

# **UJIAN AKHIR SEMESTER**

## **PEMROGRAMAN BERORIENTASI OBJEK CRUD APPLICATION WITH JAVA FX**

DOSEN PENGAMPU : Ir. Gede Humaswara Prathama, S.T., M.T.



OLEH :

Nama : Meldodi Samuel Sinturi  
NIM : (42030045)

**JURUSAN TEKNOLOGI INFORMASI  
FAKULTAS TEKNIK DAN INFORMATIKA  
UNIVERSITAS PENDIDIKAN NASIONAL (UNDIKNAS)  
2021**

## CRUD APPLICATION WITH JAVAFX

Pada kali ini saya telah membuat sebuah project aplikasi berbasis objek menggunakan bahasa pemrograman java yang dapat membuat, membaca, mengubah, dan menghapus data atau CRUD dengan bantuan javafx.

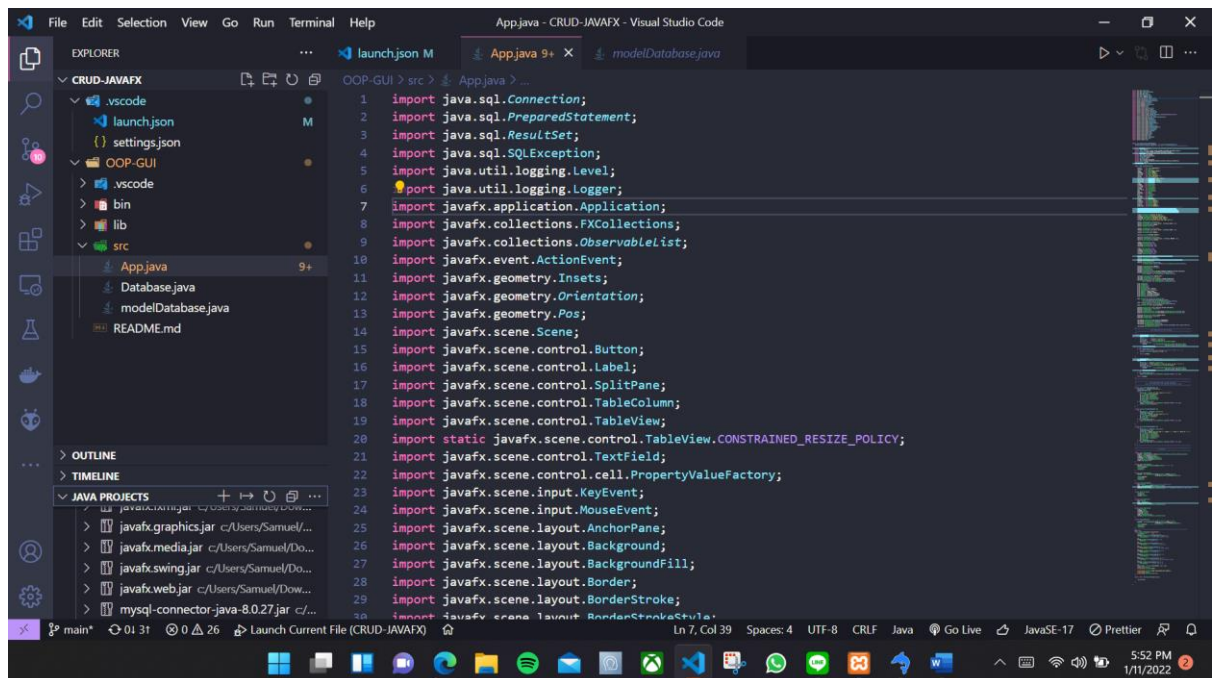
Javafx adalah sebuah platform software untuk membangun sebuah aplikasi rich internet application (RIA) yang bisa berjalan pada berbagai macam perangkat. Seperti komputer dekstop, web browser di Windows, Linux dan Mac OSX.

Sebelumnya saya telah membuat database di aplikasi sqlyog dengan nama database kossan, dan satu table dengan nama table kost. Dengan nama-nama field dan data sebagai berikut:

<input type="checkbox"/>	nk	nama	telpon	status
<input type="checkbox"/>	A01	Dede	090897	Mahasiswa
<input type="checkbox"/>	A02	Wilno	090879	Mahasiswa
<input type="checkbox"/>	B12	Heru	090765	Pekerja
*	(NULL)	(NULL)	(NULL)	(NULL)

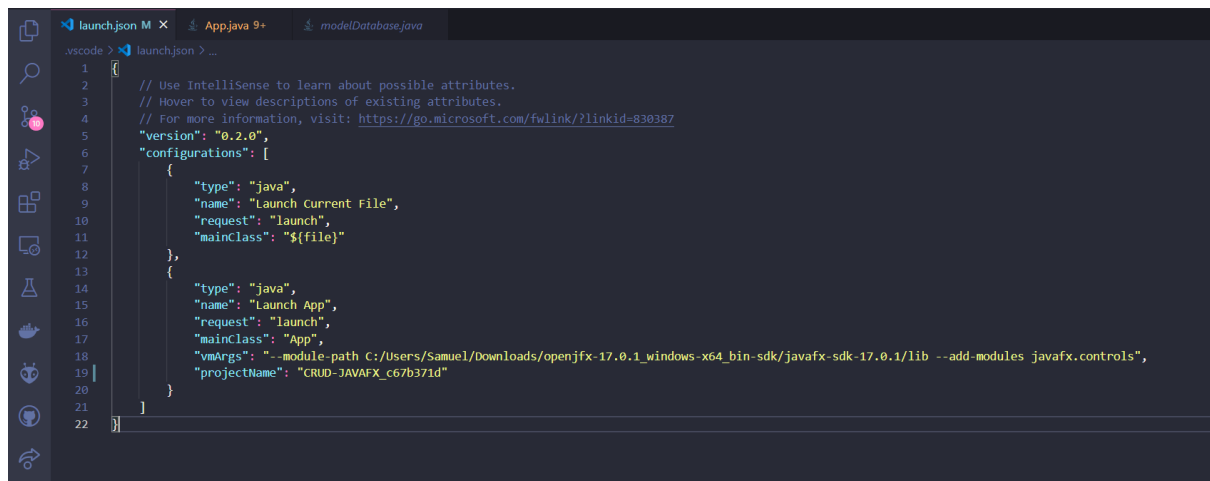
Setelah itu saya mendownload library javafx di [Getting Started with JavaFX \(openjfx.io\)](https://openjfx.io) Dan mendownload driver mysql connector di [MySQL :: Download Connector/J](https://dev.mysql.com/downloads/connector/java/)

### 1. Membuat project java menggunakan IDE Visual Studio Code



Disini saya telah membuat project dengan tiga class yaitu App.java, Database.java, dan modelDatabase.java

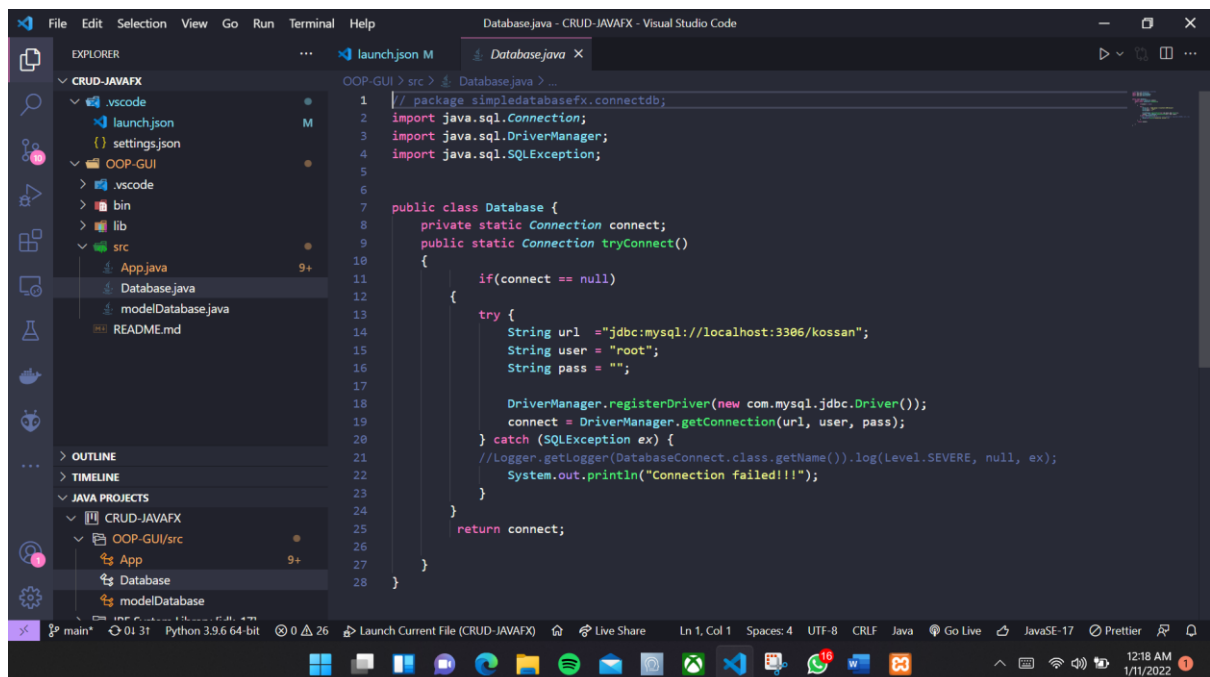
Pada project ini saya sudah menambahkan library javafx untuk menampilkan Graphic User Interface (GUI) dan mysql connector untuk menghubungkan java dengan database yang saya buat.



```
1 // Use IntelliSense to learn about possible attributes.
2 // Hover to view descriptions of existing attributes.
3 // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
4 "version": "0.2.0",
5 "configurations": [
6   {
7     "type": "java",
8     "name": "Launch Current File",
9     "request": "launch",
10    "mainClass": "${file}"
11  },
12  {
13    "type": "java",
14    "name": "Launch App",
15    "request": "launch",
16    "mainClass": "App",
17    "vmArgs": "-module-path C:/Users/Samuel/Downloads/openjfx-17.0.1_windows-x64_bin-sdk/javafx-sdk-17.0.1/lib --add-modules javafx.controls",
18    "projectName": "CRUD-JAVAFX_c67b371d"
19  }
20 ]
```

Lalu disini saya menambahkan konfigurasi untuk menempatkan directory file library javafx yang sudah saya download.

## 2. Koneksi Database



```
1 // package simpledatabasefx.connectdb;
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.SQLException;
5
6
7 public class Database {
8     private static Connection connect;
9     public static Connection tryConnect()
10    {
11        if(connect == null)
12        {
13            try {
14                String url = "jdbc:mysql://localhost:3306/kossan";
15                String user = "root";
16                String pass = "";
17
18                DriverManager.registerDriver(new com.mysql.jdbc.Driver());
19                connect = DriverManager.getConnection(url, user, pass);
20            } catch (SQLException ex) {
21                //Logger.getLogger(DatabaseConnect.class.getName()).log(Level.SEVERE, null, ex);
22                System.out.println("Connection failed!!!");
23            }
24        }
25        return connect;
26    }
27 }
28 }
```

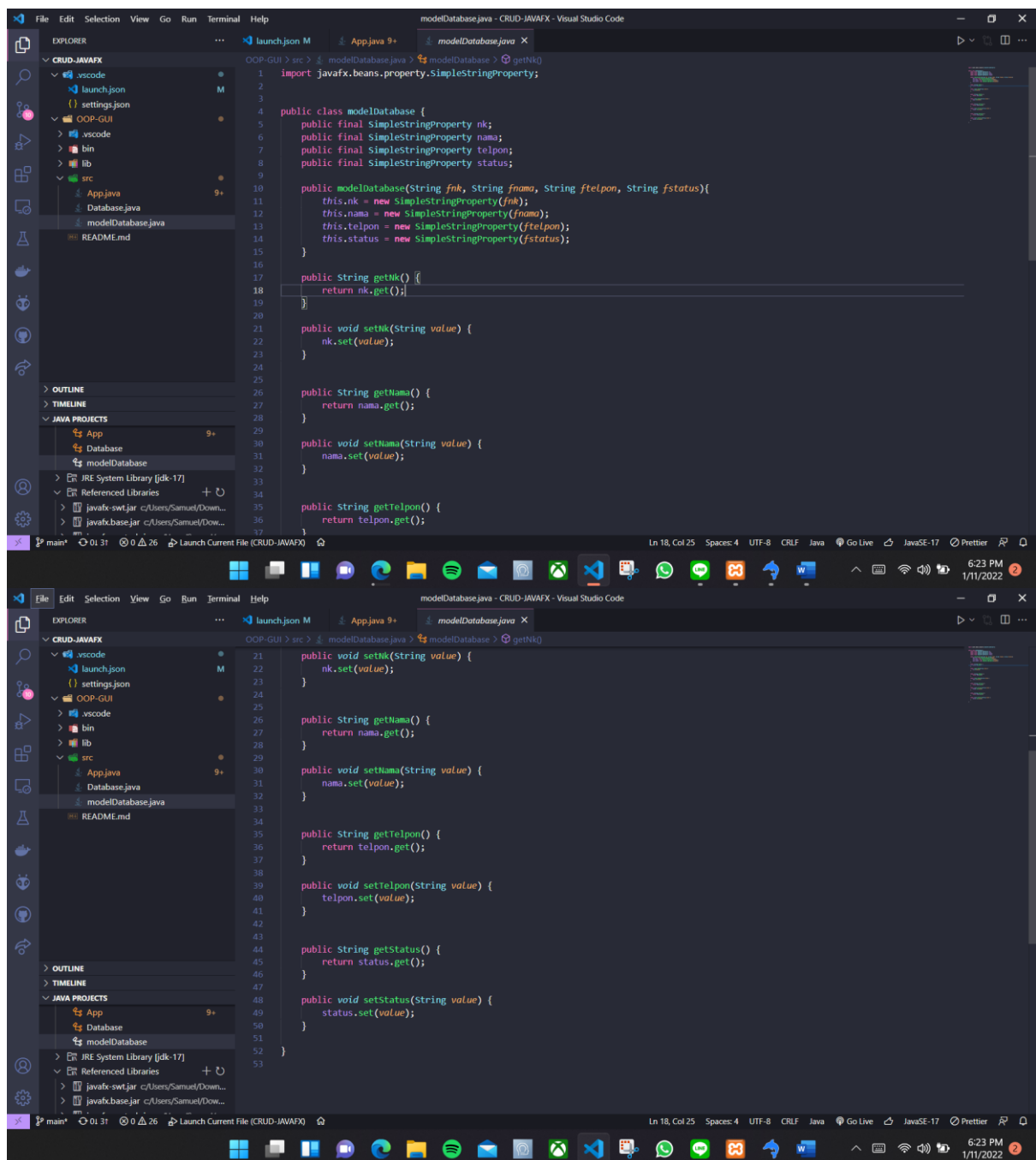
Saya memasukan/ import beberapa library yang dibutuhkan untuk membuat program pada gambar diatas

Disini saya membuat sebuah class yang berfungsi untuk membangun koneksi ke MySQL, saya deklarasikan interface Connection dengan akses modifier private dan static dan dibuat metode tryConnect() untuk mendapatkan koneksi. Di dalam method tersebut terdapat

pengecekan untuk memastikan bahwa koneksi hanya dilakukan sekali ketika aplikasi berjalan, ini biasa disebut sebagai singleton.

Selanjutnya saya mendeklarasikan beberapa variable dengan type data String, seperti gambar diatas yang dibuatkan variable adalah url, user, dan pass. Dan diakhir program saya kembalikan atau return value / nilai pada variable connect

### 3. Membuat sebuah model database



Disini saya akan membuat sebuah tabel yang dimanfaatkan untuk menampung informasi database kossan saya, sehingga saya representasikan ke dalam sebuah class yaitu `modelDatabase`. Jika di Java sering disebut dengan entitas atau juga POJO, dimana di dalam

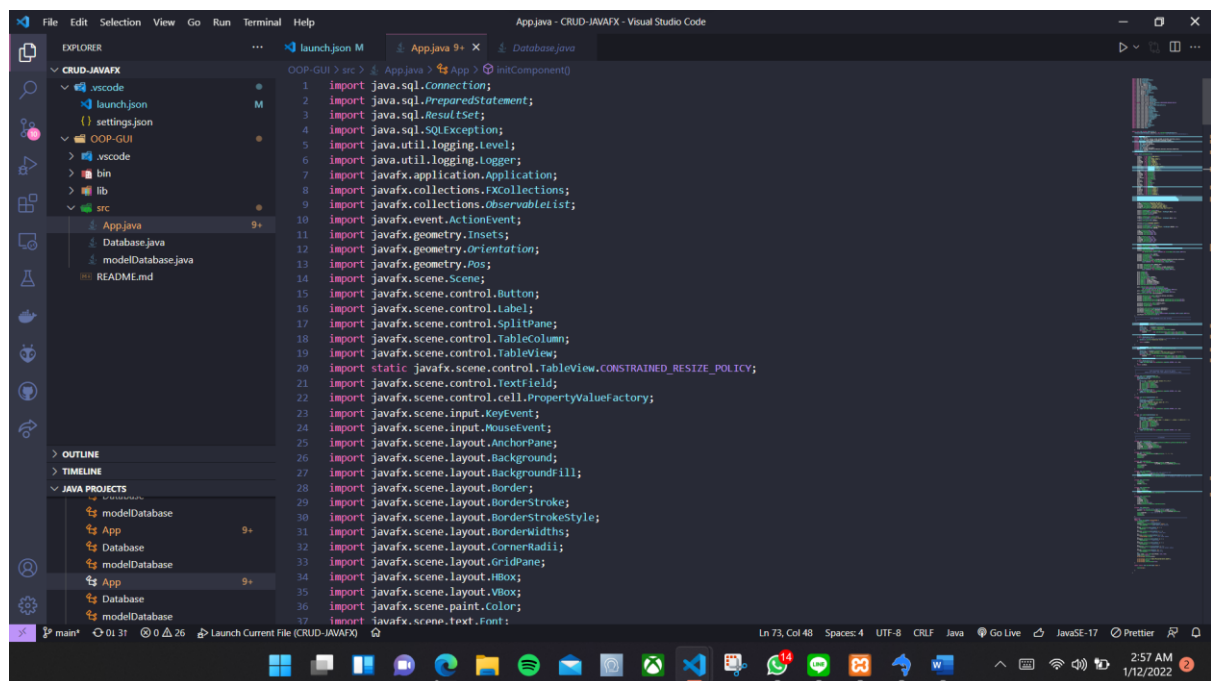
class tersebut terdapat getter dan setter. karena saya tidak mengizinkan sembarangan untuk mengubah dan mengambil nilai yang terdapat pada class modelDatabase ini. Konsep ini biasa disebut dengan enkapsulasi.

Disini saya menambahkan property dengan type data string dan modifier public final, karena disini saya tidak akan mengubah-ubah lagi propertynya. Property yang saya tambahkan yakni nk, nama, telpon, dan status

Lalu saya membuat sebuah constructor yang berfungsi untuk membuat proses awal dalam persiapan object, seperti memberi nilai kepada property, memanggil method internal serta beberapa proses lain yang dirasa perlu. Saya menambahkan parameter sesuai dengan property yang sudah saya buat.

Selanjutnya saya menambahkan method getter dan setternya untuk setiap property yang telah dibuat. Pada dasarnya metode get digunakan untuk mengembalikan nilai pada variable dengan menggunakan fungsi return dan set merupakan method void untuk mensetting atau memberikan nilai pada variable.

#### 4. Import library dan modifikasi javafx



Dapat terlihat pada gambar diatas bahwa ada banyak sekali library yang saya import untuk memasukan suatu method atau perintah dalam bahasa pemrograman Java sehingga perintah tersebut dapat aktif dan digunakan atau berfungsi. Semua import diatas saya gunakan untuk mendukung program yang saya buat

```

public class App extends Application {
    TableView<modelDatabase> tableView = new TableView<modelDatabase>();
    // ObservableList<modelDatabase> observableList = new ObservableList<modelDatabase>();

    public TableView tblView;
    private Text txtInfo;
    private Label lblTitle, lblData, lblNK, lblNAMA, lblTELPON, lblSTATUS, lblCari;
    public TextField txtNK, txtNAMA, txtTELPON, txtSTATUS, txtCARI;
    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;
    modelDatabase modelDb;
    ObservableList data = FXCollections.observableArrayList();
}

```

Lalu pada program selanjutnya saya membuat instance pertama untuk object tableview

Beberapa class yang saya deklarasikan, diantaranya:

- TableView: TableView saya gunakan untuk menampilkan tampilan tabel di dalam aplikasi JavaFX saya. TableView JavaFX diwakili oleh kelas `javafx.scene.control.TableView`.
- Teks JavaFX yang dapat menampilkan teks di dalam GUI JavaFX. Kontrol Teks JavaFX diwakili oleh kelas JavaFX `javafx.scene.text.Text`. supaya saya dapat mengatur font yang akan digunakan dengan kontrol Teks, ukuran teks, dekorasi font dan banyak hal lainnya.
- Label JavaFX untuk dapat menampilkan teks atau label gambar di dalam GUI JavaFX. Kontrol label harus ditambahkan ke grafik scene agar terlihat. Kontrol Label JavaFX diwakili oleh kelas `javafx.scene.control.Label`.
- TextField JavaFX untuk memasukkan teks yang kemudian dapat dibaca oleh aplikasi. Kontrol JavaFX TextField diwakili oleh kelas `javafx.scene.control.TextField`.
- Tablecolumn, berguna untuk membuat kolom pada tableview
- JavaFX SplitPane, adalah kontrol wadah yang dapat berisi beberapa komponen lain di dalamnya. Dengan kata lain, SplitPane dibagi antara kontrol yang dikandungnya. Antara kontrol di SplitPane adalah pembagi. Pengguna dapat memindahkan pembagi untuk mengatur berapa banyak ruang yang dialokasikan untuk setiap kontrol.
- JavaFX VBox adalah komponen tata letak yang berguna untuk menempatkan semua simpul anak (komponen) dalam kolom vertikal - di atas satu sama lain. Komponen JavaFX VBox diwakili oleh kelas `javafx.scene.layout.VBox`.
- AnchorPane pada JavaFX saya gunakan untuk menambatkan ke offset dari tepi panel jangkar oleh tepi simpul turunannya. Jika panel jangkar memiliki set pembatas atau bantalan, set off diukur dari tepi sudut bagian dalam inset.
- GridPane untuk menambahkan beberapa node dalam beberapa baris dan kolom. Ini dilihat sebagai kisi baris dan kolom yang fleksibel di mana node dapat ditempatkan di sel mana pun dari kisi. Ini diwakili oleh kelas `javafx.scene.layout.GridPane`. Kita hanya perlu membuat instance kelas ini untuk mengimplementasikan GridPane.

- JavaFX HBox adalah komponen tata letak yang memposisikan semua child node (komponen) dalam baris horizontal. Komponen Java HBox diwakili oleh kelas `javafx.scene.layout.HBox`.
- Button JavaFX untuk aplikasi JavaFX menjalankan beberapa tindakan saat pengguna aplikasi mengklik tombol. Kontrol Tombol JavaFX diwakili oleh kelas `javafx.scene.control.Button`. Tombol JavaFX dapat memiliki teks dan ikon di atasnya yang menunjukkan kepada pengguna apa yang akan dilakukan dengan mengklik tombol tersebut.

Lalu untuk kode selanjutnya saya membuat variable `modelDb` dari class `modelDatabase` agar nantinya saya lebih mudah untuk memanggil classnya saat diperlukan.

Yang terakhir saya ingin menampilkan list data dalam bentuk array maka dari itu saya perlu `ObservableList` sebagai output dari metode pengambilan data dari `DataBase`.

Dari semua class yang sudah saya deklarasikan diatas, saya hanya perlu membuat instance untuk setiap kelasnya untuk mengimplementasikan class-class tersebut. Seperti contoh pada gambar berikut:

```
public void initComponents(){
    //=====
    lblData    = new Label("FORM DATA");
    lblTitle   = new Label();
    lblNK      = new Label("NOMOR KAMAR");
    lblNAMA    = new Label("NAMA");
    lblTELPON  = new Label("TELPON");
    lblSTATUS  = new Label("STATUS");
    lblCari    = new Label("CARI DATA :");
    txtInfo    = new Text("No data");
    tblColumn1 = new TableColumn("NOMOR KAMAR");
    tblColumn2 = new TableColumn("NAMA");
    tblColumn3 = new TableColumn("TELPON");
    tblColumn4 = new TableColumn("STATUS");
    txtNK      = new TextField();
    txtNAMA    = new TextField();
    txtTELPON  = new TextField();
    txtSTATUS  = new TextField();
    txtCARI    = 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("INSERT");
    btnUpdate  = new Button("UPDATE");
    btnDelete  = new Button("DELETE");
    btnClear   = new Button("CLEAR");
    btnClose   = new Button("CLOSE");
    btnRefresh = new Button("REFRESH");
}
```



Dapat terlihat pada gambar diatas, saya membuat banyak instance untuk setiap property dan text yang saya inginkan

Untuk class label saya buat instance yang berisi : Form data, Nomor kamar, Nama, Telpon, Status, dan Cari data

Untuk class Tablecolumn berisi: Nomor kamar, nama, telpon, status

Dan untuk class Textfield saya tentukan: nk, nama, telpon, dan status

Lalu saya membuat instance untuk :

- SplitPane, dengan nama splitPaneh
- AnchorPane, dengan nama pane
- VBox, dengan nama panevbox dan panevbox2
- GridPane, dengan nama grid
- Hbox, dengan nama panehbox dan searchbox
- Tableview, dengan nama tblView

Dan terakhir saya membuat instance untuk class button yang berisi:

- INSERT
- UPDATE
- DELETE
- CLEAR
- CLOSE
- REFRESH

```
tblColumn1.setCellValueFactory(new PropertyValueFactory("nk"));
tblColumn2.setCellValueFactory(new PropertyValueFactory("nama"));
tblColumn3.setCellValueFactory(new PropertyValueFactory("telpon"));
tblColumn4.setCellValueFactory(new PropertyValueFactory("status"));

txtNK.setPromptText("Masukkan NK Anda");
txtNAMA.setPromptText("Masukkan Nama Anda");
txtTELpon.setPromptText("Masukkan Telpon Anda");
txtSTATUS.setPromptText("Masukkan Status Anda");
txtCARI.setPromptText("Masukkan data yang ingin dicari");

lblCari.setPadding(new Insets(10));
lblCari.setFont(Font.font("Arial Black", FontWeight.BOLD, 12));
lblCari.setAlignment(Pos.CENTER);
lblCari.setUnderline(true);

lblData.setPadding(new Insets(10));
lblData.setFont(Font.font("Arial Black", FontWeight.BOLD, 22));
// lblData.setUnderline(true);
lblData.setAlignment(Pos.CENTER);

lblTitle.setText("DATABASE KOSSAN");
// lblTitle.setUnderline(true);
lblTitle.setPadding(new Insets(10));
lblTitle.setFont(Font.font("Arial Black", FontWeight.MEDIUM, 22));
lblTitle.setAlignment(Pos.CENTER);

lblNK.setPrefSize(100, 30);
lblNAMA.setPrefSize(100, 30);
lblTELpon.setPrefSize(100, 30);
lblSTATUS.setPrefSize(100, 30);

txtNK.setPrefSize(250, 30);
txtNAMA.setPrefSize(250, 30);
txtTELpon.setPrefSize(250, 30);
txtSTATUS.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.DARKGRAY, new CornerRadii(10), Insets.EMPTY)));

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

grid.setHgap(10);
grid.setVgap(10);
grid.setLayoutX(5);
grid.setLayoutY(5);
grid.setAlignment(Pos.CENTER);
grid.setPadding(new Insets(10));
grid.addRow(1, lblNK, txtNK);
grid.addRow(2, lblNAMA, txtNAMA);
grid.addRow(3, lblTELpon, txtTELpon);
grid.addRow(4, lblSTATUS, txtSTATUS);
grid.setGridLinesVisible(false);

pane.setBorder(new Border(new BorderStroke(
    Color.WHITESMOKE, BorderStrokeStyle.DASHED,
```



```

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.LIGHTGREY, new CornerRadii(15), Insets.EMPTY)));
pane.getChildren().addAll(grid, panebox);

panebox.getChildren().addAll(lblTitle, tblView, searchbox);
panebox.setPadding(new Insets(5));
panebox.setSpacing(5);
panebox.minWidthProperty().bind(splitPaneH.widthProperty().multiply(0.65));
panebox.maxWidthProperty().bind(splitPaneH.widthProperty().multiply(0.65));

panebox2.getChildren().addAll(lblData, pane);
panebox2.setPadding(new Insets(5));
panebox2.setSpacing(5);

splitPaneH.setOrientation(Orientation.HORIZONTAL);
splitPaneH.getItems().addAll(panebox, panebox2);
splitPaneH.setPadding(new Insets(2));
splitPaneH.setBackground(new Background(
    new BackgroundFill(Color.DARKSLATEGRAY, CornerRadii.EMPTY, Insets.EMPTY)));
splitPaneH.setDividerPositions(0.5);
}

```

Pada class tablecolumn saya memanggil value-value yang ada pada modelDatabase.java. disinilah fungsi Get dibutuhkan. Value yang diambil mengacu pada Get yang sudah saya buat di class modelDatabase.java.

Selebihnya saya melakukan deklarasi dan modifikasi javafx sesuai keinginan saya seperti mengatur ukuran tulisan, tipe tulisan, warna, spasi, dll pada setiap instance yang sudah saya buat tadi, seperti pada gambar-gambar diatas agar dapat terlihat pada scene GUI

Modifikasi javafx saya pelajari di tutorial [JavaFX VBox \(jenkov.com\)](http://jenkov.com)

## 5. Menampilkan data yang ada pada database ke GUI Javafx

```

/**=====
 *
 *      UNTUK MENAMPUNG DATA DARI DATABASE
 *
 * =====
 */

private ObservableList loadData(){
    ObservableList listData = FXCollections.observableArrayList();
    try {
        Connection c    = Database.tryConnect();
        String sql1      = "select * from kost;";
        ResultSet rs1    = c.createStatement().executeQuery(sql1);
        while(rs1.next()){
            modelDb      = new modelDatabase(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);
        System.out.println("Connection failed!!!");
    }
    return listData;
}

```

Disini saya membuat function loadData() untuk menampilkan data. Diberi modifier private dengan class Observablelist untuk menampilkan data dalam bentuk array

- Membuat variable dengan nama listData untuk menampung method observableArrayList
- Membuat koneksi database dengan memanggil method tryConnect,
- membuat query mysql untuk menampilkan semua data pada table kost yang sudah saya buat.
- Menggunakan sebuah kondisi perulangan (while) untuk memanggil setiap value yang ada pada table kost.
- Terakhir melakukan return untuk mengembalikan value dari list data

## 6. Membuat sebuah fitur untuk mencari data

```
private ObservableList searchByNK(String n){
    ObservableList listData = FXCollections.observableArrayList();
    try {
        Connection c = Database.tryConnect();
        String sql2 = " select distinct * from kost where nk like '%" + n + "%'";
        ResultSet rs2 = c.createStatement().executeQuery(sql2);
        while(rs2.next()){
            modelDb = new modelDatabase(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;
}
```

Disini saya ingin membuat sebuah fitur untuk mencari data dengan nk atau nomor kamar yang sudah saya setting sebagai primary key sebelumnya.

Dengan cara yang masih sama seperti menampilkan data, namun saya sesuaikan dengan query mysql sesuai dengan perintah yang diinginkan

## 7. Menambahkan data pada database

```
/**=====
 *          UNTUK MELAKUKAN INSERT, DELETE DAN UPDATE
 *          DIMANA DATA DIAMBIL DARI FORM KEMUDIAN DIKUMPULKAN DI MODEL
 * =====
 */
private void insert(modelDatabase m){
    Connection c = Database.tryConnect();
    PreparedStatement ps;
    try {
        String sql = "insert into kost values (?, ?, ?, ?)";
        ps = c.prepareStatement(sql);
        ps.setString(1, m.getNk());
        ps.setString(2, m.getNama());
        ps.setString(3, m.getTelpon());
        ps.setString(4, m.getStatus());
        ps.execute();
    } catch (SQLException ex) {
        Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);
        System.out.println("Error");
    }
}
```

Masih dengan cara yang sama disini saya membuat sebuah program untuk menginput data lalu menambahkannya pada database yang sudah ada dengan syntax diatas dan dengan query mysql yang sudah disesuaikan

## 8. Menghapus data

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

Kode diatas adalah cara untuk membuat program menghapus data yang sudah ada pada database. Menghapus data berdasarkan nk sebagai primary keynya

## 9. Mengubah data

```
private void update(modelDatabase m){
    try {
        Connection c = Database.tryConnect();
        PreparedStatement ps;
        String sql = "update kost set nama = ? ,telpon = ? , status = ? where nk = ? ";
        ps = c.prepareStatement(sql);
        ps.setString(1, m.getNama());
        ps.setString(2, m.getTelpon());
        ps.setString(3, m.getStatus());
        ps.setString(4, m.getNk());
        ps.execute();
    } catch (SQLException ex) {
        Logger.getLogger(App.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Terakhir adalah program untuk mengubah data yang sudah ada pada database. Data yang diubah berdasarkan nk / nomor kamar sudah di set sebagai primary key

## 10. Membuat fungsi untuk action-event javafx

```
private void selectData(){
    modelDb = (modelDatabase) tblView.getSelectionModel().getSelectedItem().get(0);
    txtNK.setText(modelDb.getNK());
    txtNAMA.setText(modelDb.getNama());
    txtTELPON.setText(modelDb.getTelpon());
    txtSTATUS.setText(modelDb.getStatus());
    txtNK.setDisable(true);
}

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

private void updateData(){
    modelDb = new modelDatabase(txtNK.getText(),txtNAMA.getText(),
                                txtTELPON.getText(),txtSTATUS.getText());
    update(modelDb);
    clearData();
    showData();
}

private void searchbyNK(){
    data.clear(); // <- menghapus data pada penampung data
    data = searchByNK(txtCARI.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();
    txtCARI.clear();
}

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

private void clearData(){
    txtNK.clear();
    txtNAMA.clear();
    txtTELPON.clear();
    txtSTATUS.clear();
    txtNK.setDisable(false);
    tblView.getSelectionModel().clearSelection();
}

private void addData(){
    // mengambil data dari form, kemudian disusun seperti array
    modelDb = new modelDatabase(txtNK.getText(),txtNAMA.getText(),
                                txtTELPON.getText(),txtSTATUS.getText());
    insert(modelDb); //<- data dikirim ke SQL
    showData();
    clearData();
}
```

Jika pada kode sebelumnya saya telah membuat program CRUD, disini saya menambahkan sebuah fitur tambahan untuk mempermudah pengguna aplikasi saya. Diantaranya adalah:

- 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

## 11. Action Event

```
@Override
public void start(Stage primaryStage) {
    initComponents(); // <- 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
    });
    txtCARI.setOnKeyTyped((KeyEvent ke) -> {
        searchbyNK(); // <- SEARCH DATA BY NK
    });
    Scene scene = new Scene(splitPaneH, 1216, 618);
    scene.setFill(null);
    primaryStage.setScene(scene);

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

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 diatas menjelaskan bahwa sebelumnya saya telah membuat fitur button, function, dan program pendukung lainnya. Tentunya saya ingin button dan program yang saya buat dapat berfungsi dengan baik saat saya klik button 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

Pada bagian akhir program adalah untuk menampilkan scene sesuai dengan resolusi yang diinginkan saat saya minimize. Lalu syntax selanjutnya adalah untuk menambahkan judul pada stage window, dan terakhir adalah untuk menampilkan stage secara full screen atau dengan layer penuh saat pertama saya buka programnya.

## 12. RUN program

```
Run | Debug
public static void main(String[] args) {

    launch(args);

}
```

Terakhir adalah function main, dimana fungsi ini akan dipanggil paling pertama. Program akan dijalankan saat saya klik run.

## 13. Demo program

NOMOR KAMAR	NAMA	TELPON	STATUS
A01	Dede	090897	Mahasiswa
A02	Wilno	090879	Mahasiswa
A03	Roy	090897	Mahasiswa
A09	Wora	0908	Pekerja

Press ESC to exit full-screen mode.

CARI DATA :

**FORM DATA**

NOMOR KAMAR

NAMA

TELPON

STATUS

Gambar diatas menunjukkan hasil program yang sudah saat buat tadi. Dapat terlihat bahwa database berserta konten di dalamnya sudah terlihat pada scene GUI. Selanjutnya saya akan mencoba fitur-fitur yang tersedia di dalamnya.

#### 14. Insert data

**DATABASE KOSSAN**

NOMOR KAMAR	NAMA	TELPON	STATUS
A01	Dede	090897	Mahasiswa
A02	Wilno	090879	Mahasiswa
A03	Roy	090897	Mahasiswa
A09	Wora	0908	Pekerja

**CARI DATA :**

**FORM DATA**

NOMOR KAMAR

B13

NAMA

Samuel

TELPON

090896

STATUS

Mahasiswa

INSERT

UPDATE

DELETE

CLEAR

CLOSE

**DATABASE KOSSAN**

NOMOR KAMAR	NAMA	TELPON	STATUS
A01	Dede	090897	Mahasiswa
A02	Wilno	090879	Mahasiswa
A03	Roy	090897	Mahasiswa
A09	Wora	0908	Pekerja
B13	Samuel	090896	Mahasiswa

**CARI DATA :**

**FORM DATA**

NOMOR KAMAR

Masukkan NK Anda

NAMA

Masukkan Nama Anda

TELPON

Masukkan Telpn Anda

STATUS

Masukkan Status Anda

INSERT

UPDATE

DELETE

CLEAR

CLOSE

#### 15. Update data

**DATABASE KOSSAN**

NOMOR KAMAR	NAMA	TELPON	STATUS
A01	Dede	090897	Mahasiswa
A02	Wilno	090879	Mahasiswa
A03	Roy	090897	Mahasiswa
A09	Wora	0908	Pekerja
B13	Samuel	090896	Mahasiswa

**CARI DATA :**

**FORM DATA**

NOMOR KAMAR

B13

NAMA

Samuel

TELPON

090896

STATUS

Pekerja

INSERT

UPDATE

DELETE

CLEAR

CLOSE



### DATABASE KOSSAN

NOMOR KAMAR	NAMA	TELPON	STATUS
A01	Dede	090897	Mahasiswa
A02	Wilno	090879	Mahasiswa
A03	Roy	090897	Mahasiswa
A09	Wora	0908	Pekerja
B13	Samuel	090896	Pekerja

**CARI DATA :**

### FORM DATA

NOMOR KAMAR:

NAMA:

TELPON:

STATUS:

## 16. Delete data

### DATABASE KOSSAN

NOMOR KAMAR	NAMA	TELPON	STATUS
A01	Dede	090897	Mahasiswa
A02	Wilno	090879	Mahasiswa
A03	Roy	090897	Mahasiswa
A09	Wora	0908	Pekerja
B13	Samuel	090896	Pekerja

**CARI DATA :**

### FORM DATA

NOMOR KAMAR:

NAMA:

TELPON:

STATUS:

### DATABASE KOSSAN

NOMOR KAMAR	NAMA	TELPON	STATUS
A01	Dede	090897	Mahasiswa
A02	Wilno	090879	Mahasiswa
A03	Roy	090897	Mahasiswa
A09	Wora	0908	Pekerja

**CARI DATA :**

### FORM DATA

NOMOR KAMAR:

NAMA:

TELPON:

STATUS:

Dapat terlihat pada demo diatas, bahwa program CRUD yang saya buat dapat berfungsi dengan baik.