MODUL 6
PERMAINAN TIC TAC TOE

6.1. Tujuan

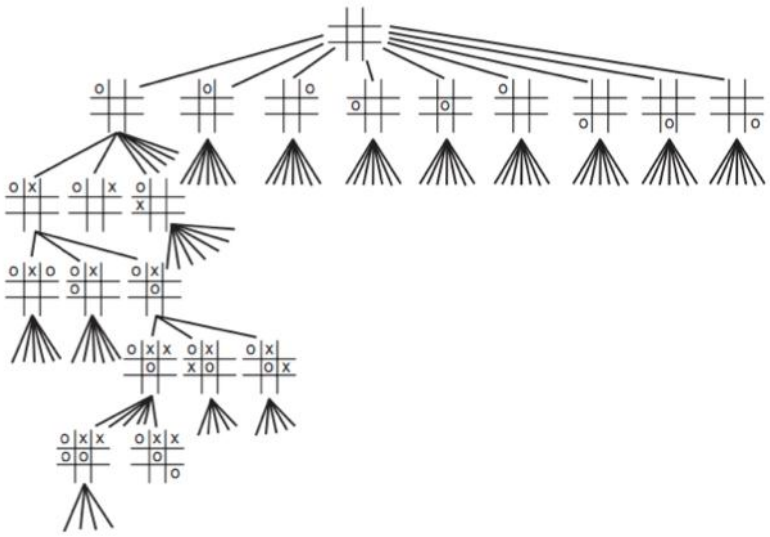
Meningkatkan pemahaman mahasiswa terhadap code permainan tic tac toe. Selain itu, modul 6 memberikan pengetahuan tentang Object Oriented Programming menggunakan bahasa pemrograman Java terutama Java Swing dan Japplet.

6.2. Dasar Teori

6.2.1. Permainan Tic Tac Toe

Penyelesaian masalah permainan tic tac toe dapat menggunakan algoritma heuristic untuk mencapai solusi yang optimal. Pada modul ini memperlihatkan bagaimana membuat sebuah permainan tic tac toe. Initial state dari permainan ini adalah puzzle ukuran 8 yang tidak berisikan apa-apa. Ketika pemain pertama menekan salah satu ubin, maka ubin tersebut akan diberikan tanda silang. Pemain kedua harus menghalangi pemain pertama untuk membuat tanda silang berjajaran baik secara vertikal, horizontal, atau diagonal. Permainan ini akan berakhir (goal state) ketika salah seorang pemain sudah menderetkan tanda meraka masing-masing baik secara vertikal, horizontal, atau diagonal.

Solusi dari permasalahan ini dapat dilakukan dengan membuat topologi Tree, kemudian setiap langkah dari pemain pertama atau kedua akan menjadikan initial state selanjutnya, kemudian langkah tersebut akan dijadikan sebagai initial state yang baru sampai menemukan goal statenya. Ilustrasi penyelesaian masalah permainan tic tac toe ini dapat dilihat melalui Gambar 6.1.



Gambar 6.1. Pohon Permainan Tic Tac Toe

6.3. Implementasi Permainan Tic Tac Toe

6.3.1. Menggunakan Graphical User Interface (GUI)

Permainan ini dibuatkan dengan menggunakan Java Swing yang terdiri dari 3 buah Java Class dan 3 buah pendeklarasian enumaration. Ke-enam file tersebut dituliskan secara terpisah namun disimpan pada folder yang sama (misalnya folder tic tac toe). Ke-enam file tersebut dituliskan nama secara berurutan seperti berikut:

1. State.Java
2. Seed.Java
3. GameState.Java
4. Cell.Java
5. Board.Java
6. GameMain.Java

Pemberian sebuah nilai integer kepada variabel untuk membedakan status penggunaan (misalnya jika tic tac toe sedang dimainkan, variable PLAYING diberikan nilai 0, variable DRAW = 1, dan sebagainya) tidak begitu efektif di dalam penulisan code. Sekarang, JDK1.5 memperkenalkan fitur baru yang dinamakan dengan *enumaration*, yang merupakan class spesial untuk menyimpan semua variabel secara berurutan. Enumaration State, Seed, dan GameState didefinisikan secara file terpisah seperti di bawah ini.

```
package TicTacToe;
/**  *  Enumeration for the various states of the game  */ public enum
GameState {  // to save as "GameState.java"
        PLAYING, DRAW, CROSS_WON, NOUGHT_WON
}
```

```
package TicTacToe;
/**  * Enumeration for the seeds and cell contents  */ public enum Seed {
// to save as "Seed.java"
        EMPTY, CROSS, NOUGHT
}
```

```
package TicTacToe;
/**  *  Enumeration for the various states of the game  */ public enum
State {  // to save as "GameState.java"
        PLAYING, DRAW, CROSS_WON, NOUGHT_WON
}
```

Kemudian diketikkan *code* untuk class Cell, Board, dan GameMain secara terpisah. Program ketiga class tersebut dapat dilihat seperti di bawah ini.

Class Cell.java

Code ini dikutip dari:

http://www3.ntu.edu.sg/home/ehchua/programming/java/JavaGame_TicTacToe.html

```
package TicTacToe;

import java.awt.Graphics;
import java.awt.*;
import java.awt.Graphics2D;

public class Cell {
      //content of this cell (Seed.EMPTY, Seed.CROSS, or Seed.NOUGHT)
      Seed content;
      int row, col; // row and column of this cell

 /**Constructor to initialize this cell with the specified row and col */
   public Cell(int row, int col) {
      this.row = row;
      this.col = col;
      clear(); // clear content
   }

   /** Clear this cell's content to EMPTY */
   public void clear() {
      content = Seed.EMPTY;
   }

   /**Paint itself on the graphics canvas, given the Graphics context */
   public void paint(Graphics g) {
      // Use Graphics2D which allows us to set the pen's stroke
      Graphics2D g2d = (Graphics2D)g;
      g2d.setStroke(new BasicStroke(GameMain.SYMBOL_STROKE_WIDTH,
            BasicStroke.CAP_ROUND,  BasicStroke.JOIN_ROUND));  //  Graphics2D
only
      // Draw the Seed if it is not empty
      int x1 = col * GameMain.CELL_SIZE + GameMain.CELL_PADDING;
      int y1 = row * GameMain.CELL_SIZE + GameMain.CELL_PADDING;
      if (content == Seed.CROSS) {
         g2d.setColor(Color.RED);
         int x2 = (col + 1) * GameMain.CELL_SIZE - GameMain.CELL_PADDING;
         int y2 = (row + 1) * GameMain.CELL_SIZE - GameMain.CELL_PADDING;
         g2d.drawLine(x1, y1, x2, y2);
         g2d.drawLine(x2, y1, x1, y2);
      } else if (content == Seed.NOUGHT) {
         g2d.setColor(Color.BLUE);
         g2d.drawOval(x1, y1, GameMain.SYMBOL_SIZE, GameMain.SYMBOL_SIZE);
      }
   }
}
```

Kemudian tuliskan code program untuk Class Board.java untuk membuat dan mengatur papan permainan tic tac toe.

Class Board.java

```
package TicTacToe;
import java.awt.*;

/**
 * The Board class models the ROWS-by-COLS game-board.
 */
public class Board {
   // package access
   // composes of 2D array of ROWS-by-COLS Cell instances
   Cell[][] cells;

   /** Constructor to initialize the game board */
   public Board() {

      // allocate the array
      cells = new Cell[GameMain.ROWS][GameMain.COLS];
      for (int row = 0; row < GameMain.ROWS; ++row) {
         for (int col = 0; col < GameMain.COLS; ++col) {
            // allocate element of array
            cells[row][col] = new Cell(row, col);
         }

     }
   }

   /** Initialize (or re-initialize) the game board */
   public void init() {
      for (int row = 0; row < GameMain.ROWS; ++row) {
         for (int col = 0; col < GameMain.COLS; ++col) {
            // clear the cell content
            cells[row][col].clear();
         }
      }
   }

   /** Return true if it is a draw (i.e., no more EMPTY cell) */
   public boolean isDraw() {
      for (int row = 0; row < GameMain.ROWS; ++row) {
         for (int col = 0; col < GameMain.COLS; ++col) {
            if (cells[row][col].content == Seed.EMPTY) {
              // an empty seed found, not a draw, exit
               return false;
            }
         }
      }
      return true; // no empty cell, it's a draw
   }

   /** Return true if the player with "seed" has won after placing at
      (seedRow, seedCol) */
   public boolean hasWon(Seed seed, int seedRow, int seedCol) {
      return (cells[seedRow][0].content == seed   // 3-in-the-row
                  && cells[seedRow][1].content == seed
                  && cells[seedRow][2].content == seed
            || cells[0][seedCol].content == seed // 3-in-the-column
                  && cells[1][seedCol].content == seed
```

```
                && cells[2][seedCol].content == seed
            || seedRow == seedCol              // 3-in-the-diagonal
                && cells[0][0].content == seed
                && cells[1][1].content == seed
                && cells[2][2].content == seed
            || seedRow + seedCol == 2          // 3-in-the-opposite-diagonal
                && cells[0][2].content == seed
                && cells[1][1].content == seed
                && cells[2][0].content == seed);
   }

   /** Paint itself on the graphics canvas, given the Graphics context */
   public void paint(Graphics g) {
      // Draw the grid-lines
      g.setColor(Color.GRAY);
      for (int row = 1; row < GameMain.ROWS; ++row) {
         g.fillRoundRect(0,       GameMain.CELL_SIZE      *       row       -
GameMain.GRID_WIDHT_HALF,
               GameMain.CANVAS_WIDTH-1, GameMain.GRID_WIDTH,
               GameMain.GRID_WIDTH, GameMain.GRID_WIDTH);
      }
      for (int col = 1; col < GameMain.COLS; ++col) {
         g.fillRoundRect(GameMain.CELL_SIZE         *         col         -
GameMain.GRID_WIDHT_HALF, 0,
               GameMain.GRID_WIDTH, GameMain.CANVAS_HEIGHT - 1,
               GameMain.GRID_WIDTH, GameMain.GRID_WIDTH);
      }

      // Draw all the cells
      for (int row = 0; row < GameMain.ROWS; ++row) {
         for (int col = 0; col < GameMain.COLS; ++col) {
            cells[row][col].paint(g);  // ask the cell to paint itself
         }
      }
   }
}
```

Kemudian tuliskan code program untuk Class GameMain.java untuk menjalankan permainan tic tac toe.

Class GameMain.java

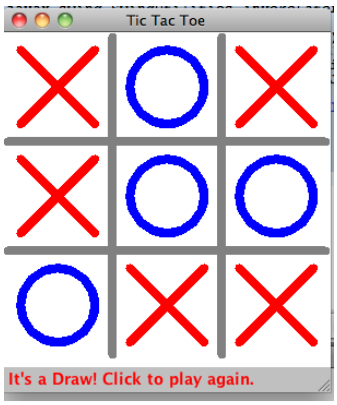
```
package TicTacToe;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/**
 * Tic-Tac-Toe: Two-player Graphic version with better OO design.
 * The Board and Cell classes are separated in their own classes.
 */
@SuppressWarnings("serial")
public class GameMain extends JPanel {
   // Named-constants for the game board
   public static final int ROWS = 3;  // ROWS by COLS cells
   public static final int COLS = 3;
```

```
   public static final String TITLE = "Tic Tac Toe";

   // Name-constants for the various dimensions used for graphics drawing
   public static final int CELL_SIZE = 100; // cell width and height (square)
   public static final int CANVAS_WIDTH = CELL_SIZE * COLS;   // the drawing
canvas
   public static final int CANVAS_HEIGHT = CELL_SIZE * ROWS;
   public static final int GRID_WIDTH = 8;  // Grid-line's width
   public static final int GRID_WIDHT_HALF = GRID_WIDTH / 2; // Grid-line's
half-width
   // Symbols (cross/nought) are displayed inside a cell, with padding from border

   public static final int CELL_PADDING = CELL_SIZE / 6;
   public static final int SYMBOL_SIZE = CELL_SIZE - CELL_PADDING * 2;
   public static final int SYMBOL_STROKE_WIDTH = 8; // pen's stroke width

   private Board board;             // the game board
   private GameState currentState;//the current state of the game
   private Seed currentPlayer;      // the current player
   private JLabel statusBar;     // for displaying status message

   /** Constructor to setup the UI and game components */
   public GameMain() {

      // This JPanel fires MouseEvent
       this.addMouseListener(new MouseAdapter() {
          @Override
          public void mouseClicked(MouseEvent e) {   // mouse-clicked handler
             int mouseX = e.getX();
             int mouseY = e.getY();
            // Get the row and column clicked
             int rowSelected = mouseY / CELL_SIZE;
             int colSelected = mouseX / CELL_SIZE;

             if (currentState == GameState.PLAYING) {
                if (rowSelected >= 0 && rowSelected < ROWS
                      && colSelected >= 0 && colSelected < COLS
                      &&   board.cells[rowSelected][colSelected].content   ==
Seed.EMPTY) {
                   board.cells[rowSelected][colSelected].content           =
currentPlayer; // move
                   updateGame(currentPlayer,  rowSelected,  colSelected);  //
update currentState
                // Switch player
                   currentPlayer   =   (currentPlayer   ==   Seed.CROSS)   ?
Seed.NOUGHT : Seed.CROSS;
                }
             } else {         // game over
                initGame();  // restart the game
             }
             // Refresh the drawing canvas
             repaint();  // Call-back paintComponent().
          }
       });

       // Setup the status bar (JLabel) to display status message
       statusBar = new JLabel("          ");
       statusBar.setFont(new Font(Font.DIALOG_INPUT, Font.BOLD, 14));
       statusBar.setBorder(BorderFactory.createEmptyBorder(2, 5, 4, 5));
       statusBar.setOpaque(true);
       statusBar.setBackground(Color.LIGHT_GRAY);

       setLayout(new BorderLayout());
```

```
      add(statusBar, BorderLayout.PAGE_END); // same as SOUTH
      setPreferredSize(new Dimension(CANVAS_WIDTH, CANVAS_HEIGHT + 30));
            // account for statusBar in height

      board = new Board();    // allocate the game-board
      initGame();  // Initialize the game variables
   }

   /** Initialize the game-board contents and the current-state */
   public void initGame() {
      for (int row = 0; row < ROWS; ++row) {
         for (int col = 0; col < COLS; ++col) {
            board.cells[row][col].content = Seed.EMPTY; // all cells empty
         }
      }
      currentState = GameState.PLAYING;  // ready to play
      currentPlayer = Seed.CROSS;         // cross plays first
   }

   /** Update the currentState after the player with "theSeed" has placed on (row, col) */
   public void updateGame(Seed theSeed, int row, int col) {
      if (board.hasWon(theSeed, row, col)) {  // check for win
         currentState = (theSeed == Seed.CROSS) ? GameState.CROSS_WON :
GameState.NOUGHT_WON;
      } else if (board.isDraw()) {  // check for draw
         currentState = GameState.DRAW;
      }
     // Otherwise, no change to current state (PLAYING).
   }

  /** Custom painting codes on this JPanel */
   @Override
   public void paintComponent(Graphics g){
      //invoke via repaint()
      super.paintComponent(g);     // fill background
      setBackground(Color.WHITE); // set its background color

      board.paint(g);  // ask the game board to paint itself

     // Print status-bar message
      if (currentState == GameState.PLAYING) {
         statusBar.setForeground(Color.BLACK);
         if (currentPlayer == Seed.CROSS) {
            statusBar.setText("X's Turn");
         } else {
            statusBar.setText("O's Turn");
         }
      } else if (currentState == GameState.DRAW) {
         statusBar.setForeground(Color.RED);
         statusBar.setText("It's a Draw! Click to play again.");
      } else if (currentState == GameState.CROSS_WON) {
         statusBar.setForeground(Color.RED);
         statusBar.setText("'X' Won! Click to play again.");
      } else if (currentState == GameState.NOUGHT_WON) {
         statusBar.setForeground(Color.RED);
         statusBar.setText("'O' Won! Click to play again.");
      }
   }

  /** The entry "main" method */
   public static void main(String[] args) {
      // Run GUI construction codes in Event-Dispatching thread for thread safety
      javax.swing.SwingUtilities.invokeLater(new Runnable() {
```

```
            public void run() {
                JFrame frame = new JFrame(TITLE);
        // Set the content-pane of the JFrame to an instance of main JPanel
                frame.setContentPane(new GameMain());
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.pack();
                // center the application window
                frame.setLocationRelativeTo(null);
                frame.setVisible(true);    // show it
            }
        });
    }
}
```

Untuk menjalankan program ini, perlu di-compile dan dijalankan file Java yang berisikan ethod void main. Pada program ini Java Class GameMain.java yang akan di-compile dan dijalankan. Output dari program ini menghasilkan GUI tic tac toe seperti Gambar 6.2.



Gambar 6.2. GUI Permainan Tic Tac Toe

6.3.2. Menggunakan Java Applet (JApplet)

Program permainan tic tac toe ini juga dapat dimainkan melalui web browser. Buatlah sebuah file Java Applet dengan nama AppletMain.java dengan memasukkan class `javax.swing.JApplet`. Program file AppletMain.java diimplementasikan seperti berikut.

```
import javax.swing.*;

/** Tic-tac-toe Applet */
@SuppressWarnings("serial")
public class AppletMain extends JApplet {
      /** init() to setup the GUI components */
      @Override
      public void init() {
            // Run GUI codes in the Event-Dispatching thread for thread safety
            try {
                // Use invokeAndWait() to ensure that init() exits after GUI construction
            SwingUtilities.invokeAndWait(new Runnable() {
                @Override
                public void run() {
                    setContentPane(new GameMain());
```

```
                }
            });
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
}
```

Terakhir, tuliskan sebuah file HTML dengan nama (misalnya TicTacToe.html) yang menempelkan Class AppletMain. Berikut code program TicTacToe.html.

```
<html>
   <head>
        <title>Tic Tac Toe</title>
   </head>
<body>
   <h1>Tic Tac Toe</h1>
   <applet code="AppletMain.class" width="300" height="330" alt="Error
Loading Applet?!">   Your browser does not seem to support &lt;APPLET&gt;
tag!
   </applet>
</body>
</html>
```

Tugas:

Tuliskan semua Java Code diatas, kemudian pelajari code nya, dan ubahkan beberapa bagian untuk melihat perubahannya.