UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

Rekayasa Perangkat Lunak

Merancang Antarmuka Pemakai *(User Interface)*

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Fakultas Ilmu Komputer	Teknik Informatika	03D		Anis Cherid, S.E, M.T.I

Abstrak

Debugging adalah proses penelusuran kode-kode program dan evaluasi terhadap nilai berbagai variabel, sehingga berbagai kesalahan logika dalam program dapat ditemukan. Antarmuka pemakai harus dirancang dengan menggunakan prinsip-prinsip terbaik (*best practices)*.

Kompetensi

Mahasiswa juga diharapkan mampu melakukan debugging dengan IDE Eclipse dan melakukan perancangan antarmuka dengan IDE Eclipse dan SceneBuilder.

Merancang Antarmuka Pemakai *(User Interface)*

Dalam bagian ini anda akan belajar konsep *boundary object*, perancangan antarmuka pengguna sederhana dengan menggunakan JavaFX dan menjelaskan *best practices* untuk perancangan antarmuka pemakai aplikasi.

Pada Gambar AA disajikan antarmuka pemakai yang terdapat pada *dashboard* sebuah mobil. Dalam contoh ini, antarmuka yang diperlihatkan adalah antarmuka yang sangat berantakan, sehingga mobil ini sudah tidak bias lagi dikemudikan. Dengan demikian, menjadi cukup jelas bahwa perancangan antarmuka adalah hal yang cukup sulit untuk dilakukan dan masalah ini relevan untuk sistem apa pun yang di dalamnya perangkat lunak mengendalikan perangkat keras.

Gambar AA

Upaya memangun perangkat lunak mengkonsumsi sumber daya dalam jumlah yang cukup signifikan, dan ditemukan bahwa sumber daya yang dihabiskan untuk merancang, melakukan pemrograman dan melakukan pemeliharaan terhadap antarmuka pemakai menghabiskan sekitar setengah dari keseluruhan waktu yang dihabiskan untuk membangun sistem perangkat lunak. Kode program untuk antarmuka pemakai berukuran setengah dari keseluruhan kode program sistem.

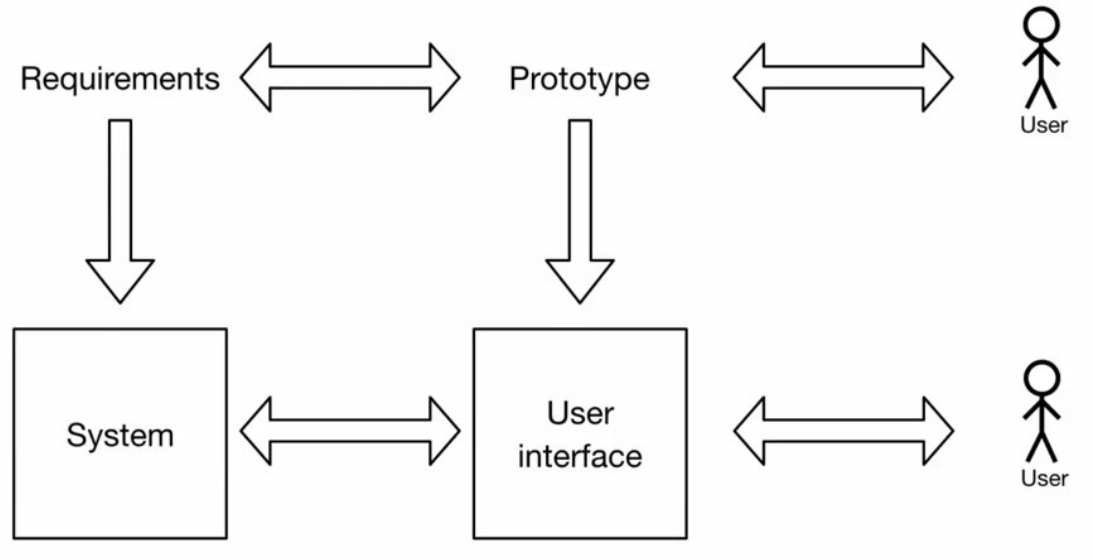
Membangun perangkat lunak yang kompleks, tetapi pada saat yang sama masih dapat digunakan, adalah masalah sulit dan hanya bisa diselesaikan oleh menggunakan teknik dan kebiasaan yang berbeda. Solusinya bukan membangun sepuluh sistem untuk memenuhi sepuluh kebutuhan pemakai, tetapi membangun sebuah sistem yang dapat memenuhi dan sesuai dengan seluruh kebutuhan tersebut.

Kedengarannya saling bertentangan *(paradox)*, namun demikian ternyata untuk mendapatkan antarmuka pemakai yang benar, hanya bisa diperoleh dari sejumlah besar kegagalan. Proses perancangan yang baik adalah proses yang dilakukan secara iteratif. Hasil penelitian yang dilakukan Henry Petroski menyimpulkan informasi bahwa perancangan yang baik berasal dari perancangan yang gagal dibanding dari perancangan yang berhasil. Alasannya adalah karena kegagalan (misalnya kegagalan yang menyebabkan runtuhnya jembatan pada Gambar BB) menyebabkan dilakukannya penyelidikan yang mendalam, mendorong dilakukannya analisis lebih lanjut terhadap masalah yang menyebabkan kegagalan, sehingga dapat ditemukan pendekatan alternatif yang lebih baik. Tanpa kegagalan, semua pihak akan cepat puas dan para perancang akan berhenti berinovasi. Terdapat kutipan yang sangat terkenal dari Petroski yaitu “sukses dalam rekayasa ditentukan oleh berbagai kegagalan yang dialaminya”. Jadi hanya jika kita belajar dari kegagalan kita, maka kita dapat menjadi sukses di masa mendatang. Budaya ini penting untuk diterapkan baik dalam rekayasa perangkat lunak secara umum, maupun dalam proses perancangan antarmuka secara khusus. Keseluruhan konsep ini dapat dinyatakan dengan sebuah ungkapan, ***successful failure*** (kegagalan yang berhasil).

Gambar BB

Dalam Gambar CC disajikan hubungan antara *requirement* (prasyarat perangkat lunak), prototipe, sistem yang dibangun dan antarmuka pemakai. Pembuatan prototipe *(prototyping)* adalah teknik untuk mengidentifikasi kegagalan dalam perancangan antarmuka pemakai,

yang dilakukan sejak dini, yaitu di awal pelaksanaan proyek perangkat lunak. Prototipe yang dihasilkan tergantung kepada prasyarat *(requirement)* dari sistem dan dapat dievaluasi bersama-sama dengan pemakai untuk menemukan gambaran dan bentuk antarmuka pemakai yang paling sesuai dan paling tepat. Sementara prasyarat *(requirement)* pada akhirnya akan menjadi dasar dibangunnya sistem perangkat lunak, prototipe dapat digunakan sebagai dasar untuk membangun antarmuka pemakai. Antarmuka yang dirumuskan dan dibangun berdasarkan prototipe tersebut, akan membungkus dan menjauhkan berbagai rincian rumit yang dimiliki oleh sistem perangkat lunak dari pemakainya dan akan membuat perangkat lunak yang dibangun menjadi dapat digunakan dan yang menyediakan *user experience* yang baik bagi para pemakainya.

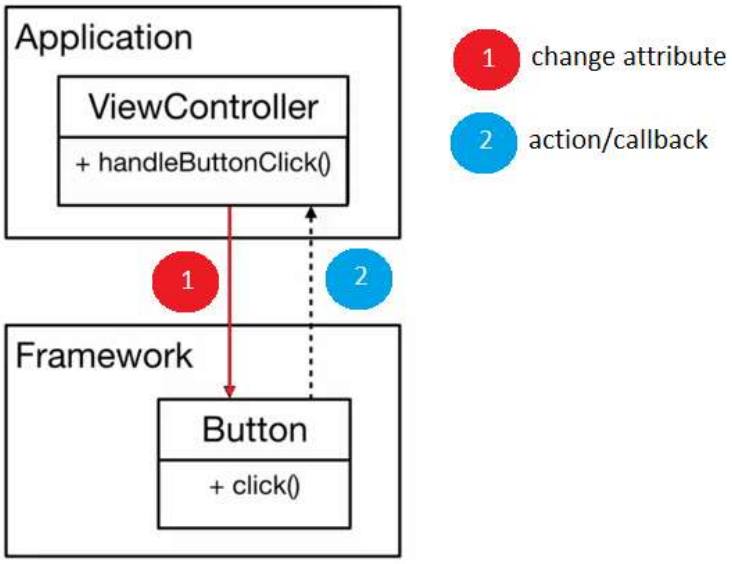


Gambar CC

Untuk mendapatkan *usability* dan *user experience* yang tepat dan sesuai, kita harus mempelajari konsep ***boundary object*** yang menggambarkan interaksi antara pemakai dan sistem dan merupakan bagian dari antarmuka pemakai. Contoh dari *boundary object* adalah label, tombol *(button)*, tabel dan daftar yang ditata dalam hierarki tampilan dengan menggunakan batasan tata letak yang beragam.

Pada Gambar DD, disajikan *class diagram* yang menggambarkan bagaimana aplikasi berinteraksi dengan *library* atau *framework* yang menyediakan beragam *boundary object* yang kaya fitur dan terstandarisasi. Jika seorang pembangun perangkat lunak membutuhkan sebuah tombol untuk diletakkan dalam antarmuka pemakai misalnya, tombol tersebut tidak perlu dibangun dari awal dan dibuat kode programnya, tetapi cukup mengambil dan menggunakan kembali *(reuse) boundary object* berupa tombol dari dalam *library/framework*, dan kemudian melakukan konfigurasi terhadap berbagai atribut dan perilakuknya. Misalnya, setelah pembangun perangkat lunak membuat *instance* atau objek dari tombol, dia dapat

melakukan perubahan atribut yaitu dengan mengubah ukuran, warna, bayangan dan atribut lainnya. Kita juga bisa memanipulasi perilaku dengan menerapkan yang disebut dengan *action* atau disebut juga disebut *callback*. Kemudian berbagai langkah dalam *main loop* akan menunggu hingga pemakai melakukan klik pada tombol dan kemudian memanggil *method* click() yang dimiliki tombol tersebut. Tombol ini kemudian meneruskan panggilan ini *method* handleButtonClick(), yang terdapat dalam ViewController yang terdapat di dalam aplikasi, untuk kemudian memicu dilaksanakannya fungsionalitas yang spesifik dari perangkat lunak. Gambar DD adalah sebuah contoh yang bagus untuk menggambarkan bagaimana cara kerja sebuah *framework*. *Framework* memungkinkan untuk melakukan penyesuaian terhadap Mereka memungkinkan penyesuaian dan pada saat yang sama memungkinkan untuk memanggil aplikasi jika tindakan tertentu harus dilaksanakan, seperti bereaksi terhadap klik yang dilakukan terhadap tombol.

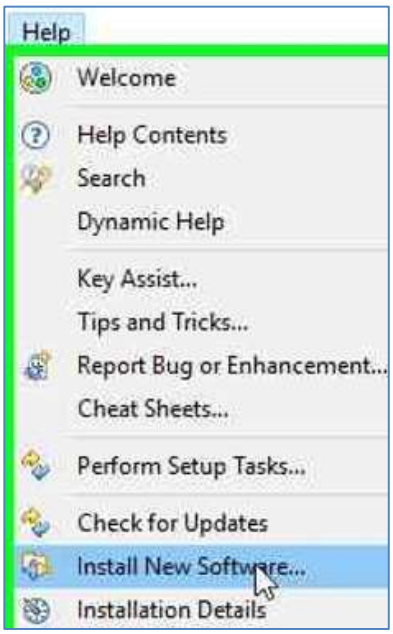
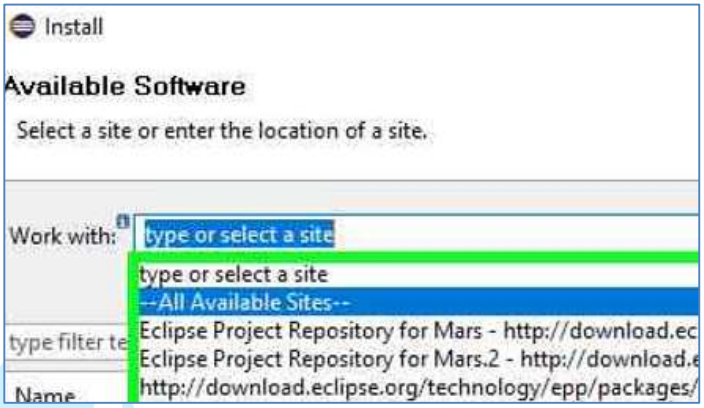


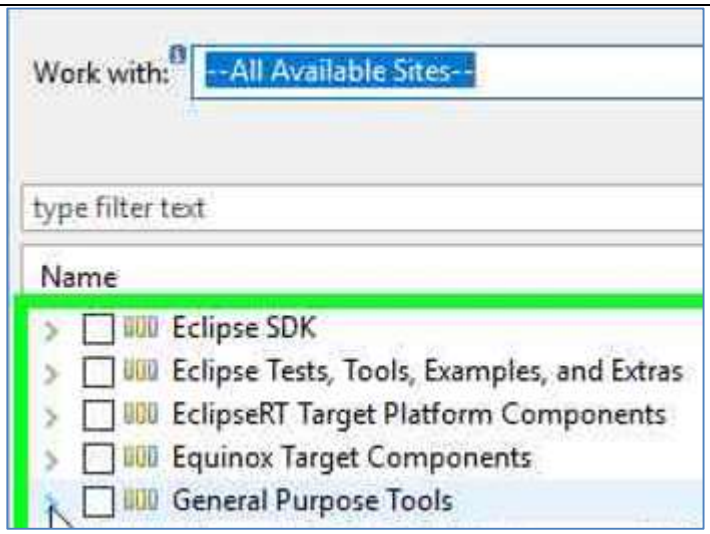
Gambar DD

JavaFX adalah perluasan dari *standard library* bahasa Java yang memiliki berbagai objek UI (*user interface,* antarmuka pemakai) yang kaya fitur. JavaFX merupakan bagian yang integral dari Java 8 dan berbagai fiturnya dapat dipergunakan dengan mudah melalui aplikasi SceneBuilder. Aplikasi ini dapat diintegrasikan ke dalam Eclipse dan NetBeans, sehingga memungkinkan kita untuk membuat antarmuka pemakai dengan menggunakan teknik *drag-and-drop* dan menghasilkan apa yang disebut dengan dokumen FXML yang menjelaskan antarmuka dengan menggunakan model berorientasi objek yang sebanding dengan diagram kelas sebagai latar belakangnya, sehingga kita dapat menjelaskan dalam dokumen ini, seperti apa elemen antarmuka pemakai yang digunakan dan bagaimana konfigurasi dilakukan terhadapnya serta *class* mana saja di dalam aplikasi yang bereaksi terhadap *action* yang dilakukan pemakai pada antarmuka pemakai. JavaFX sangat canggih dan fleksibel serta

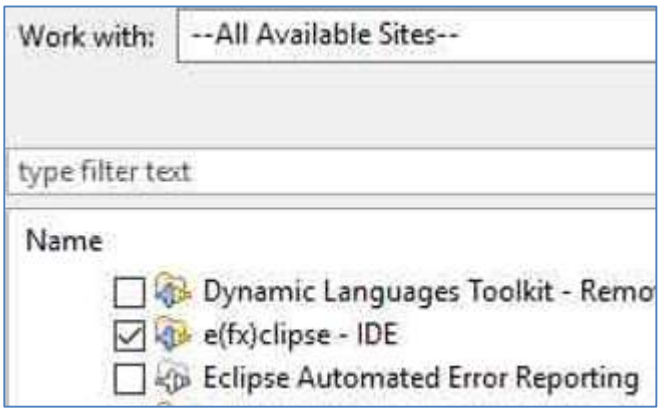
memungkinkan untuk mengendalikan perancangan aplikasi dengan menggunakan sintaks yang mirip dengan CSS yang digunakan dalam halaman web HTML sehingga cukup akrab bagi banyak pembangun perangkat lunak.

Dalam tutorial berikut ini, akan dibahas bagaimana cara menggunakan JavaFX dan SceneBuilder dalam membangun antarmuka pemakai yang sederhana:

1.	Lakukan instalasi JavaFX ke dalam Eclipse (dalam contoh ini, versi Eclipse yang digunakan adalah Eclipse Mars 2 dan JavaFX yang digunakan adalah JavaFX 2)	a. Jalankan Eclipse  b. Klik menu Help >> Install New Software  c. Klik combo box “Work With” dan klik All Available Sites untuk menampilkan daftar software dari seluruh situs yang tersedia dan bisa diakses oleh Eclipse  d. Sesudah daftar software yang tersedia ditampilkan (butuh kurang lebih 5 menit untuk mengunduh daftar ini), lakukan scroll down pada daftar software serta lakukan klik pada “General Purpose Tools”

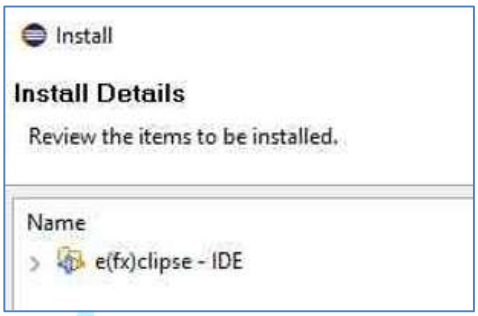


e. Kemudian cari dan klik check box “e(fx)clipse - IDE"

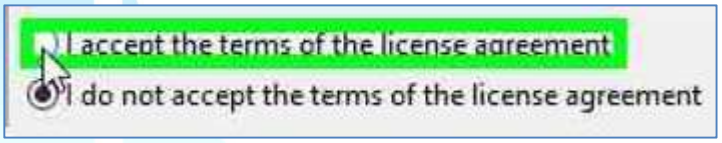


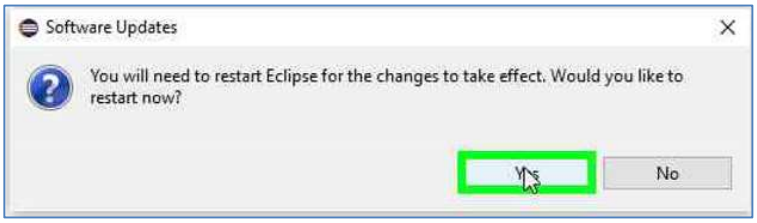
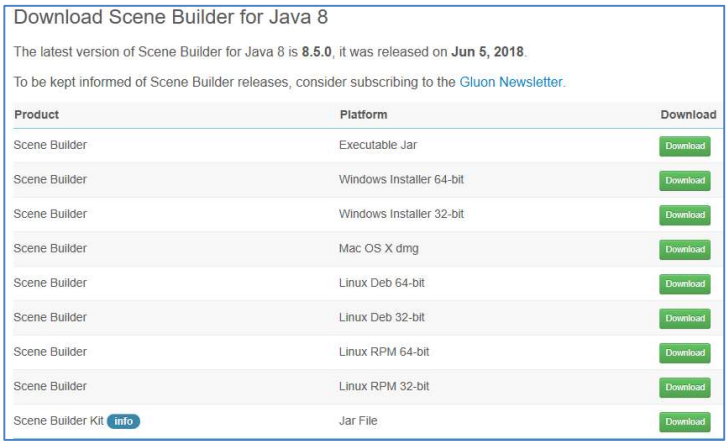
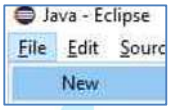
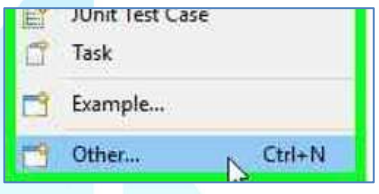
f. Klik tombol Next

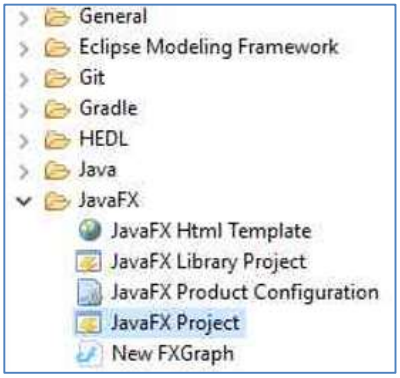
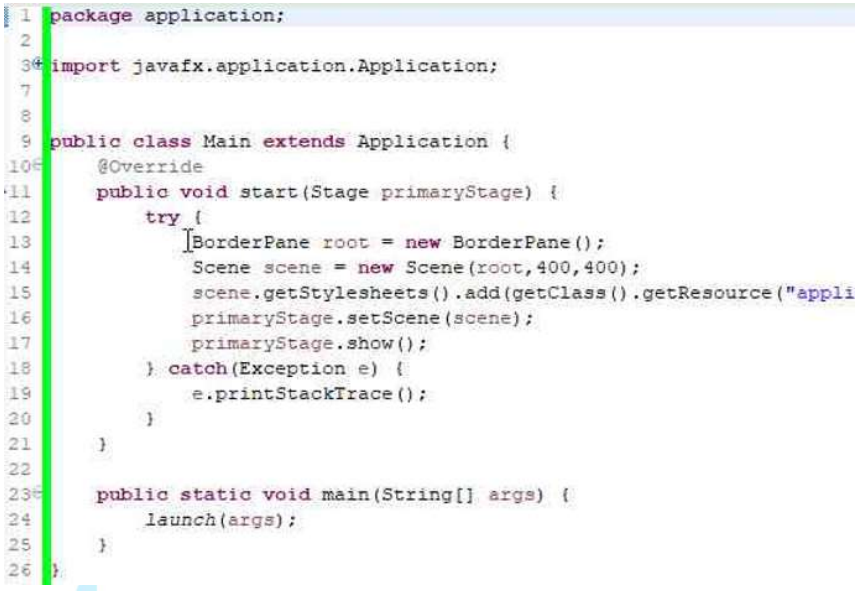
g. Jika sudah muncul halaman “Install Details”, klik pada tombol Next



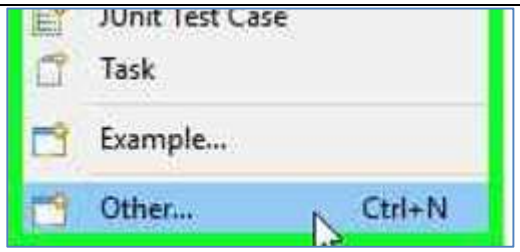
g. Klik radio button “I accept the terms of the license agreement”

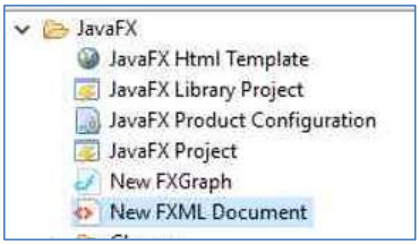


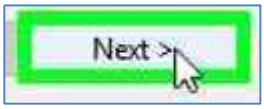
		h. Klik tombol Finish  i. Tunggu hingga proses instalasi selesai (kurang lebih 15 menit) dan ketika diminta untuk melakukan restart Eclipse, klik tombol OK. 
2.	Lakukan instalasi SceneBuilder	a. Gunakan tautan https://gluonhq.com/products/scene-builder/ untuk mengunduh SceneBuilder (dalam contoh ini yang digunakan adalah SceneBuilder Windows Installer 64-bit untuk Java 8, dengan ukuran 55.4 MB untuk komputer belum melakukan instalasi Java 8 atau SceneBuilder Executable Jar untuk Java 8 bagi komputer yang sudah melakukan instalasi Java 8).  b. Lakukan proses instalasi sampai selesai (untuk kasus mengunduh SceneBuilder Windows Installer) atau cukup ingat dalam folder apa file SceneBuilder.jar disimpan (untuk kasus mengunduh SceneBuilder Executable Jar)
3.	Jalankan Eclipse dan buat project JavaFX yang baru.	a. Klik menu File >> New  b. Klik menu Other 

		c. Klik JavaFX >> JavaFX Project
		 
		
		d. Klik tombol Next
		 
		
		e. Tulis nama project menjadi “Tutorial Java FX”
		 
		
		f. Klik tombol Finish
		 
		
		g. Klik-ganda file Main.java untuk membukanya
		 
		1 package application;
		2
		3 import javafx.application.Application;
		7
		8
		9 public class Main extends Application {
		10 @Override
		11 public void start(Stage primaryStage) {
		12 try {
		13 BorderPane root = new BorderPane();
		14 Scene scene = new Scene(root,400,400);
		15 scene.getStylesheets().add(getClass().getResource("appli
		16 primaryStage.setScene(scene);
		17 primaryStage.show();
		18 } catch(Exception e) {
		19 e.printStackTrace();
		20 }
		21 }
		22
		23 public static void main(String[] args) {
		24 launch(args);
		25 }
		26 }

4.	Tambahkan file FXML baru bernama MainScreen .fxml ke dalam project	a. Klik menu File >> New  b. Klik menu Other

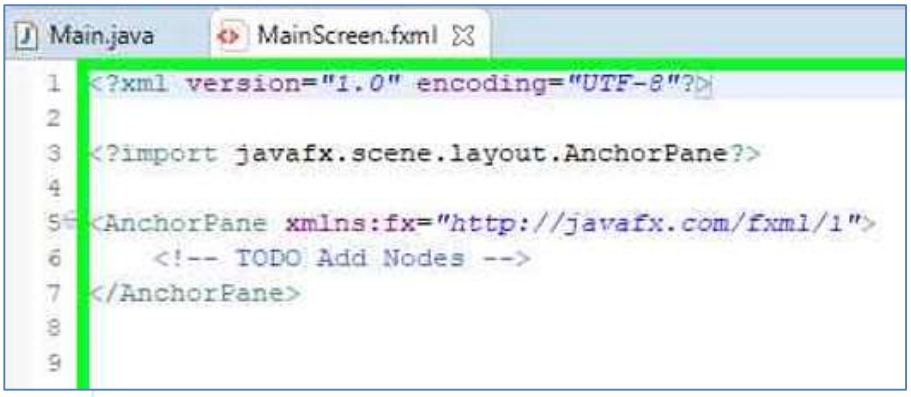
<table>
<tr>
<td></td>
<td></td>
<td>


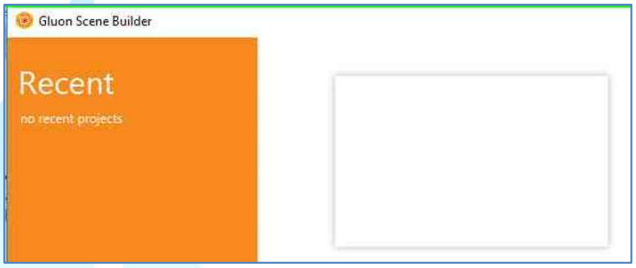
c. Klik JavaFX >> New FXML Document


d. Klik tombol Next


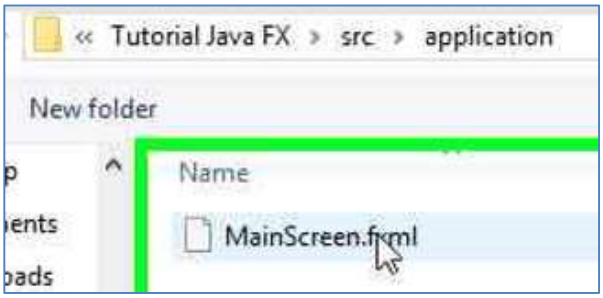
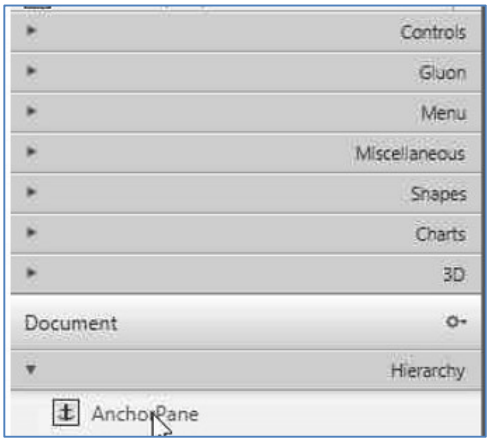
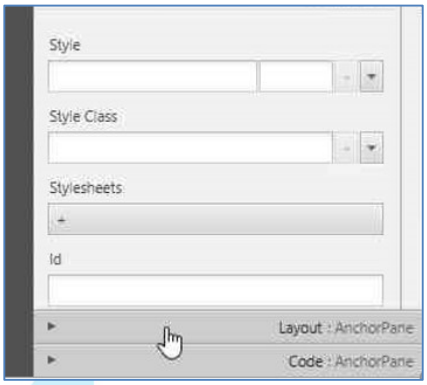
e. Tulis nama file menjadi “MainScreen”


f. Klik tombol Finish

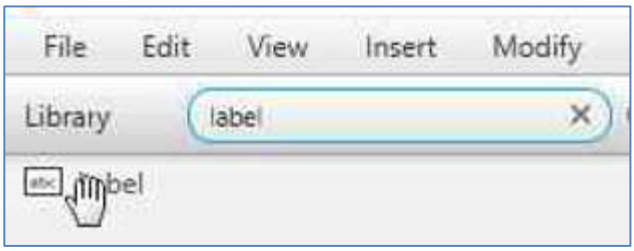

g. File MainScreen.FXML akan langsung ditampilkan


</td>
</tr>
<tr>
<td>5.</td>
<td>Buat antarmuka sederhana menggunakan SceneBuilder dengan cara memperba-</td>
<td>
a. Jalankan SceneBuilder.exe atau SceneBuilder.jar


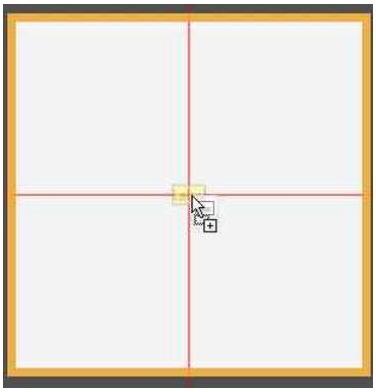
</td>
</tr>
</table>

	harui file MainScreen .fxml.	b. Klik menu Open Project  c. Buka folder project “Tutorial Java FX”, buka folder “src”, buka folder “application” dan klik file “MainScreen.FXML” untuk membukanya dalam SceneBuilder.  d. Klik AnchorPane (kiri bawah layer)  e. Klik drop down menu Layout:AnchorPane (kanan bawah layer)  f. Ubah nilai Pref Width menjadi 300 dan Pref Height menjadi 300 

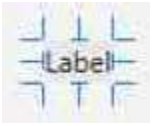
g. Ketikkan “label” pada kotak pencarian library (kiri atas layar) dan tekan ENTER



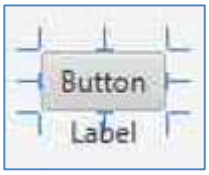
h. Klik dan tarik *control* Label ke tengah-tengah tampilan antarmuka



i. Hasil akhir:



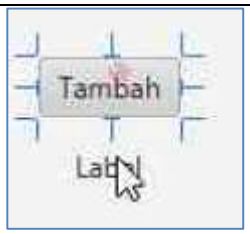
j. Ulangi langkah g dan h untuk *control* Button. Hasil akhirnya:



k. Klik drop down menu “Properties” (kanan atas layer)

l. Ubah property Text dari “Button” menjadi “Tambah”

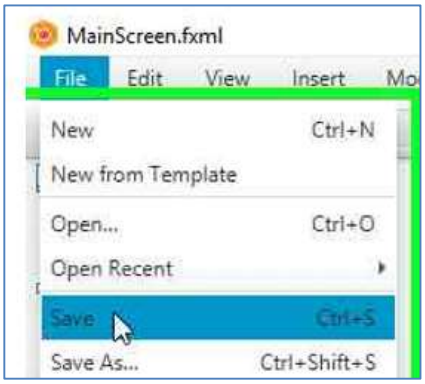
m. Klik lagi *control* Label pada rancangan antarmuka



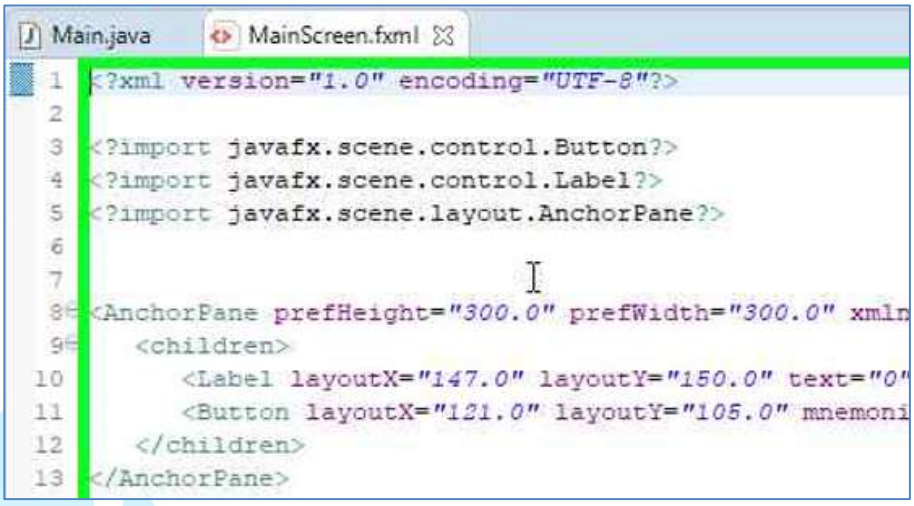
n. Ubah property Text dari “Label” menjadi angka “0”

o. Atur ulang tata letak rancangan antarmuka sehingga tampak seperti gambar berikut ini:

p. Klik menu Save >> File



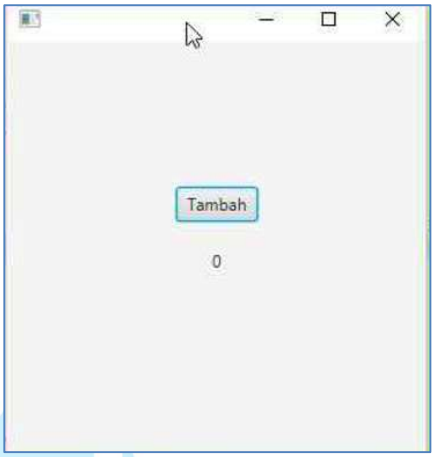
q. Kembali lagi ke Eclipse, file MainScreen.FXML sudah berubah

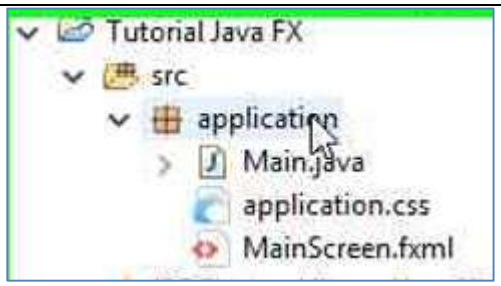
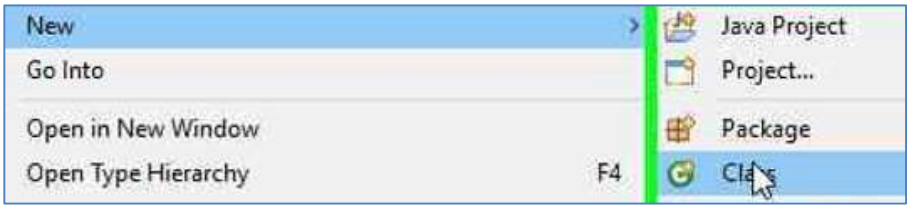
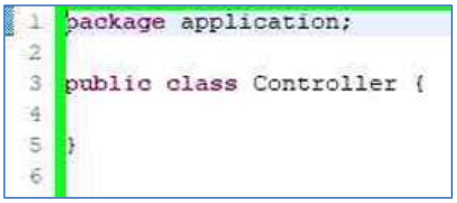
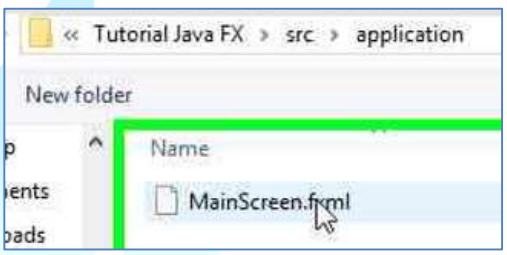


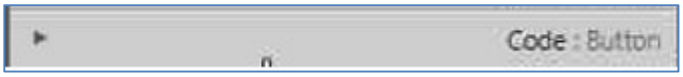
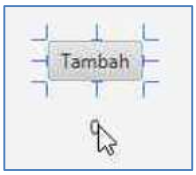
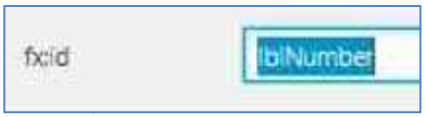
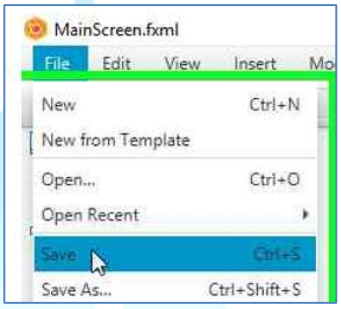
6.	Modifikasi file Main.java sehingga dapat memuat antarmuka yang didefinisikan dalam file MainScreen .fxml	

Main.java | MainScreen.fxml

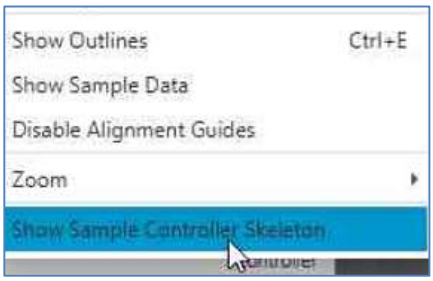
```
1  package application;
2
3  import javafx.application.Application;
4  import javafx.fxml.FXMLLoader;
5  import javafx.stage.Stage;
6  import javafx.scene.Scene;
7
8  public class Main extends Application {
9      @Override
10     public void start(Stage primaryStage) {
11         try {
12             FXMLLoader loader = new FXMLLoader();
13             loader.setLocation(Main.class
14                     .getResource("MainScreen.fxml"));
15             Scene scene = new Scene(loader.load());
16             primaryStage.setScene(scene);
17             primaryStage.show();
18         } catch(Exception e) {
19             e.printStackTrace();
20         }
21     }
22
23     public static void main(String[] args) {
24         launch(args);
25     }
```

7.	Jalankan Main.java	a. Klik tombol Run  b. Maka akan dihasilkan window aplikasi yang sedang dijalankan, yang memiliki *control* label dan button. Namun demikian, belum ada fungsi yang dapat dijalankan pada aplikas ini. 
8.	Buat *class* baru yang bernama	a. Klik kanan pada package “application”

	Controller untuk melakukan *handling* terhadap berbagai *action* yang dilakukan pada *button*	 b. Klik menu New >> Class  c. Beri nama “Controller”  d. Klik tombol “Finish”  e. Class yang baru akan ditampilkan sebagai berikut: 
9.	Lakukan perubahan terhadap file MainScreen .fxml melalui SceneBuilder sehingga menyertakan berbagai atribut yang memungkin-kan ditanganinya berbagai	a. Jalankan SceneBuilder.jar b. Klik menu “Open Project”  c. Buka folder project “Tutorial Java FX”, buka folder “src”, buka folder “application” dan klik file “MainScreen.fxml” untuk membukanya dalam SceneBuilder. 

	action pada antarmuka. Selain itu, gunakan SceneBuilder untuk menghasilkan kerangka kode program bagi class Controller.	d. Klik menu drop down “Controller” (kiri bawah layer)  e. Tulis nama *controller class* menjadi nama package diikuti dengan nama class controller (“application.Controller”)  f. Klik button Tambah yang terdapat pada rancangan antarmuka  g. Klik menu dropdown Code (kanan bawah layar)  h. Ganti fx:id menjadi “btnTambah”  i. Scroll ke bawah dan ganti OnMouseClick menjadi “handleBtnTambahClick”  j. Klik label angka 0 yang terdapat pada rancangan antarmuka  k. Klik menu dropdown Code (kanan bawah layar), scroll ke atas dan ganti field fx:id menjadi “lblNumber”  l. Klik menu Save >> File 

m. Klik menu View >> Show Sample Controller Skeleton



n. Perhatikan kerangka kode program yang dihasilkan untuk *class* Controller

Sample Skeleton for 'MainScreen.fxml' Controller Cl

```
package application;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;

public class Controller {

   @FXML
   private Label lblNumber;

   @FXML
   private Button btnTambah;

   @FXML
   void handleBtnTambahClick(MouseEvent event) {

   }

}
```

o. Klik tombol Copy dan kemudian tutup window Sample Skeleton

p. Kembali ke Eclipse, sorot semua kode program pada class Controller dengan tombol Ctrl-A dan kemudian *paste* dengan kode program dari Sample Skeleton. Perbaiki kode program sehingga menjadi tampak seperti gambar di bawah ini:

Main.java | MainScreen.fxml | Controller.java

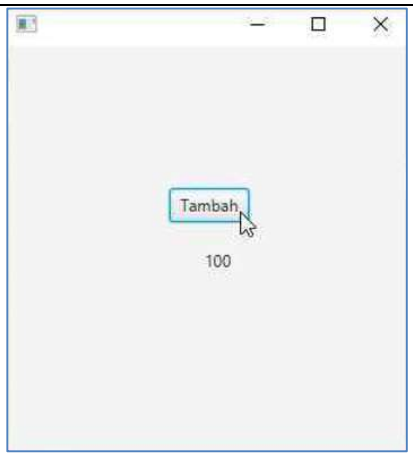
```
1  package application;
2
3  import javafx.fxml.FXML;
4  import javafx.scene.control.Button;
5  import javafx.scene.control.Label;
6  import javafx.scene.input.MouseEvent;
7
8  public class Controller {
9
10     @FXML
11     private Label lblNumber;
12
13     @FXML
14     private Button btnTambah;
15
16     @FXML
17     void handleBtnTambahClick(MouseEvent event) {
18         Integer number = new Integer(lblNumber
19                 .getText());
20         number++;
21         lblNumber.setText(number.toString());
22     }
23
24 }
```

q. Perhatikan juga bagaimana file MainScreen.fxml telah berubah sehingga memiliki atribut “fx:id” dan atribut “onMouseClicked” .

Main.java | MainScreen.fxml | Controller.java

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.control.Button?>
4  <?import javafx.scene.control.Label?>
5  <?import javafx.scene.layout.AnchorPane?>
6
7  <AnchorPane prefHeight="300.0" prefWidth="300.0" xmlns="http:/
8     <children>
9        <Label fx:id="lblNumber" layoutX="147.0" layoutY="150.0"
10       <Button fx:id="btnTambah" layoutX="121.0" layoutY="105.(
11    </children>
12 </AnchorPane>
```

r. Jalankan aplikasi, sehingga sekarang jika pemakai melakukan klik pada tombol “Tambah”, maka angka pada label akan bertambah satu.

Untuk merangkum apa yang sudah dijelaskan pada bagian ini, kita telah mempelajari bagaimana cara membuat antarmuka pemakai dengan menggunakan JavaFX dan SceneBuilder. Kita juga telah membahas cara menghubungkan elemen UI dengan kode program dan bagaimana kita dapat mengkonfigurasinya untuk kebutuhan kita.

Ada berbagai ***best practice*** terkait dengan perancangan antarmuka pemakai, yaitu:

1. **Pusatkan perhatian pada upaya memaksimalkan *usability* dan *user experience***. Perancangan antarmuka pemakai atau rekayasa antarmuka pemakai adalah perancangan antarmuka pemakai untuk berbagai mesin dan perangkat lunak seperti komputer, peralatan rumah tangga, perangkat mobile dan perangkat elektronik lainnya, yang memusatkan perhatiannya pada memaksimalkan *usability* (kemudahan penggunaan) dan *user experience* (pengalaman pemakai). Tujuannya adalah untuk membuat interaksi pemakai sesederhana dan seefisien mungkin dalam hal mencapai tujuan pemakai, yang kemudian juga disebut *user centered design* (perancangan yang menjadikan pemakai sebagai tolok ukur). Perancangan antarmuka pemakai yang baik memfasilitasi penyelesaian tugas yang ada di depan mata, tanpa terlalu banyak merepotkan pemakai dan tanpa perlu memaksa pemakai melakukan hal yang tidak perlu.
2. **Terapkan panduan dan pedoman *(guideline)* yang pada umumnya dipergunakan dalam melakukan perancangan antarmuka pemakai, tetapi sesuaikan juga dengan kebutuhan yang spesifik dari masalah yang dihadapi**. Perancangan antarmuka dilakukan dalam berbagai proyek, mulai dari sistem komputer, mobil, pesawat komersial hingga berbagai sistem lainnya. Semua ini proyek melibatkan banyak interaksi manusia yang pada dasarnya berjenis sama, tetapi juga membutuhkan keterampilan dan pengetahuan yang sangat spesifik. Akibatnya para perancangan antarmuka cenderung mengkhususkan keahlian mereka pada proyek dengan jenis tertentu dan memiliki keterampilan yang secara khusus berkaitan dengan berbagai keahlian, seperti

perancangan perangkat lunak, penelitian pengguna, perancangan web atau perancangan industri.

3. **Lakukan analisis terhadap pera pemakai dan aktivitas yang mereka lakukan.** Analisis pemakai dan tugas-tugasnya (*user and task analysis)* memungkinkan pembangun perangkat lunak untuk memahami masalah dan alur kerja yang spesifik dari berbagai aktivitas yang dilakukan pemakai perangkat lunak, sebelum mulai membangun prototipe yang pertama.
4. **Gunakan teknik *prototyping* untuk memperbaiki perancangan antarmuka secara iterative.** Prototyping memungkinkan untuk menerapkan prinsip "belajar dari kegagalan" dalam melakukan perancangan antarmuka pemakai sejak masa awal proyek dan yang secara iteratif meningkatkan dan memperbaiki antarmuka pemakai.
5. **Lakukan inspeksi *usability* dan *user experience* dengan melakukan uji coba pada pemakai yang sebenarnya.** Inspeksi *usability* memungkinkan untuk memahami masalah yang dihadapi oleh pemakai ketika menggunakan sistem dan antarmuka pemakainya, dan untuk menemukan pendekatan alternatif yang dapat meningkatkan *user experience*.
6. **Gunakan pedoman atau panduan antarmuka untuk manusia *(human interface guidelines)*.** Panduan dan pedoman perancangan antarmuka untuk manusia menjelaskan *best practice* untuk isu-isu *usability* yang bersifat umum dan juga menyediakan jawaban atas berbagai masalah yang secara khusus berlaku pada *platform* tertentu.

Pada bagian ini kita telah mempelajari konsep *boundary object* dan kita telah belajar untuk mengimplementasikan perancangan antarmuka pemakai dengan menggunakan JavaFX dan akhirnya kita juga telah membahas tentang *best practice* dalam melakukan perancangan antarmuka pemakai.

Daftar Pustaka

Bernd Bruegge, Stephen Krusche, Andreas Seitz, Jan Knobloch. **Software Engineering Essentials**. Free Online Course from Technical University of Munich trough edX (https://courses.edx.org/courses/course-v1:TUMx+EASEx+2T2017/course/)