNotAttackedCount(Position position) {          List<Position> adjacentCells = getAdjacentCells(position);          int notAttackedCount = 0;          for(Position adjacentCell : adjacentCells) {              if(!playerGrid.getMarkerAtPosition(adjacentCell).isMarked()) {                  notAttackedCount++;              }          }          return notAttackedCount;      }      private boolean atLeastTwoHitsInDirection(Position start, Position direction) {          Position testPosition = new Position(start);          testPosition.add(direction);          if(!shipHits.contains(testPosition)) return false;          testPosition.add(direction);          if(!shipHits.contains(testPosition)) return false;          if(debugAI) System.out.println("Smarter match found AT: " + start + " TO: " + testPosition);          return true;      }      private List<Position> getAdjacentSmartMoves() {          List<Position> result = new ArrayList<>();          for(Position shipHitPos : shipHits) {              List<Position> adjacentPositions = getAdjacentCells(shipHitPos);              for(Position adjacentPosition : adjacentPositions) {                  if(!result.contains(adjacentPosition) && validMoves.contains(adjacentPosition)) {                      result.add(adjacentPosition);                  }              }          }          if(debugAI) {              printPositionList("Ship Hits: ", shipHits);              printPositionList("Adjacent Smart Moves: ", result);          }          return result;      }      private void printPositionList(String messagePrefix, List<Position> data) {          String result = "[";          for(int i = 0; i < data.size(); i++) {              result += data.get(i);              if(i != data.size()-1) {                  result += ", ";              }          }          result += "]";          System.out.println(messagePrefix + " " + result);      }      private List<Position> getAdjacentCells(Position position) {          List<Position> result = new ArrayList<>();          if(position.x != 0) {              Position left = new Position(position);              left.add(Position.LEFT);              result.add(left);          }          if(position.x != SelectionGrid.GRID\_WIDTH-1) {              Position right = new Position(position);              right.add(Position.RIGHT);              result.add(right);          }          if(position.y != 0) {              Position up = new Position(position);              up.add(Position.UP);              result.add(up);          }          if(position.y != SelectionGrid.GRID\_HEIGHT-1) {              Position down = new Position(position);              down.add(Position.DOWN);              result.add(down);          }          return result;      }      private void updateShipHits(Position testPosition) {          Marker marker = playerGrid.getMarkerAtPosition(testPosition);          if(marker.isShip()) {              shipHits.add(testPosition);              List<Position> allPositionsOfLastShip = marker.getAssociatedShip().getOccupiedCoordinates();              if(debugAI) printPositionList("Last Ship", allPositionsOfLastShip);              boolean hitAllOfShip = containsAllPositions(allPositionsOfLastShip, shipHits);              if(hitAllOfShip) {                  for(Position shipPosition : allPositionsOfLastShip) {                      for(int i = 0; i < shipHits.size(); i++) {                          if(shipHits.get(i).equals(shipPosition)) {                              shipHits.remove(i);                              if(debugAI) System.out.println("Removed " + shipPosition);                              break;                          }                      }                  }              }          }      }      private boolean containsAllPositions(List<Position> positionsToSearch, List<Position> listToSearchIn) {          for(Position searchPosition : positionsToSearch) {              boolean found = false;              for(Position searchInPosition : listToSearchIn) {                  if(searchInPosition.equals(searchPosition)) {                      found = true;                      break;                  }              }              if(!found) return false;          }          return true;      }  } |

#### Ya por ultimo agregamos todo en una clase para que ejecute todas las funciones y ya podríamos probar este maravilloso juego.

|  |
| --- |
| import java.util.Collections;  public class SimpleRandomAI extends BattleshipAI{      public SimpleRandomAI(SelectionGrid playerGrid) {          super(playerGrid);          Collections.shuffle(validMoves);      }      public void reset() {          super.reset();          Collections.shuffle(validMoves);      }      public Position selectMove() {          Position nextMove = validMoves.get(0);          validMoves.remove(0);          return nextMove;      }  } |

#### Código en Visual Studio : Lenguaje C# :

#### Primeramente crearemos la ventana principal, la cual será lo que nos aparecerá al iniciar el juego, c# nos permite hacer esto gráficamente por lo es de gran ayuda

|  |
| --- |
|  |

#### Procedemos a darle nombre y funcionalidad a cada botón en esta sección

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Threading;  using System.Windows.Forms;  namespace BattleshipPractica4  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  private void BtNv\_Click(object sender, EventArgs e)  {  }  private void button1\_Click(object sender, EventArgs e)  {  Form2 jg = new Form2();  button1.Text = "¿OTRO INTENTO?";  jg.ShowDialog();  }  private void button2\_Click(object sender, EventArgs e)  {  MessageBox.Show("Inicialmente, el jugador deberá elegir la posición de sus cuatro barcos haciendo clic en los botones de la mesa. Gana el que consiga más muertes de barcos en un ciclo de 10 rondas. Los enemigos pueden ganar puntos al retrasar el movimiento del jugador, por lo que el juego tendrá algunas dificultades. ");  }  private void Form1\_Load(object sender, EventArgs e)  {  }  private void button3\_Click(object sender, EventArgs e)  {  Environment.Exit(1);  }  } } |

#### Procedemos a agregar nuevo elemento el cual será el tablero y donde se desarrollará todos los movimientos, una vez agregado el nuevo elemento procederemos a darle una vista agradable

|  |
| --- |
|  |

#### Ahora procederemos a darle funcionalidad a cada elemento dentro del recuadro, hay que aplicar bastantes funcionalidades ya que este será el juego que queremos recrear.

|  |
| --- |
| using System;using System.Collections.Generic;using System.ComponentModel;using System.Data;using System.Drawing;using System.Linq;using System.Text;using System.Threading.Tasks;using System.Windows.Forms;using System.Diagnostics;using BattleshipPractica4.Properties;namespace BattleshipPractica4{public partial class Form2 : Form{public static int RODADA = 1;public static int barcosP = 5;public static int barcosE = 5;int rodadas = 10;Random aleatorio = new Random();public static int pontuacionP = 0;public static int pontuacionE = 0;List<Button> botonesjugador;List<Button> botonesenemigos;//criando numero aleatóriopublic Form2(){InitializeComponent();barcosP = 5;RODADA = 1;pontuacionE = 0;pontuacionP = 0;this.DoubleBuffered = true;botones();alo.Text = null;inimigoCasas.Enabled = false;inimigoCasas.Text = null;textoAuxiliar.Text = "Poscionamiento de su navio clickeado los botones";alo.Text = "Rondas: " + rodadas;PONTUACAOE.Text = "Puntuación enemiga: "+pontuacionE;PONTUACAOP.Text = "Su puntuación: " + pontuacionP;}private void Form1\_Load(object sender, EventArgs e){}private void button32\_Click(object sender, EventArgs e){}private void label2\_Click(object sender, EventArgs e){}private void button1\_Click(object sender, EventArgs e){}private void tempoJogadaInimigo\_Tick(object sender, EventArgs e){}private void Btn\_a2\_Click(object sender, EventArgs e){}private void button1\_Click\_1(object sender, EventArgs e){}private void AtaqueBot(object sender, EventArgs e){if (rodadas == 0){if (pontuacionE > pontuacionP){MessageBox.Show("RIVAL HA GANADO, HAS PERDIDO!!" +" :( ");inimigoCasas.Enabled = false;Environment.Exit(1);}else if (pontuacionP > pontuacionE){MessageBox.Show("RIVAL HA PERDIDO, LA VICTORIA ES TUYA!!" +" :) ");inimigoCasas.Enabled = false;Environment.Exit(1);}else{MessageBox.Show("EMPATE");inimigoCasas.Enabled = false;Environment.Exit(1);}}for (int i = 0; i < 1; i++){if (RODADA == 0){int resultado = aleatorio.Next(botonesenemigos.Count);if (botonesenemigos[resultado].Tag == "NAVIO"){botonesenemigos[resultado].Tag = "";botonesenemigos[resultado].BackgroundImage = Properties.Resources.bombardeiroEncontrado;pontuacionE++;PONTUACAOE.Text = "Puntuación enemiga: " + pontuacionE;inimigoCasas.Enabled = true;rodadas--;alo.Text = "Rondas: " + rodadas;}else {botonesenemigos[resultado].BackgroundImage = Properties.Resources.ONDA;inimigoCasas.Enabled = true;rodadas--;alo.Text = "Rondas: " + rodadas;}}}RODADA = 1;}private void inimigoCasas\_SelectedIndexChanged(object sender, EventArgs e){if (inimigoCasas.Text != ""){if (rodadas == 0){if (pontuacionE > pontuacionP){MessageBox.Show("RIVAL HA GANADO, HAS PERDIDO!!" +" :( ");inimigoCasas.Enabled = false;Environment.Exit(1);}else if (pontuacionP > pontuacionE){MessageBox.Show("RIVAL HA PERDIDO, LA VICTORIA ES TUYA!!" +" :) ");inimigoCasas.Enabled = false;Environment.Exit(1);}else{MessageBox.Show("EMPATE");inimigoCasas.Enabled = false;Environment.Exit(1);}}var escolha = inimigoCasas.Text;escolha = escolha.ToLower();int coordenada = botonesjugador.FindIndex(a => a.Name == escolha);if (botonesjugador[coordenada].Enabled){if (botonesjugador[coordenada].Tag == "NAVIO"){pontuacionP++;PONTUACAOP.Text = "Su Puntuación: " + pontuacionP;botonesjugador[coordenada].Enabled = false;botonesjugador[coordenada].BackgroundImage = Properties.Resources.bombardeiroEncontrado;inimigoCasas.Enabled = false; RODADA = 0;RODADA = 0;AtaqueBot(sender,e);}else{botonesjugador[coordenada].Enabled = false;botonesjugador[coordenada].BackgroundImage = Properties.Resources.ONDA;inimigoCasas.Enabled = false; RODADA = 0;AtaqueBot(sender, e);}}}else{textoAuxiliar.Text = "Elija una casilla existente";}}private void jogadorTableLayout\_Paint(object sender, PaintEventArgs e){}private void posiçãodeataqueinimigo(object sender, EventArgs e){}private void botones(){botonesjugador = new List<Button> { a1, a2, a3, a4, a5, this.b1, this.b2, this.b3, this.b4, this.b5, c1, c3, c2, c4, c5, d1, d2, d3, d4, d5, e1, e2, e3, e4, e5};botonesenemigos = new List<Button> { BtnBot\_a1, BtnBot\_a2, BtnBot\_a3, BtnBot\_a4, BtnBot\_a5, BtnBot\_b1, BtnBot\_b2, BtnBot\_b3, BtnBot\_b4, BtnBot\_b5, BtnBot\_c1, BtnBot\_c2, BtnBot\_c3, BtnBot\_c4, BtnBot\_c5, BtnBot\_d1, BtnBot\_d2, BtnBot\_d3, BtnBot\_d4 , BtnBot\_d5,BtnBot\_e1, BtnBot\_e2, BtnBot\_e3, BtnBot\_e4, BtnBot\_e5};for (int i = 0; i < botonesjugador.Count; i++){botonesjugador[i].Tag = i;}for (int i = 0; i < botonesenemigos.Count; i++){botonesenemigos[i].Tag = null;inimigoCasas.Items.Add(botonesenemigos[i].Text);}// PARTE MAIS ARCAICA DO PROGAMARandom aleatorio = new Random();int b1 = 0; int b2 = 0; int b3 = 0; int b4 = 0; int b5 = 0; int l1 = 0; int l2 = 0; int l3 = 0; int l4 = 0; int l5 = 0;for (int i = 0; i < 5; i++){int linha = aleatorio.Next(1, 5);int coluna = aleatorio.Next(1, 5);switch (i){case 0:l1 = linha;b1 = coluna;break;case 1:l2 = linha;b2 = coluna;break;case 2:l3 = linha;b3 = coluna;break;case 3:l4 = linha;b4 = coluna;break;case 4:l5 = linha;b5 = coluna;break;}if (i == 3){if (l1 == l2 && b2 == b1){i = -1;}if (l1 == l3 && b1 == b3){i = -1;}if (l1 == l4 && b1 == b4){i = -1;}if (l1 == l5 && b1 == b5){i = -1;}if (l3 == l2 && b2 == b3){i = -1;}if (l3 == l5 && b5 == b3){i = -1;}if (l4 == l3 && b4 == b3){i = -1;}if (l4 == l2 && b2 == b4){i = -1;}if (l4 == l5 && b4 == b5){i = -1;}if (l2 == l5 && b2 == b5){i = -1;}}}string[] a = new string[5] { "a", "b", "c", "d" , "e"};string l11 = "";string l22 = "";string l33 = "";string l44 = "";string l55 = "";int pa = 1;for (int i = 0; i < 5; i++){if (l1 == pa){l11 = a[i];}if (l2 == pa){l22 = a[i];}if (l3 == pa){l33 = a[i];}if (l4 == pa){l44 = a[i];}if (l5 == pa){l55 = a[i];}pa++;}l11 += (b1);l22 += (b2);l33 += (b3);l44 += (b4);l55 += (b5);int coordenada1 = botonesjugador.FindIndex(b => b.Name == l11);int coordenada2 = botonesjugador.FindIndex(b => b.Name == l22);int coordenada3 = botonesjugador.FindIndex(b => b.Name == l33);int coordenada4 = botonesjugador.FindIndex(b => b.Name == l44);int coordenada5 = botonesjugador.FindIndex(b => b.Name == l55);botonesjugador[coordenada1].Tag = "NAVIO";botonesjugador[coordenada2].Tag = "NAVIO";botonesjugador[coordenada3].Tag = "NAVIO";botonesjugador[coordenada4].Tag = "NAVIO";botonesjugador[coordenada5].Tag = "NAVIO";}private void Posiciondelenemigo(object sender, EventArgs e){if (barcosP > 0){var butao = (Button)sender;butao.Enabled = false;indicadorNavioslb.Text = "Navios Restantes: " + (barcosP -1);butao.Tag = "NAVIO";butao.BackgroundImage = Properties.Resources.bombardeiro;barcosP--;}if (barcosP == 0){textoAuxiliar.Top = 100;textoAuxiliar.Left = 70;textoAuxiliar.Text = "Ahora elige la posición de bombardeo enemiga ";if (RODADA == 1){inimigoCasas.Enabled = true;RODADA = 0;}}if (rodadas == 0){if (pontuacionE > pontuacionP){MessageBox.Show("RIVAL HA GANADO, HAS PERDIDO!!" +" :( ");inimigoCasas.Enabled = false;Environment.Exit(1);}else if (pontuacionP > pontuacionE){MessageBox.Show("RIVAL HA PERDIDO, LA VICTORIA ES TUYA!!" +" :) ");inimigoCasas.Enabled = false;Environment.Exit(1);}else{MessageBox.Show("EMPATE");inimigoCasas.Enabled = false;Environment.Exit(1);}}}private void Butao\_BackgroundImageLayoutChanged(object sender, EventArgs e){throw new NotImplementedException();}private void label18\_Click(object sender, EventArgs e){}private void a1\_Click(object sender, EventArgs e){}}} |

#### Una vez agregado toda la funcionalidad como lo son los barcos, el tablero, la posición donde pondrás tu barco, un identificador para saber si acertaste o fallaste, etc. Procedemos a disfrutar el juego

#### -REFERENCIAS

**EJECUCIÓN DEL PROGRAMA EN EL LENGUAJE JAVA**

|  |
| --- |
| **INICIO DEL PROGRAMA** |
| **PRESENTACIÓN DEL TABLERO Y LOS BARCOS** |
| **PRESENTACIÓN DEL JUEGO CORRIENDO** |
| **PRESENTACIÓN DEL JUEGO CONCLUIDO** |

**EJECUCIÓN DEL PROGRAMA EN EL LENGUAJE C#**

|  |
| --- |
| **INICIO DEL PROGRAMA** |
| **PRESENTACIÓN DE CÓMO JUGAR** |
| **PRESENTACIÓN DEL TABLERO Y LOS BARCOS** |
| **PRESENTACIÓN DEL JUEGO CORRIENDO** |
| **PRESENTACIÓN DEL JUEGO CONCLUIDO** |

Oracle © Java Thread. Link: [https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html.](https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html) Febrero 2016.

#### -CONCLUSIÓN

Sinceramente el programa presentó un reto demasiado grande, tuve apoyo de varios videos y artículos que leía en foros de internet, sin embargo la practica y el reto de hacer el programa fue concluido de manera exitosa, ahora solo queda disfrutar este maravilloso juego y seguir con las demás prácticas.

#### -ANEXOS