Laporan Hasil Proyek Ujian Akhir Semester

Object Orientated Program

JENJANG STUDI : Strata Satu (S1)

PROGRAM STUDI : Teknologi Informasi

DISUSUN OLEH :

Andi Alif Lakipadada N. (42030077)

PROGRAM STUDI TEKNOLOGI INFORMASI FAKULTAS TEKNIK DAN INFORMATIKA UNIVERSITAS PENDIDIKAN NASIONAL

Laporan hasil pembuatan project aplikasi berbasis OOP menggunakan Bahasa pemrograman java dengan system berbasis CRUD (Create,Read,Update,Delete)

Sebelum masuk ke penjelasan codingnya, berikut adalah database yang akan saya gunakan untuk disambungkan ke program yang akan saya buat.

Dbgamingjoki.sql

Nama	EmailAkun	Pass	Nick
Muhammad Azzah	monkeysupreme@gmailcom	azzah123	Caa...
Kayla Annisa	cheeseburger@gmail.com	liug3ndut	liuwliuw
Winda Retno	windarfh@gmail.com	moodswing123	moodswingz
Andi Axl	axlmpd@gmail.com	axlacce26	NataDeCoco
Aprian Sanger	aprians@gmail.com	bananaredv3lvet	ternity
(NULL)	*(NULL)*	*(NULL)*	*(NULL)*

Database.java

Pada file database.java ini, kita hanya mendeklarasikan beberapa kelas yang nantinya akan digunakan untuk penampilan database yang kita ambil dari database yang telah dibuat pada sqlyog

```
import javafx.beans.property.SimpleStringProperty
```

Sintak import diatas digunakan untuk memanggil ataupun memasukkan method/perintah dari library, yang dimana pada kasus ini kita menggunakannya untuk memasukkan package method dari java beans.

```
public Database(String fNama, String fEmailAkun, String fPass, String fNick){
    this.Nama = new SimpleStringProperty(fNama);
    this.EmailAkun = new SimpleStringProperty(fEmailAkun);
    this.Pass = new SimpleStringProperty(fPass);
    this.Nick = new SimpleStringProperty(fNick);
}

public String getNama() {
    return Nama.get();
}

public void setNama(String Value) {
    Nama.set(Value);
}

public String getEmailAkun() {
    return EmailAkun.get();
}

public void setEmailAkun(String Value) {
    EmailAkun.set(Value);
}

public String getPass() {
    return Pass.get();
}
    public final SimpleStringProperty Nick;
```

Sintak diatas adalah pembuatan property/field database yang dimana dibuat dengan tipe “public” untuk menandakan bahwa method dapat diakses dari kelas lain, yang dimana kelas bawahnya berupa “public final SimpleStringProperty ****;”. Public final ini menandakan bahwa kelas tersebut tidak ada pewarisan atau sudah merupakan tahap akhir yang berarti saya tidak lagi akan mengubahnya. Sementara SimpleStringProperty merupakan bawaan dari javabeans yang sedang kita gunakan, untuk membuat kelas.
Kemudian sisa dari sintak yang ada pada file ini hanya digunakan untuk pembuatan kelas serta untuk mengembalikan value/nilai yang tersimpan dari kelas tersebut. Pada kelas yang saya buat diatas tersebut terdapat setter dan getter dari masing-masing property yang telah saya buat diatas.

DatabaseConnector.java
Kemudian pada file ini, sesuai dengan namanya, kita menggunakan file ini untuk menyambungkan database dengan program yang akan kita buat.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

Sama seperti sebelumnya, fungsi import ini digunakan untuk memasukkan sebuah method dari library, yang dimana kali ini kita memasukkan method untuk menghubungkan java ke database, dan juga sebuah driver yang dibutuhkan agar java dan database mampu berkomunikasi.

```
public class DatabaseConnector {
    private static Connection connect;
    public static Connection tryConnect()
    {
            if(connect == null)
        {
            try {
                String url  ="jdbc:mysql://localhost:3306/dbgamingjoki";
                String user = "root";
                String pass = "";

                DriverManager.registerDriver(new com.mysql.jdbc.Driver());
                connect = DriverManager.getConnection(url, user, pass);
            } catch (SQLException ex) {
            //Logger.getLogger(DatabaseConnector.class.getName()).log(Level.SEVERE, null, ex);
                System.out.println("Koneksi Gagal");
            }
        }
         return connect;
```

Kemudian pada sintak selanjutnya, kita membuat sebuah kelas yang berisi kelas lain untuk menghubungkan database dengan program java yang kita buat. Disini saya mendeklarasikan kelas connect dan tryConnect, yang dimana tryConnect ini digunakan untuk mengecek dan memastikan bahwa koneksi hanya berlangsung sekali Ketika aplikasi dilaunch. Kemudian untuk kelas connect nantinya akan direturnkan pada akhir sintak. Diatas dapat kita lihat ada beberapa variable dengan tipe data string

yang berupa url,user,dan pass yang merupakan server xampp yang digunakan untuk mengakses database yang telah saya buat.

App.java

App.java > App > initComponent()

```
import static javafx.scene.control.TableView.CONSTRAINED_RESIZE_POLICY;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.geometry.Insets;
import javafx.geometry.Orientation;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.SplitPane;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
```

Sama seperti sebelumnya, saya mengimport banyak library yang digunakan untuk memasukkan dan mengaktifkan sebuah method.

```
public class App extends Application {
    TableView<Database> tableView = new TableView<Database>();
    public TableView tblView;
    private Text txtInfo;
    private Label lblTitle,lblData,lblNama,lblEmailAkun,lblPass,lblNick,lblSearch;
    public TextField txtNama,txtEmailAkun,txtNick,txtPass,txtSearch;
    public TableColumn tblColumn1,tblColumn2,tblColumn3,tblColumn4;
    private SplitPane splitPaneH;
    private VBox panevbox,panevbox2;
    private AnchorPane pane;
    private GridPane grid;
    private HBox panehbox,searchbox;
    private Button btnAdd,btnUpdate,btnDelete,btnClear,btnClose,btnRefresh;
    Database modelDb;
    ObservableList data = FXCollections.observableArrayList();
```

Sintak diatas berfungsi untuk menciptakan object dari tableview yang akan kita gunakan untuk menampilkan program databasenya. Diatas dapat terlihat begitu banyak kelas yang dimana kelas tersebut digunakan untuk mengatur mulai dari jenis font,ukuran font,ukuran table,label,kolom table,pembatas antar table dan lain lain yang dimana kelas-kelas tersebut saya panggil ulang dalam sintak “import” untuk

mengaktifkannya saat aplikasi dijalankan. Dan dari keseluruhan kelas saya kemudian membuat variable modelDb yang digunakan untuk mempermudah pemanggilan kelas.

```
public void initComponent(){
    //========================================================================
      lblData    = new Label("FORM DATA");
      lblTitle   = new Label();
      lblNama     = new Label("Nama");
      lblEmailAkun    = new Label("Email Akun");
      lblPass = new Label("Password Akun");
      lblNick  = new Label("Nickname ingame");
      lblSearch    = new Label("Cari data :");
      txtInfo    = new Text("data not available");
      tblColumn1 = new TableColumn("Nama");
      tblColumn2 = new TableColumn("EmailAkun");
      tblColumn3 = new TableColumn("Pass");
      tblColumn4 = new TableColumn("Nick");
      txtNama     = new TextField();
      txtEmailAkun    = new TextField();
      txtPass = new TextField();
      txtNick  = new TextField();
      txtSearch    = new TextField();
      splitPaneH = new SplitPane();
      pane       = new AnchorPane();
      panevbox   = new VBox();
      panevbox2  = new VBox();
      grid       = new GridPane();
      panehbox   = new HBox(5);
      searchbox  = new HBox(5);
      tblView    = new TableView();
      btnAdd     = new Button("ADD");
      btnUpdate  = new Button("UPDATE");
      btnDelete  = new Button("DELETE");
      btnClear   = new Button("CLEAR");
      btnClose   = new Button("CLOSE");
      btnRefresh = new Button("REFRESH");
```

Kemudian saya membuat beberapa instance pada setiap kelas yang telah saya buat sebelumnya seperti pada gambar diatas. Saya membuat beberapa instance untuk masing-masing property dan teks. Dan pada bagian akhir saya membuat instance untuk tombol INSERT/ADD, UPDATE, DELETE, CLEAR, CLOSE, dan REFRESH.

```
tblColumn1.setCellValueFactory(new PropertyValueFactory("Nama"));
tblColumn2.setCellValueFactory(new PropertyValueFactory("EmailAkun"));
tblColumn3.setCellValueFactory(new PropertyValueFactory("Pass"));
tblColumn4.setCellValueFactory(new PropertyValueFactory("Nick"));

txtNama.setPromptText("Masukkan Nama");
txtEmailAkun.setPromptText("Masukkan Email Akun");
txtPass.setPromptText("Masukkan Password");
txtNick.setPromptText("Masukkan Nickname");
txtSearch.setPromptText("Masukkan data yang ingin dicari");

lblSearch.setPadding(new Insets(10));
lblSearch.setFont(Font.font("Franklin Gothic Demi", FontWeight.MEDIUM, 12));
lblSearch.setAlignment(Pos.CENTER);
lblSearch.setUnderline(true);

lblData.setPadding(new Insets(10));
lblData.setFont(Font.font("Franklin Gothic Demi", FontWeight.MEDIUM, 22));
lblData.setUnderline(true);
lblData.setAlignment(Pos.CENTER);

lblTitle.setText("Database GamingJoki");
lblTitle.setUnderline(true);
lblTitle.setPadding(new Insets(10));
lblTitle.setFont(Font.font("Franklin Gothic Demi", FontWeight.MEDIUM, 22));
lblTitle.setAlignment(Pos.CENTER);

lblNama.setPrefSize(100, 30);
lblEmailAkun.setPrefSize(100, 30);
lblPass.setPrefSize(100, 30);
lblNick.setPrefSize(100, 30);
```

```
txtNama.setPrefSize(250, 30);
txtEmailAkun.setPrefSize(250, 30);
txtPass.setPrefSize(250, 30);
txtNick.setPrefSize(250, 30);
txtSearch.setPrefSize(250, 30);

tblView.setColumnResizePolicy(CONSTRAINED_RESIZE_POLICY);
tblView.setPlaceholder(txtInfo);
tblView.setPadding(new Insets(10));
tblView.getColumns().addAll(tblColumn1,tblColumn2,tblColumn3,tblColumn4);
tblView.setPrefHeight(250);
tblView.setBackground(new Background(
        new BackgroundFill(Color.LIGHTBLUE,new CornerRadii(15),Insets.EMPTY)));

panehbox.setAlignment(Pos.CENTER);
panehbox.setPadding(new Insets(10));
panehbox.setLayoutX(23);
panehbox.setLayoutY(194);
panehbox.getChildren().addAll(btnAdd,btnUpdate,btnDelete,btnClear,btnClose);
panehbox.setBackground(new Background(new BackgroundFill(
        Color.DARKGRAY, new CornerRadii(10), Insets.EMPTY)));

searchbox.setAlignment(Pos.CENTER_LEFT);
searchbox.setPadding(new Insets(5));
searchbox.getChildren().addAll(lblSearch,txtSearch,btnRefresh);
searchbox.setBackground(new Background(new BackgroundFill(
        Color.DARKGRAY, new CornerRadii(10), Insets.EMPTY)));

grid.setHgap(10);
grid.setVgap(10);
```

```
grid.setLayoutY(5);
grid.setAlignment(Pos.CENTER);
grid.setPadding(new Insets(10));
grid.addRow(1, lblNama,txtNama);
grid.addRow(2, lblEmailAkun,txtEmailAkun);
grid.addRow(3, lblPass,txtPass);
grid.addRow(4, lblNick,txtNick);
grid.setGridLinesVisible(false);

pane.setBorder(new Border(new BorderStroke(
        Color.WHITESMOKE,BorderStrokeStyle.DASHED,
        new CornerRadii(15),new BorderWidths(5),Insets.EMPTY)));
pane.setBackground(new Background(new BackgroundFill(
        Color.LIGHTGRAY, new CornerRadii(15),Insets.EMPTY)));
pane.getChildren().addAll(grid,panehbox);

panevbox.getChildren().addAll(lblTitle,tblView,searchbox);
panevbox.setPadding(new  Insets(5));
panevbox.setSpacing(5);
panevbox.minWidthProperty().bind(splitPaneH.widthProperty().multiply(0.70));
panevbox.maxWidthProperty().bind(splitPaneH.widthProperty().multiply(0.70));

panevbox2.getChildren().addAll(lblData,pane);
panevbox2.setPadding(new Insets(5));
panevbox2.setSpacing(5);

splitPaneH.setOrientation(Orientation.HORIZONTAL);
splitPaneH.getItems().addAll(panevbox,panevbox2);
splitPaneH.setPadding(new Insets(2));
splitPaneH.setBackground(new Background(
            new BackgroundFill(Color.GRAY,CornerRadii.EMPTY,Insets.EMPTY)));
```

Setelah membuat tampilan table yang ada pada gambar-gambar diatas, saya kemudian membuat penampungan dari data yang diambil pada database yang telah saya buat untuk menampilkannya pada program GUI Javafx yang telah saya buat seperti sebagai berikut :

```
private ObservableList loadData(){
        ObservableList listData = FXCollections.observableArrayList();
        try {
        Connection c     = DatabaseConnector.tryConnect();
        String sql1      = "select * from dbgamingjoki;";
        ResultSet rs1    = c.createStatement().executeQuery(sql1);
        while(rs1.next()){
            modelDb          = new Database(rs1.getString(1),rs1.getString(2),
                                            rs1.getString(3),rs1.getString(4));
            listData.add(modelDb);
        }
    } catch (SQLException ex) {
        Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);
    }
        return listData;

}
```

Disini saya membuat fungsi loadData yang bertujuan untuk menampilkan data dari database.

- Membuat variable listData untuk menampung perintah dari observableArrayList.
- Memanggil tryConnect Kembali untuk menghubungkan database.
- Query mysql
- Perulangan untuk pemanggilan nilai dari setiap data pada table jokigaming
- Dan pengembalian value/nilai dari listData.

Kemudian dengan cara yang sama dan dengan penyesuaian query mysql, saya membuat fitur search data, add/insert data,delete data, dan update data dengan menggunakan nama (pelanggan) yang merupakan primary key.

```
private ObservableList searchByNama(String n){
    ObservableList listData = FXCollections.observableArrayList();
    try {
        Connection c = DatabaseConnector.tryConnect();
        String sql2 = " select distinct * from dbgamingjoki where nama like '%"+n+"%';";
        ResultSet rs2 = c.createStatement().executeQuery(sql2);
        while(rs2.next()){
            modelDb         = new Database(rs2.getString(1),rs2.getString(2),
                                           rs2.getString(3),rs2.getString(4));
            listData.add(modelDb);
        }
    } catch (SQLException ex) {
        Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);
    }
    return listData;
}
```

```
private void insert(Database m){
    Connection c = DatabaseConnector.tryConnect();
    PreparedStatement ps;
    try {
        String sql = "insert into dbgamingjoki values (?,?,?,?);";
        ps  = c.prepareStatement(sql);
        ps.setString(1,m.getNama());
        ps.setString(2,m.getEmailAkun());
        ps.setString(3,m.getPass());
        ps.setString(4,m.getNick());
        ps.execute();
    } catch (SQLException ex) {
        Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);
        System.out.println("Error");
    }
}
```

```
private void delete(Database m){
    try {
        Connection c = DatabaseConnector.tryConnect();
        PreparedStatement ps;
        String sql = "delete from dbgamingjoki where nama = ?;";
        ps = c.prepareStatement(sql);
        ps.setString(1, m.getNama());
        ps.execute();
    } catch (SQLException ex) {
        Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```

```
private void update(Database m){
    try {
        Connection c = DatabaseConnector.tryConnect();
        PreparedStatement ps;
        String sql = "update dbgamingjoki set nickname = ? ,password = ? , email akun = ? where nama = ? ;";
        ps = c.prepareStatement(sql);
        ps.setString(1, m.getNick());
        ps.setString(2, m.getPass());
        ps.setString(3, m.getEmailAkun());
        ps.setString(4, m.getNama());
        ps.execute();
    } catch (SQLException ex) {
        Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Kemudian selanjutnya saya membuat action event javafx, yang dimana pada sebelumnya saya telah membuat program berbasis CRUD (Create, Read, Update, Delete) disini saya akan menambahkan beberapa fitur yang mampu mempermudah aplikasi saya dalam penggunaannya, fitur tersebut ialah :

- selectData(): berfungsi untuk menyeleksi data dengan cara klik cursor pada setiap field yang tersedia
- deleteData(): berfungsi untuk menghapus data berdasarkan seleksi kursor yang saya klik. Jika biasanya kita perlu menginput nk atau primary keynya, dengan ini saya tidak perlu lagi melakukan itu
- updateData(): berfungsi untuk mengubah data berdasarkan seleksi kursor yang di klik. Jika biasanya kita perlu menginput nk atau primary keynya, dengan ini saya tidak perlu lagi melakukan itu
- searchbynk(): berfungsi untuk menghapus pada penampung data, menaruh data pada table agar bisa tampil, dan menghapus seleksi baris pada table
- refresh(): menampilkan data semula, menghapus setiap field yang sebelumnya terisi telah terisi, agar terlihat seperti semula saat GUI ditampilkan
- showData(): menampilkan data yang ada pada database
- clearData(): menghapus semua textfield yang terisi, yang ada pada scene
- addData(): mengambil data dari form, kemudian disusun seperti array

```
/**=========================================================================================
 *                                    ACTIONEVENT
 * ========================================================================================
 **/
private void selectData(){
    modelDb = (Database) tblView.getSelectionModel().getSelectedItems().get(0);
    txtNama.setText(modelDb.getNama());
    txtEmailAkun.setText(modelDb.getNick());
    txtPass.setText(modelDb.getPass());
    txtNick.setText(modelDb.getNick());
    txtNama.setDisable(true);
}

private void deleteData(){
    modelDb = new Database(txtNama.getText(), "", "", "");
    delete(modelDb);
    clearData();
    showData();
}

private void updateData(){
    modelDb = new Database(txtNama.getText(),txtEmailAkun.getText(),
                           txtPass.getText(),txtNick.getText());
    update(modelDb);
    clearData();
    showData();
}

private void searchbyNama(){
    data.clear(); // <- menghapus data pada penampung data
    data = searchByNama(txtSearch.getText().trim());
    tblView.setItems(data); // <- menaruh data pada tabel agar bisa tampil
    tblView.getSelectionModel().clearSelection(); // <- menghapus seleksi baris pada tabel
}

private void refresh(){
    showData();
    clearData();
    txtSearch.clear();
```

```
    private void showData(){
        data.clear();
        data = loadData();
        tblView.setItems(data);
        tblView.getSelectionModel().clearSelection();
    }

    private void clearData(){
        txtNama.clear();
        txtEmailAkun.clear();
        txtPass.clear();
        txtNick.clear();
        txtNama.setDisable(false);
        tblView.getSelectionModel().clearSelection();
    }

    private void addData(){
        // mengambil data dari form, kemudian disusun seperti array
        modelDb = new Database(txtNama.getText(),txtEmailAkun.getText(),
                               txtPass.getText(),txtNama.getText());
        insert(modelDb); //<- data dikirim ke SQL
        showData();
        clearData();
    }
```

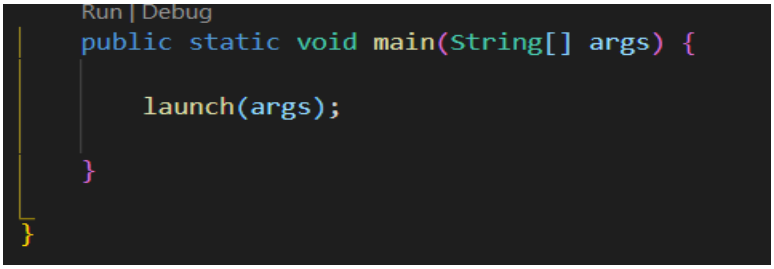
Kemudian pada program selanjutnya saya menambahkan fitur override, overriding adalah sebuah fitur yang memungkinkan sebuah subkelas atau kelas anak yang menyediakan sebuah implementasi yang spesifik dari metode yang sudah disediakan oleh salah satu dari super kelas atau parent class.

Program ini menjelaskan bahwa sebelumnya saya telah membuat fitur button, function, dan program pendukung lainnya. Tentunya saya ingin tombol dan program yang saya buat dapat berfungsi dengan baik saat saya klik tombol tersebut, maka dari itu saya membutuhkan fitur ActionEvent. Fitur ini memungkinkan saya melakukan aksi saat saya klik setiap button yang tersedia pada scene GUI. Jadi saat saya klik setiap button yang ada, maka program akan memanggil function / method yang berkaitan

```
//================================================================================
    @Override
    public void start(Stage primaryStage) {
        initComponent(); // <- VIEW
        showData();     // <- MENAMPILKAN DATA
        tblView.setOnMousePressed((MouseEvent event) -> {
            selectData(); // <- EVENT BARIS KETIKA DIPILIH
        });
        btnAdd.setOnAction((ActionEvent e) -> {
            addData(); // <- INSERT DATA
        });
        btnClear.setOnAction((ActionEvent e) -> {
            clearData(); // <- CLEAR FIELD INPUT DATA
        });
        btnClose.setOnAction((ActionEvent e) -> {
            primaryStage.close(); // <- CLOSE SCENE WINDOW
        });
        btnUpdate.setOnAction((ActionEvent e) -> {
            updateData(); // <- UPDATE DATA
        });
        btnDelete.setOnAction((ActionEvent e) -> {
            deleteData(); // <- DELETE DATA
        });
        btnRefresh.setOnAction((ActionEvent e) -> {
            refresh(); // <- MENGEMBALIKAN TAMPILAN SEPERTI SEMULA
        });
        txtSearch.setOnKeyTyped((KeyEvent ke) -> {
            searchbyNama(); // <- SEARCH DATA BY Nama
        });
        Scene scene = new Scene(splitPaneH, 1216, 618);
        scene.setFill(null);
        primaryStage.setScene(scene);

        primaryStage.setTitle("CRUD APPLICATION WITH JAVAFX");
        // primaryStage.setScene(scene);
        primaryStage.show();
        primaryStage.setFullScreen(true);
    }
```

Dan pada bagian paling akhir adalah perintah untuk debug/run, dimana fungsi ini adalah fungsi main yang paling pertama akan dipanggil Ketika aplikasi di *Run* atau dijalankan.


Run | Debug
public static void main(String[] args) {
launch(args);
}
}