

Introduction to JavaFX

Untuk membuat aplikasi berbasis GUI dengan bahasa pemrograman Java, Oracle telah menyediakan JavaFX. JavaFX pada dasarnya mirip dengan Java Swing. Tidak ada perbedaan yang signifikan antara JavaFX dengan Java Swing. Hanya saja JavaFX dapat diimplementasikan diberbagai *device* yang ada, seperti *mobile*.

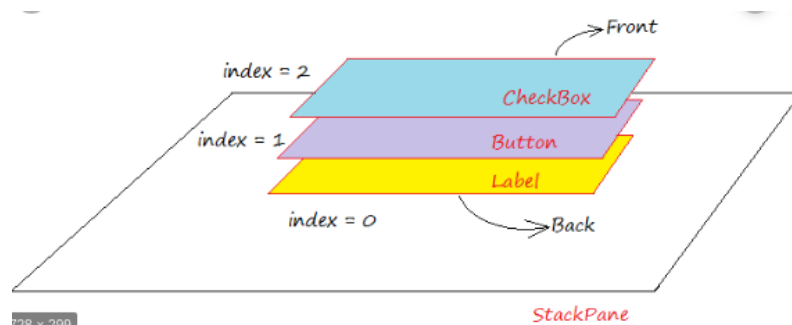
Ketika membuat aplikasi GUI dengan Java, anda perlu meng-*extends* sebuah *class abstract Application* (*javafx.application*). Kelas ini menyediakan segala “kelas inti” yang diperlukan ketika anda membuat aplikasi GUI. Karena anda meng-*extends* kelas *abstract* ini, maka anda perlu mengimplementasikan beberapa *abstract method* nya, seperti **start method** yang akan menjalankan aplikasi GUI.

Ada **tiga** hal penting yang perlu anda ingat ketika membuat *GUI-based application* dengan Java:

- Stage (*javafx.stage*) – men-*define* top-level container untuk semua *interface object*. Ibarat *canvas* yang menjadi dasar/alas dari segala *object* yang ada di atasnya. Ingat! Dalam sebuah aplikasi JavaFX hanya ada **satu stage**. Ketika anda meng-*extends* *abstract class Application*, maka *method start* yang akan di-*override* telah menyediakan *stage* sendiri (dari parameternya). Anda tinggal pakai saja.
- Scene (*javafx.scene*) – merupakan sebuah “holder” yang memegang kendali atas komponen-komponen atau *nodes* yang anda “letakkan” di atas *scene*.
- Nodes – merupakan komponen-komponen yang anda perlukan. Contoh: *button*, *textbox*, *dropdown*, dsb.

Di dalam sebuah *scene* anda bebas menggunakan *pane* atau tidak. Apa yang dimaksud dengan *pane*? *Pane* merupakan metode/teknik yang memudahkan kita didalam mengatur posisi dan ukuran komponen/*nodes* di aplikasi kita. Ada beberapa *pane* yang ada di JavaFX:

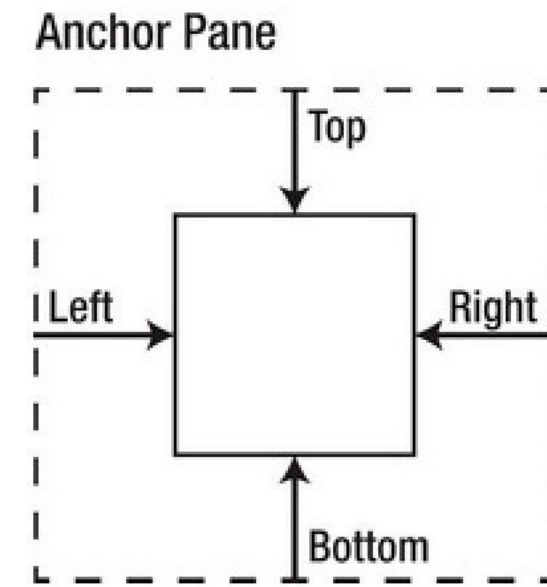
- **StackPane**: menampilkan *nodes* yang tumpang-tindih satu dengan lainnya. Misalnya, jika anda membuat sebuah *rectangle shape* dan kemudian menambah *circle shape*, maka yang muncul di-atasnya itu adalah *rectangle shape*, sedangkan *circle shape* sudah ditimpa.



Ilustrasi 1: **StackPane** yang menunjukkan tumpang-tindih *nodes* satu dengan lainnya

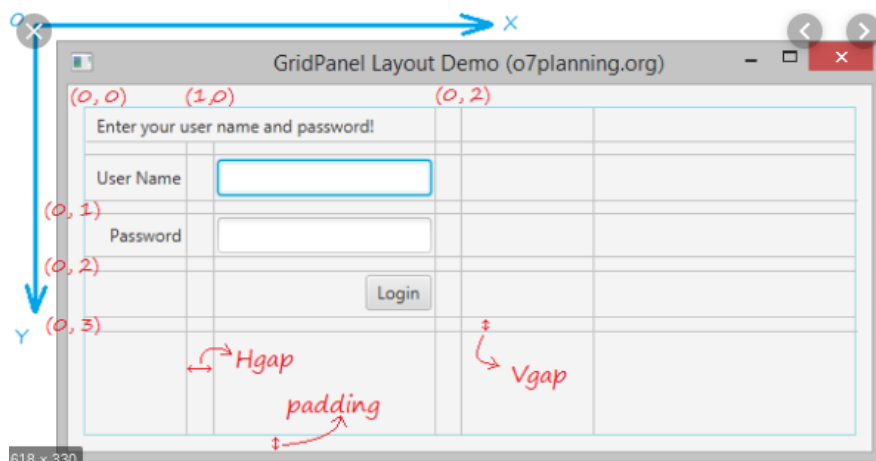
- **AnchorPane**: dengan menggunakan *anchorpane* anda dapat memposisikan *nodes* yang ada ke setiap sudut-sudut *panel* (*top*, *right*, *bottom*, *left*, dan *center*). Ketika anda menggunakan *anchorpane* mirip dengan **borderpane**, namun bedanya adalah:
 - Kalau di **borderpane** layout dibagi menjadi 4 area (*top*, *right*, *bottom*, *left*, dan *center*), sedangkan *anchorpane* tidak dibagi meskipun dapat memposisikan setiap *node* pada area tertentu. Pada *borderpane*, sebuah *node* hanya dapat diposisikan ke **sebuah** area (one-to-one), misal: *top* atau *left*, dsb. Sedangkan *anchorpane* memungkinkan anda bisa meletakkan lebih dari satu area (one-to-many).

- Karena *anchorpane* adalah one-to-many maka sebuah *node* dapat ter-*stretch* apabila ditarik dari sudut yang berlawanan. Misal: sebuah aplikasi dengan ukuran window 300x500, kemudian dibuat sebuah *rectangle* yang ukurannya 200x100 yang diposisikan di area *left*, lalu diposisikan lagi ke *right*, maka ukurannya **tidak lagi** 200x100, melainkan sudah menjadi 200x500, hal ini dikarenakan sudah di-*stretch* dari kiri ke kanan. Hal ini **tidak dapat** dilakukan di *borderpane*.



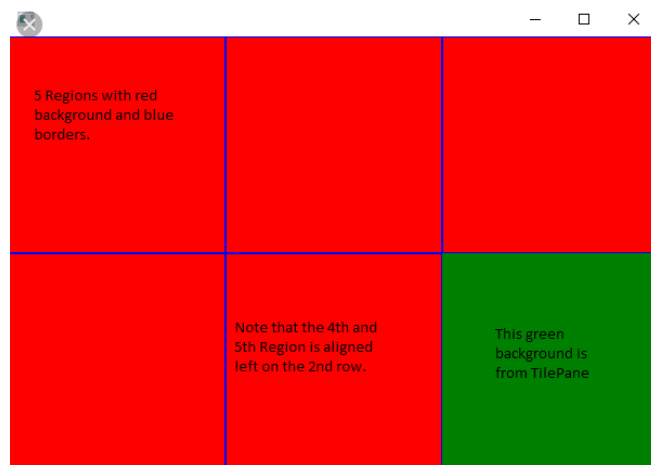
Ilustrasi 2: Posisi sebuah square dengan menerapkan *anchorpane*

- **GridPane:** ketika anda menggunakan *gridpane* anda dapat mengatur *node-node* dengan konsep *row-column*. Ketika anda bekerja dengan *gridpane* juga, anda harus terbiasa dengan konsep *colspan* dan *rowspan* yang ada di HTML. Ketika anda membayangkan sebuah *grid*, ukuran tiap-tiap *cell* berbeda, **tidak seperti** ukuran *cell* “papan catur”, yang sama antara satu *cell* dengan *cell* lainnya. Untuk praktikum kali ini, anda akan menggunakan *gridpane*.



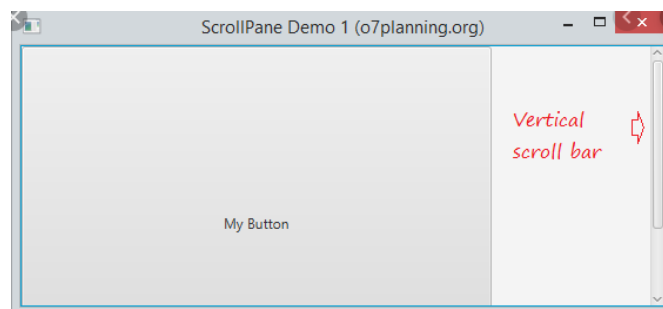
Ilustrasi 3: Memposisikan *nodes/komponen* jauh lebih mudah dengan *gridpane*

- **TilePane:** mirip dengan *gridpane*, bedanya adalah pada *tilepane* ini setiap *cell* itu memiliki ukuran yang sama (contoh: papan catur).



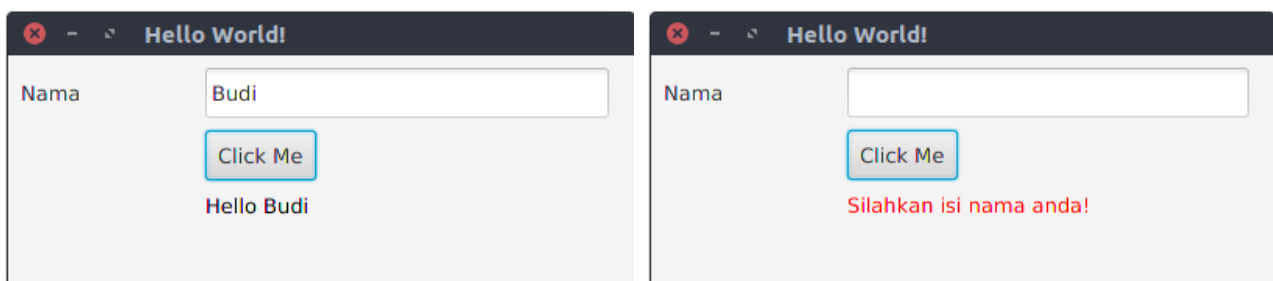
Ilustrasi 4: Ukuran tiap-tiap cell yang sama bila menggunakan tilepane

- **ScrollPane:** *pane* yang satu ini sebenarnya tidak dikategorikan sebagai *pane* yang sebenarnya. Hal ini dikarenakan kelas *scrollpane* **tidak** diturunkan dari kelas **Pane**, melainkan dari kelas **Control**. Akan tetapi, apabila anda ingin membuat aplikasi dengan ukuran window yang sangat besar sehingga diperlukan *scroll bar*, maka anda dapat memiliki *scrollpane*.



Ilustrasi 5: Gunakan scrollpane apabila ukuran window aplikasi sangat besar

Untuk menambah pemahaman anda, buatlah sebuah aplikasi sederhana yang menampilkan pesan “Hello Budi” ketika *text-box* diisi dengan nama ‘Budi’ lalu menekan tombol “Click Me”. Apabila *user* tidak mengisi *text-box* maka akan ada pesan “Silahkan isi nama anda!”. Pada praktikum kita kali ini, akan difokuskan hanya pada *gridpane*, untuk *pane* lainnya, anda dapat melakukan eksplorasi.



```

6 package hellobudiapp;
7
8 import javafx.application.Application;
9 import javafx.event.ActionEvent;
10 import javafx.event.EventHandler;
11 import javafx.geometry.Insets;
12 import javafx.scene.Scene;
13 import javafx.scene.control.*;
14 import javafx.scene.layout.*;
15 import javafx.scene.paint.Color;
16 import javafx.stage.Stage;
17
18 /**
19  *
20  * @author teamsar
21  */
22 public class HelloBudiApp extends Application {
23

```

```

24 @Override
25 public void start(Stage primaryStage) {
26     GridPane grdPnl = new GridPane();
27     Label lblNama = new Label("Nama");
28     Label lblOutput = new Label();
29     TextField txtNama = new TextField();
30     Button btn = new Button();
31     btn.setText("Click Me");
32
33     btn.setOnAction(new EventHandler<ActionEvent>() {
34         @Override
35         public void handle(ActionEvent event) {
36             if(txtNama.getText().trim().isEmpty()){
37                 lblOutput.setTextFill(Color.RED);
38                 lblOutput.setText("Silahkan isi nama anda!");
39             }
40             else{
41                 lblOutput.setTextFill(Color.BLACK);
42                 lblOutput.setText("Hello " + txtNama.getText().trim());
43             }
44         }
45     });
46
47     grdPnl.add(lblNama, 0, 0); grdPnl.add(txtNama, 1, 0);
48     grdPnl.add(btn, 1, 1, 2, 1);
49     grdPnl.add(lblOutput, 1, 2, 2, 1);
50
51     txtNama.setPadding(new Insets(8, 8, 8, 3));
52     btn.setPadding(new Insets(8));
53
54     ColumnConstraints cons1 = new ColumnConstraints();
55     ColumnConstraints cons2 = new ColumnConstraints();
56     cons1.setPercentWidth(30);
57     cons2.setPercentWidth(70);
58
59     grdPnl.getColumnConstraints().addAll(cons1, cons2); // span column
60     grdPnl.setPadding(new Insets(8, 8, 0, 8));
61     grdPnl.setHgap(8); grdPnl.setVgap(8);
62     Scene scene = new Scene(grdPnl, 400, 150);
63
64     primaryStage.setTitle("Hello World!");
65     primaryStage.setScene(scene);
66     primaryStage.show();
67 }

```

```

68
69  /**
70   * @param args the command line arguments
71   */
72  public static void main(String[] args) {
73      launch(args);
74  }
75
76  }
77

```

Penjelasan:

Bila anda perhatikan kode program 37 dan 41, itu menunjukkan bagaimana memberi warna pada *font label*. Sedangkan *line* 47-49, dengan menggunakan fungsi *add* anda dapat memposisikan sebuah *node*/komponen berdasarkan *row* dan *column*. Contoh untuk kode pada baris ke-47, disana label **lblNama** diposisikan pada *column* 0 dan *row* 0 (Ingat! Kolom duluan baru baris – beda kayak biasanya yang dimulai dari baris dulu baru kolom). Pada baris ke-49, disana menunjukkan bahwa label **lblOutput** diposisikan pada *column* ke-1 dan *row* ke-2, dan disana juga terjadi *colspan* sebesar 2 dan *rowspan* 1 (lihat *value* pada argumen ke-4 dan ke-5 yang nilainya berturut-turut adalah 2 dan 1).

Penggunaan *padding* pada baris kode 51 dan 52 menunjukkan “spasi” yang diberikan sebesar 8. lihat [dokumentasi javafx](#) untuk lebih jelasnya. Sedangkan penggunaan fungsi **setHgap** (horizontal) dan **setVgap** (vertikal) pada baris ke-61 menunjukkan bagaimana memberi “jarak” antara satu *node* dengan *node* yang lainnya.

JavaFX – JDBC in action!

Berikut ini contoh bagaimana membuat aplikasi JavaFX sederhana yang terhubung dengan JDBC. Perhatikan kode programnya baik-baik (mohon jangan asal ketik) dan cari tahu penggunaan fungsi-fungsi yang tidak anda mengerti. Lakukan eksplorasi sebanyak mungkin! Note: gunakan *database PROFIL* yang telah anda buat di praktikum minggu lalu.

Kelas: TMahasiswa.java

```

6  package javafxjdbcapp;
7
8  import java.sql.Date;
9
10 /**
11  *
12  * @author teamsar
13  */
14 public class TMahasiswa {
15     private String _nim, _nama, _email;
16     private Date _dob;
17
18     public void setNim(String nim){
19         _nim = nim;
20     }
21
22     public void setNama(String nama){
23         _nama = nama;
24     }
25

```

```

26 public void setEmail(String email){
27     _email = email;
28 }
29
30 public void setDob(Date dob){
31     _dob = dob;
32 }
33
34 public String getNim(){ return _nim; }
35 public String getNama(){ return _nama; }
36 public String getEmail(){ return _email; }
37 public Date getDob(){ return _dob; }
38 }

```

Kelas: TMatakuliah.java

```

6 package javafxjdbcapp;
7
8 /**
9  *
10  * @author teamsar
11  */
12 public class TMatakuliah {
13     private String idMatkul, namaMatkul;
14     private int sks;
15
16     public void setIDMatkul(String idMatkul){ this.idMatkul = idMatkul; }
17     public void setNamaMatkul(String namaMatkul){ this.namaMatkul = namaMatkul; }
18     public void setSks(int sks){ this.sks = sks; }
19
20     public String getIdMatkul(){ return idMatkul; }
21     public String getNamaMatkul(){ return namaMatkul; }
22     public int getSks(){ return sks; }
23 }
24

```

Kelas: TMkMhs.java

```

6 package javafxjdbcapp;
7
8 /**
9  *
10  * @author teamsar
11  */
12 public class TMkMhs {
13     private String nim, id_matkul;
14
15     public void setNim(String nim){ this.nim = nim; }
16     public void setIdMatkul(String id_matkul){ this.id_matkul = id_matkul; }
17
18     public String getNim(){ return nim; }
19     public String getIdMatkul(){ return id_matkul; }
20 }
21

```

Kelas: DBUtils.java

```
6 package javafxjdbcapp;
7
8 import java.sql.Connection;
9 import java.sql.DriverManager;
10 import java.sql.PreparedStatement;
11 import java.sql.ResultSet;
12 import java.sql.SQLException;
13 import java.util.ArrayList;
14 import java.util.List;
15
16 /**
17  *
18  * @author teamsar
19  */
20 public class DBUtils {
21
22     public static Connection getConnection() throws SQLException,
23         ClassNotFoundException{
24         String className = "com.mysql.cj.jdbc.Driver";
25         Class.forName(className);
26         System.out.println("Driver loaded successfully!");
27
28         Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306"
29             + "/PROFIL", "root", "newpassword123");
30
31         if(con.isClosed()) return null;
32         else return con;
33     }
34
35     public static void closeConnection(Connection con) throws SQLException{
36         con.close();
37     }
38
39     public static int saveMatakuliah(TMatakuliah obj) throws SQLException,
40         ClassNotFoundException{
41         Connection con = getConnection();
42         String _sql = "INSERT INTO TMatakuliah(ID_MATKUL, NAMA_MATKUL, SKS) "
43             + "VALUES(?, ?, ?)";
44         PreparedStatement ps = con.prepareStatement(_sql);
45         ps.setString(1, obj.getIdMatkul());
46         ps.setString(2, obj.getNamaMatkul());
47
48         ps.setInt(3, obj.getSks());
49
50         int response = ps.executeUpdate();
51         ps.close();
52         con.close();
53
54         return response;
55     }
56
57     public static int saveMahasiswa(TMahasiswa obj) throws SQLException,
58         ClassNotFoundException{
59         Connection con = getConnection();
60         String _sql = "INSERT INTO TMahasiswa(NIM, NAMA, DOB, EMAIL) "
61             + "VALUES(?, ?, ?, ?)";
62         PreparedStatement ps = con.prepareStatement(_sql);
63         ps.setString(1, obj.getNim());
64         ps.setString(2, obj.getNama());
65         ps.setDate(3, obj.getDob());
66         ps.setString(4, obj.getEmail());
67
68         int response = ps.executeUpdate();
69         ps.close();
70         con.close();
71
72         return response;
73     }
74 }
```

```

74 public static int saveKrs(TMkMhs obj) throws SQLException,
75     ClassNotFoundException{
76     Connection con = getConnection();
77     String _sql = "INSERT INTO TMkMhs(NIM, ID_MATKUL) VALUES(?, ?)";
78     PreparedStatement ps = con.prepareStatement(_sql);
79     ps.setString(1, obj.getNim());
80     ps.setString(2, obj.getIdMatkul());
81
82     int response = ps.executeUpdate();
83     ps.close();
84     con.close();
85
86     return response;
87 }
88
89 public static List<TMahasiswa> getAllMahasiswa() throws SQLException,
90     ClassNotFoundException{
91     Connection con = getConnection();
92     String _sql = "SELECT * FROM TMahasiswa";
93     PreparedStatement ps = con.prepareStatement(_sql);
94
95     List<TMahasiswa> lstMahasiswa = new ArrayList<>();
96     ResultSet rs = ps.executeQuery();
97
98     TMahasiswa objMhs;
99     while(rs.next()){
100         objMhs = new TMahasiswa();
101         objMhs.setNim(rs.getString("NIM"));
102         objMhs.setNama(rs.getString("NAMA"));
103         objMhs.setDob(rs.getDate("DOB"));
104         objMhs.setEmail(rs.getString("EMAIL"));
105
106         lstMahasiswa.add(objMhs); // record to list
107     }
108
109     return lstMahasiswa;
110 }
111
112 public static List<TMatakuliah> getAllMatakuliah() throws SQLException,
113     ClassNotFoundException{
114     Connection con = getConnection();
115     String _sql = "SELECT * FROM TMatakuliah";
116     PreparedStatement ps = con.prepareStatement(_sql);
117
118     List<TMatakuliah> lstMatkul = new ArrayList<TMatakuliah>();
119     ResultSet rs = ps.executeQuery();
120
121     TMatakuliah objMatkul;
122     while(rs.next()){
123         objMatkul = new TMatakuliah();
124         objMatkul.setIDMatkul(rs.getString("ID_MATKUL"));
125         objMatkul.setNamaMatkul(rs.getString("NAMA_MATKUL"));
126         objMatkul.setSks(rs.getInt("SKS"));
127
128         lstMatkul.add(objMatkul); // record to list
129     }
130
131     return lstMatkul;
132 }

```



```

134 public static List<TMkMhs> getAllKrs() throws SQLException,
135     ClassNotFoundException{
136     Connection con = getConnection();
137     String _sql = "SELECT * FROM TMkMhs";
138     PreparedStatement ps = con.prepareStatement(_sql);
139
140     List<TMkMhs> lstMkMhs = new ArrayList<TMkMhs>();
141     ResultSet rs = ps.executeQuery();
142
143     TMkMhs objMkMhs;
144     while(rs.next()){
145         objMkMhs = new TMkMhs();
146         objMkMhs.setIdMatkul(rs.getString("ID_MATKUL"));
147         objMkMhs.setNim((rs.getString("NIM")));
148
149         lstMkMhs.add(objMkMhs); // record to list
150     }
151
152     return lstMkMhs;
153 }
154 }

```

Kelas: GeneralUtils.java

```

6 package javafxjdbcapp;
7
8 import java.text.ParseException;
9 import java.text.SimpleDateFormat;
10 import java.util.Date;
11 import java.util.Iterator;
12 import java.util.List;
13
14 /**
15  *
16  * @author teamsar
17  */
18 public class GeneralUtils {
19     public static java.sql.Date convertToDateFromString(String date,
20         String format) throws ParseException{
21         SimpleDateFormat sdf1 = new SimpleDateFormat(format);
22         Date dt = sdf1.parse(date);
23         return new java.sql.Date(dt.getTime());
24     }
25
26     public static TMahasiswa findMahasiswa(String nim,
27         List<TMahasiswa> listMahasiswa) {
28         Iterator<TMahasiswa> iterator = listMahasiswa.iterator();
29         while (iterator.hasNext()) {
30             TMahasiswa mhs = iterator.next();
31             if (mhs.getNim().toLowerCase().equals(nim)) return mhs;
32         }
33         return null;
34     }
35
36     public static TMatakuliah findMatakuliah(String kodeMatkul,
37         List<TMatakuliah> listMatakuliah) {
38         Iterator<TMatakuliah> iterator = listMatakuliah.iterator();
39         while (iterator.hasNext()) {
40             TMatakuliah mkl = iterator.next();
41             if (mkl.getIdMatkul().toLowerCase().equals(kodeMatkul)) return mkl;
42         }
43         return null;
44     }

```

```

    public static TMkMhs checkKRSExist(String kodeMatkul,
47     String nim, List<TMkMhs> listMkMhs) {
48         Iterator<TMkMhs> iterator = listMkMhs.iterator();
49         while (iterator.hasNext()) {
50             TMkMhs mkmhs = iterator.next();
51             if (mkmhs.getIdMatkul().toLowerCase().equals(kodeMatkul) &&
52                 mkmhs.getNim().toLowerCase().equals(nim)) return mkmhs;
53         }
54         return null;
55     }
56 }

```

Kelas: JavaFXJDBCApp.java (driver)

```

6     package javafxjdbcapp;
7
8     import java.sql.SQLException;
9     import java.text.ParseException;
10    import java.time.LocalDate;
11    import java.time.format.DateTimeFormatter;
12    import java.util.List;
13    import java.util.Locale;
14    import java.util.logging.Level;
15    import java.util.logging.Logger;
16    import javafx.application.Application;
17    import javafx.beans.value.ObservableValue;
18    import javafx.event.*;
19    import javafx.geometry.Insets;
20    import javafx.scene.Scene;
21    import javafx.scene.control.*;
22    import javafx.scene.layout.*;
23    import javafx.scene.paint.Color;
24    import javafx.stage.Stage;
25    import javafx.util.StringConverter;
26    import javafx.beans.value.ChangeListener;
27
28    /**
29     *
30     * @author teamsar
31     */
32    public class JavaFXJDBCApp extends Application{
33        public static List<TMahasiswa> listMahasiswa;
34        public static List<TMatakuliah> listMatakuliah;
35        public static List<TMkMhs> listKrs;
36
37        @Override
38        public void start(Stage primaryStage) {
39            // kumpulan nodes/komponen di masing-masing tab
40            TabPane tp = new TabPane();
41            Tab tab1 = new Tab("Mahasiswa");
42            Tab tab2 = new Tab("Mata Kuliah");
43            Tab tab3 = new Tab("Isi KRS");
44            tp.getTabs().addAll(tab1, tab2, tab3);
45
46            // buat grid untuk tiap-tiap tab
47            GridPane grdPaneMhs = new GridPane();
48            GridPane grdPaneMatKul = new GridPane();
49            GridPane grdKrs = new GridPane();

```

```

51 tab1.setContent(grdPaneMhs); // letakkan grid pane mahasiswa di tab 1
52 tab2.setContent(grdPaneMatKul); // letakkan grid pane matkul di tab 2
53 tab3.setContent(grdKrs); // letakkan grid pane krs di tab 3
54
55 // kasih padding untuk komponen yang berada di masing2 grid sebesar 8
56 grdPaneMhs.setPadding(new Insets(8));
57 grdPaneMatKul.setPadding(new Insets(8));
58 grdKrs.setPadding(new Insets(8));
59
60 // kasih jarak tiap-tiap komponen yang berada di masing-masing tab
61 // sebesar 8
62 grdPaneMhs.setHgap(8); grdPaneMhs.setVgap(8);
63 grdPaneMatKul.setHgap(8); grdPaneMatKul.setVgap(8);
64 grdKrs.setHgap(8); grdKrs.setVgap(8);
65
66 // set proporsi antara label dengan text inputan 30:70
67 ColumnConstraints cons1 = new ColumnConstraints();
68 ColumnConstraints cons2 = new ColumnConstraints();
69 cons1.setPercentWidth(30);
70 cons2.setPercentWidth(70);
71
72 // isi konten grid pane dosen dengan berbagai komponen yang diperlukan
73 // tab 1 (mahasiswa)
74 Label lblNama = new Label("Nama Mahasiswa");
75 TextField txtNama = new TextField();
76 Label lblNim = new Label("NIM");
77 TextField txtNim = new TextField();
78 Label lblDob = new Label("Date of Birth");
79 DatePicker dpDob = new DatePicker(LocalDate.now());
80 dpDob.setEditable(false);
81
82 dpDob.setConverter(new StringConverter<LocalDate>() {
83     String pattern = "dd/MM/yyyy";
84     DateTimeFormatter dateFormatter = DateTimeFormatter.
85         ofPattern(pattern, Locale.US);
86     {
87         dpDob.setPromptText(pattern.toLowerCase());
88     }
89
90     @Override public String toString(LocalDate date) {
91         if (date != null) {
92             return dateFormatter.format(date);
93         } else {
94             return "";
95         }
96     }
97
98     @Override public LocalDate fromString(String string) {
99         if (string != null && !string.isEmpty()) {
100             return LocalDate.parse(string, dateFormatter);
101         } else {
102             return null;
103         }
104     }
105 });
106
107 Label lblEmail = new Label("Email");
108 TextField txtEmail = new TextField();
109 Label lblInfoTab1 = new Label();
110 lblInfoTab1.setStyle("-fx-font-style: italic");
111
112 Button btnSaveTab1 = new Button();
113 btnSaveTab1.setPadding(new Insets(8));
114 btnSaveTab1.setText("Simpan Data Mahasiswa");

```

```

116 grdPaneMhs.setColumnConstraints().addAll(cons1, cons2);
117 grdPaneMhs.add(lblNama, 0, 0); grdPaneMhs.add(txtNama, 1, 0);
118 grdPaneMhs.add(lblNim, 0, 1); grdPaneMhs.add(txtNim, 1, 1);
119 grdPaneMhs.add(lblDob, 0, 2); grdPaneMhs.add(dpDob, 1, 2);
120 grdPaneMhs.add(lblEmail, 0, 3); grdPaneMhs.add(txtEmail, 1, 3);
121 grdPaneMhs.add(btnSaveTab1, 1, 4);
122 grdPaneMhs.add(lblInfoTab1, 1, 5);
123
124 btnSaveTab1.setOnAction((ActionEvent event) -> {
125     if(txtNim.getText().trim().isEmpty() ||
126         txtNama.getText().trim().isEmpty() ||
127         txtEmail.getText().trim().isEmpty()){
128         lblInfoTab1.setTextFill(Color.RED);
129         lblInfoTab1.setText("Silahkan isi dengan lengkap setiap "
130             + "kolom yang ada!");
131     }else{
132         TMahasiswa obj = new TMahasiswa();
133         obj.setNim(txtNim.getText().trim());
134         obj.setNama(txtNama.getText().trim());
135         obj.setEmail(txtEmail.getText().trim());
136         try {
137             obj.setDob(GeneralUtils.convertToDateFromString(dpDob.
138                 getEditor().getText(), "dd/MM/yyyy"));
139         } catch (ParseException ex) {
140             Logger.getLogger(JavaFXJDBCApp.class.getName()).
141                 log(Level.SEVERE, null, ex);
142         }
143
144         try {
145             int response = DBUtils.saveMahasiswa(obj);
146             loadAllData();
147             if(response > 0){
148                 lblInfoTab1.setTextFill(Color.BLACK);
149                 lblInfoTab1.setText("Data is successfully saved");
150             }else{
151                 lblInfoTab1.setTextFill(Color.RED);
152                 lblInfoTab1.setText("Unable to save the data!");
153             }
154         } catch (SQLException ex) {
155             Logger.getLogger(JavaFXJDBCApp.class.getName()).
156                 log(Level.SEVERE, null, ex);
157         } catch (ClassNotFoundException ex) {
158             Logger.getLogger(JavaFXJDBCApp.class.getName()).
159                 log(Level.SEVERE, null, ex);
160         }
161     }
162 });
163 // end
164 // tab 2 (mata kuliah)
165 Label lblIdMatKul = new Label("Kode Mata Kuliah");
166 TextField txtIdMatKul = new TextField();
167 Label lblNamaMatKul = new Label("Nama Mata Kuliah");
168 TextField txtNamaMatKul = new TextField();
169 Label lblJlhSks = new Label("Jumlah SKS");
170 TextField txtJlhSks = new TextField();
171 Label lblInfoTab2 = new Label();
172 lblInfoTab2.setStyle("-fx-font-style: italic");
173
174 // force the field to be numeric only
175 txtJlhSks.textProperty().addListener((ObservableValue extends String
176     observable, String oldValue, String newValue) -> {
177         if (!newValue.matches("\\d*")) {
178             txtJlhSks.setText(newValue.replaceAll("[^\\d]", ""));
179         }
180     });
181
182

```

```

184 Button btnSaveTab2 = new Button();
185 btnSaveTab2.setPadding(new Insets(8));
186 btnSaveTab2.setText("Simpan Data Mata Kuliah");
187
188 grdPaneMatKul.getColumnConstraints().addAll(cons1, cons2);
189 grdPaneMatKul.add(lblIdMatKul, 0, 0);
190 grdPaneMatKul.add(txtIdMatKul, 1, 0);
191 grdPaneMatKul.add(lblNamaMatKul, 0, 1);
192 grdPaneMatKul.add(txtNamaMatKul, 1, 1);
193 grdPaneMatKul.add(lblJlhSks, 0, 2);
194 grdPaneMatKul.add(txtJlhSks, 1, 2);
195 grdPaneMatKul.add(btnSaveTab2, 1, 3);
196 grdPaneMatKul.add(lblInfoTab2, 1, 4);
197
198 btnSaveTab2.setOnAction((t) -> {
199     if(txtIdMatKul.getText().trim().isEmpty() ||
200        txtNamaMatKul.getText().trim().isEmpty() ||
201        txtJlhSks.getText().trim().isEmpty()){
202         lblInfoTab2.setTextFill(Color.RED);
203         lblInfoTab2.setText("Silahkan isi dengan lengkap setiap "
204            + "kolom yang ada!");
205     }else{
206         TMatakuliah obj = new TMatakuliah();
207         obj.setIDMatkul(txtIdMatKul.getText().trim());
208         obj.setNamaMatkul(txtNamaMatKul.getText().trim());
209         obj.setSks(Integer.valueOf(txtJlhSks.getText().trim()));
210
211         try {
212             int response = DBUtils.saveMatakuliah(obj);
213             loadAllData();
214             if(response > 0){
215                 lblInfoTab2.setTextFill(Color.BLACK);
216                 lblInfoTab2.setText("Data is successfully saved");
217             }else{
218                 lblInfoTab2.setTextFill(Color.RED);
219                 lblInfoTab2.setText("Unable to save the data!");
220             }
221         } catch (SQLException ex) {
222             Logger.getLogger(JavaFXJDBCApp.class.getName()).
223                 log(Level.SEVERE, null, ex);
224         } catch (ClassNotFoundException ex) {
225             Logger.getLogger(JavaFXJDBCApp.class.getName()).
226                 log(Level.SEVERE, null, ex);
227         }
228     }
229 });
230 // end
231 // tab 3 (krs)
232 Label okMatkul = new Label("0"); Label okMhs = new Label("0");
233 okMatkul.setVisible(false); okMhs.setVisible(false);
234 Button btnSaveTab3 = new Button("Simpan Data KRS");
235 btnSaveTab3.setDisable(true);
236 Label lblInfoTab3 = new Label();
237 Label lblIdMatKulKrs = new Label("Kode Mata Kuliah");
238 Label lblNamaMatKulKrs = new Label("Tidak ditemukan");
239 lblNamaMatKulKrs.setTextFill(Color.RED);
240 Label lblNamaMatKul1 = new Label("Nama Mata Kuliah");
241 TextField txtIdMatKulKrs = new TextField();
242
243
244
245
246 Label lblNimKrs = new Label("NIM");
247 Label lblNamaMhsKrs = new Label("Tidak ditemukan");
248 lblNamaMhsKrs.setTextFill(Color.RED);
249 Label lblNama1 = new Label("Nama Mahasiswa");
250 TextField txtNimKrs = new TextField();

```



```

253 // add listener to text field
254 txtIdMatKulKrs.textProperty().addListener((ObservableValue<?
255     extends String>
256     observable, String oldValue, String newValue) -> {
257     Object objMatkul = GeneralUtils.findMatakuliah(newValue,
258         listMatakuliah);
259     if(objMatkul == null){
260         lblNamaMatKulKrs.setTextFill(Color.RED);
261         lblNamaMatKulKrs.setText("Tidak ditemukan");
262         okMatkul.setText("0");
263     }
264     else{
265         lblNamaMatKulKrs.setTextFill(Color.BLUE);
266         lblNamaMatKulKrs.setText(((TMatakuliah)objMatkul).
267             getNamaMatkul());
268         okMatkul.setText("1");
269     }
270
271     if(okMatkul.getText().equals("0") || okMhs.getText().equals("0"))
272         btnSaveTab3.setDisable(true);
273     else
274         btnSaveTab3.setDisable(false);
275 });
276
277 txtNimKrs.textProperty().addListener((ObservableValue<? extends String>
278     observable, String oldValue, String newValue) -> {
279     Object objMhs = GeneralUtils.findMahasiswa(newValue, listMahasiswa);
280     if(objMhs == null){
281         lblNamaMhsKrs.setTextFill(Color.RED);
282         lblNamaMhsKrs.setText("Tidak ditemukan");
283         okMhs.setText("0");
284     }
285     else{
286         lblNamaMhsKrs.setTextFill(Color.BLUE);
287         lblNamaMhsKrs.setText(((TMahasiswa)objMhs).getNama());
288         okMhs.setText("1");
289     }
290
291     if(okMatkul.getText().equals("0") || okMhs.getText().equals("0"))
292         btnSaveTab3.setDisable(true);
293     else
294         btnSaveTab3.setDisable(false);
295 });
296
297 grdKrs.getColumnConstraints().addAll(cons1, cons2);
298 grdKrs.add(lblIdMatKulKrs, 0, 0); grdKrs.add(txtIdMatKulKrs, 1, 0);
299 grdKrs.add(lblNamaMatKul1, 0, 1); grdKrs.add(lblNamaMatKulKrs, 1, 1);
300 grdKrs.add(lblNimKrs, 0, 2); grdKrs.add(txtNimKrs, 1, 2);
301 grdKrs.add(lblNama1, 0, 3); grdKrs.add(lblNamaMhsKrs, 1, 3);
302 grdKrs.add(btnSaveTab3, 1, 4); grdKrs.add(lblInfoTab3, 1, 5);
303
304 lblNamaMatKulKrs.setStyle("-fx-font-style: italic");
305 lblNamaMhsKrs.setStyle("-fx-font-style: italic");
306 lblInfoTab3.setStyle("-fx-font-style: italic");

```

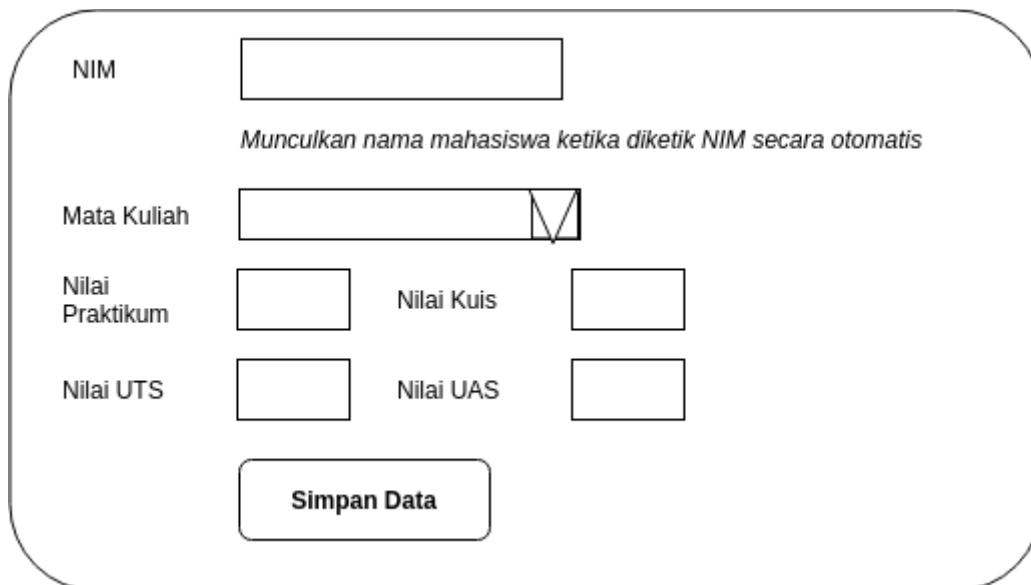
```

308 btnSaveTab3.setOnAction((t) -> {
309     if(txtIdMatKulKrs.getText().trim().isEmpty() ||
310        txtNimKrs.getText().trim().isEmpty()){
311         lblInfoTab3.setTextFill(Color.RED);
312         lblInfoTab3.setText("Silahkan isi dengan lengkap setiap "
313            + "kolom yang ada!");
314     }else{
315         TMkMhs obj = new TMkMhs();
316         obj.setIdMatkul(txtIdMatKulKrs.getText().trim().toUpperCase());
317         obj.setNim(txtNimKrs.getText().trim());
318
319         if(GeneralUtils.checkKRSExist(txtIdMatKulKrs.getText().trim().
320            toUpperCase(),
321            txtNimKrs.getText().trim(), listKrs) != null){
322             lblInfoTab3.setTextFill(Color.RED);
323             lblInfoTab3.setText(String.format("Mahasiswa %s telah "
324                + "mengambil mata kuliah ini!",
325                txtNimKrs.getText().trim()));
326         }else{
327             try {
328                 int response = DBUtils.saveKrs(obj);
329                 loadAllData();
330                 if(response > 0){
331                     lblInfoTab3.setTextFill(Color.BLACK);
332                     lblInfoTab3.setText("Data is successfully saved");
333                 }else{
334                     lblInfoTab3.setTextFill(Color.RED);
335                     lblInfoTab3.setText("Unable to save the data!");
336                 }
337             } catch (SQLException ex) {
338                 Logger.getLogger(JavaFXJDBCApp.class.getName()).
339                     log(Level.SEVERE, null, ex);
340             } catch (ClassNotFoundException ex) {
341                 Logger.getLogger(JavaFXJDBCApp.class.getName()).
342                     log(Level.SEVERE, null, ex);
343             }
344         }
345     }
346 });
347 // end
350 Scene scene = new Scene(tp, 750, 300);
351
352 primaryStage.setTitle("Aplikasi Pengisian KRS Mahasiswa IT-Del");
353 primaryStage.setScene(scene);
354 primaryStage.show();
355 }
356
357 /**
358  * @param args the command line arguments
359  * @throws java.sql.SQLException
360  * @throws java.lang.ClassNotFoundException
361  */
362 public static void main(String[] args) throws SQLException,
363     ClassNotFoundException {
364     loadAllData();
365     launch(args);
366 }
367
368 public static void loadAllData() throws SQLException,
369     ClassNotFoundException{
370     listMahasiswa = DBUtils.getAllMahasiswa();
371     listMatakuliah = DBUtils.getAllMatakuliah();
372     listKrs = DBUtils.getAllKrs();
373 }
374 }

```

TUGAS ANDA

- Buatlah *form* inputan nilai mahasiswa (di dalam sebuah **tab baru**) seperti gambar di bawah ini. Admin akan mengetikkan NIM mahasiswa dan secara otomatis akan keluar nama mahasiswa tersebut. Bila, NIM mahasiswa tersebut tidak terdaftar, maka tombol **Simpan Data** harus di-*disable*. *Field mata kuliah* merupakan *dropdown* yang datanya akan di-*load* berdasarkan NIM mahasiswa (note: setiap mahasiswa **pasti** berbeda-beda mata kuliah yang diambil). Lakukan perhitungan nilai akhir dengan proporsi:



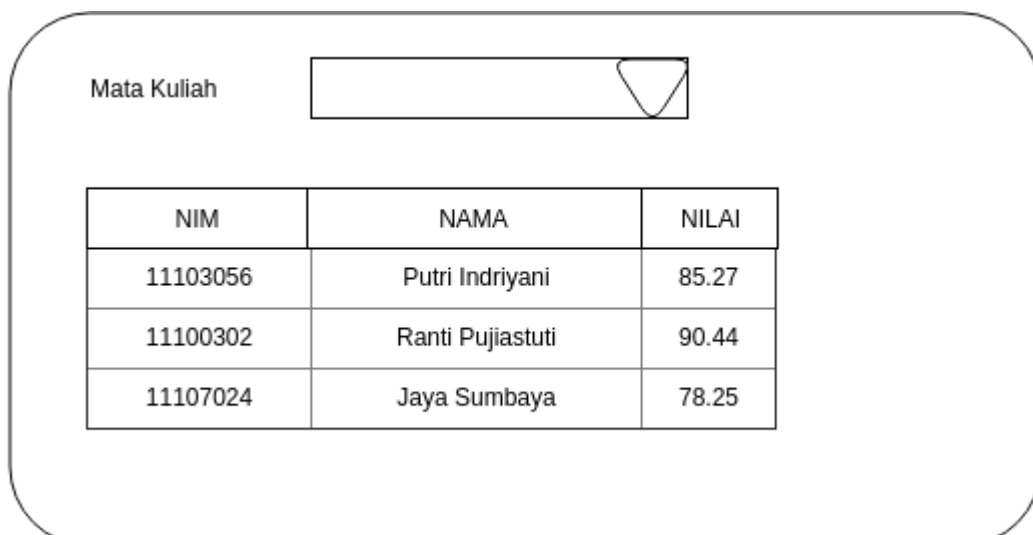
The form is enclosed in a rounded rectangle. It contains the following elements:

- A label "NIM" followed by a text input field.
- A text instruction: "Munculkan nama mahasiswa ketika diketik NIM secara otomatis".
- A label "Mata Kuliah" followed by a dropdown menu showing the letter "M".
- Two input fields for "Nilai Praktikum" and "Nilai Kuis".
- Two input fields for "Nilai UTS" and "Nilai UAS".
- A button labeled "Simpan Data" at the bottom.

Gambar 1: Form input nilai mahasiswa

Nilai Praktikum	10%
Nilai Kuis	20%
Nilai UTS	35%
Nilai UAS	35%

- Lakukan riset kecil terhadap **TableView** di JavaFX, lalu tampilkan seluruh nilai mahasiswa (di sebuah **tab baru**) beserta nilai nya berdasarkan mata kuliah yang dipilih (untuk dilihat) melalui *dropdown*. Lihat contoh layout di bawah ini:



The form is enclosed in a rounded rectangle. It contains the following elements:

- A label "Mata Kuliah" followed by a dropdown menu.
- A table displaying student data:

NIM	NAMA	NILAI
11103056	Putri Indriyani	85.27
11100302	Ranti Pujiastuti	90.44
11107024	Jaya Sumbaya	78.25

Gambar 2: Daftar nilai mahasiswa berdasarkan mata kuliah yang dipilih