Tallinna Ülikool
Digitehnoloogiate instituut

# JavaFX rakenduste kujundamine CSS abil

Seminaritöö

Autor: Hendrik Spiegelberg
Juhendaja: Jaagup Kippar

Autor: ..... „ ..... „2016
Juhendaja: ..... „ ..... „2016
Instituudi direktor: ..... „ ..... „2016

Tallinn 2016

## Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

..... .....

(kuupäev) (autor)

# Sisukord

# Sissejuhatus

Antud seminaritöö eesmärgiks on tutvustada *JavaFX* ja selle abil loodud rakenduste kujundamise võimalusi kasutades selleks *CSS*-i. Loon väikese ülevaate *JavaFX*-ist ja *CSS*-ist ning sellest, kuidas *CSS* koos *JavaFX*-iga toimib. Näidatakse, kuidas toimub vajaliku tarkvara paigaldamine: *JDK(Java Development Kit), Java* integreeritud programmeerimiskeskond *Eclipse* ning selle lisamoodul *e(fx)clipse* ja *JavaFX* kasutajaliidese loomiseks mõeldud rakendus *SceneBuilder*.

Seminaritöö käigus luuakse ka *JavaFX* kasutades kaks rakendust, millest esimene on lihtne näiterakendus. Teiseks rakenduseks on keerulisem rakendus, mis kuvab informatsiooni ja laseb seda ka ära kustutada. Teist rakendust kasutatakse ka kujunduse muutmise protsessi kirjeldamiseks, kus rakenduse erinevate elementide kujundusi muudetakse, kasutades selleks *CSS*.

Töö vajalikkus on põhjendatud *Java* ja *CSS* populaarsusega, samuti pole samal teemal eestikeelset materjali saadaval. Seda seminaritööd saab kasutada ka juhendina selleks, et näha kuidas *JavaFX* programme luua ja neid *CSS* kasutades kujundada.

# 1. JavaFX ja CSS tutvustus

## 1.1. JavaFX

JavaFX on graafika ja meedia pakettide kogum, mis võimaldab arendajatel disainida, luua ja juurutada rikka kliendi rakendusi, mis toimivad järjekindlalt mitmesugustel platvormidel. JavaFX on kirjutatud Java API-na (*Application programming interface*), JavaFX rakenduse kood võib viidata API-dele ükskõik millisest Java teegist. Näiteks võivad JavaFX rakendused kasutada Java API teeke, et pääseda ligi süsteemi ressurssidele ja ühendada serveri põhistele vahevara rakendustega. JavaFX 2.2 ja hilisemad redaktsioonid on täielikult integreeritud Java SE 7 Runtime Environment (JRE) ja ka Java Development Kit (JDK). Kuna JDK on saadaval kõikidele peamistele arvuti platvormidele (Windows, Mac OS X ja Linux), siis JavaFX rakendused, mis on kompileeritud JDK 7 ja hilisemale, töötavad ka kõikidel peamistel arvuti platvormidel. (Pawlan & Castillo, 2013)

## 1.2. CSS

*Cascading Style Sheets* (CSS) on keel, mis on loodud kirjeldamaks dokumentide välimust, mis on kirjutatud näiteks HTML märgistuskeeles. CSS-i kasutades saab muuta tekstivärvi, teksti fonti, paragrahvide vahel olevat vahet, veergude suurusi ja nende paigutust muuta, milliseid taustavärve või pilte on kasutatud ja palju muid erinevaid visuaalseid efekte. Üheks suurimaks CSS-i eeliseks on, et sama CSS saab kasutada mitmetel lehekülgedel, mis tähendab, et terve veebilehe stiili saab muuta ilma, et peaks muutma igat lehekülge eraldi. Tavaliselt kasutatakse CSS veebilehtede kujundamiseks ja kasutades seda koos HTML ja JavaScript-iga on CSS väga võimekas tööriist. (Pouncey & York, 2011)

# 2. JavaFX ja CSS

JavaFX üheks võimsamaks omaduseks on CSS kasutamisvõimalus rakenduse kasutajaliidese välimuse muutmiseks. JavaFX rakenduse kujundamiseks CSS kasutades, on võimalik muuta rakenduse välimust ja olemust ilma, et peaks Java rakenduse koodi muutma. CSS põhiliselt lahutab rakenduse visuaalse poole rakenduse loogika poolest. (Lowe, 2015)

CSS stiilid rakendatakse JavaFX *scene graph*-is sõlmedele sarnaselt nagu CSS stiilid rakendatakse elementidele HTML DOM-is. Esimesena rakendatakse stiilid vanematele ja siis edasi järglastele. Kood on kirjutatud niimoodi, et käiakse läbi ainult need harud *scene graph*-is, kus on vaja CSS uuesti rakendada. Sõlmele lisatakse stiil siis, kui ta lisatakse *scene graph*-i. Stiile rakendatakse uuesti, kui on muutunud *node pseudo-class* olek, stiili klass, id, *inline* stiil või *parent.* (Oracle, 2013)

CSS stiilid rakendatakse asünkroonselt, mis tähendab, et CSS stiilid laetakse ära, väärtused teisaldatakse ja määratakse peale seda, kui *scene graph*-i objekt on loodud ja lisatud *scene graph*-i aga enne kui *scene graph* on esmalt *laid out* ja *painted.* Lisaks, kui stiilid, mis on lisatud mingile objektile muutuvad, siis uue stiili väärtusi ei rakendata koheselt vaid rakendatakse pärast seda kui objekti olek on muutunud aga enne seda kui järgmine stseen on *painted.* Kuna CSS stiile rakendatakse asünkroonselt, siis võib juhtuda, et väärtused on määratud algselt programmi poolt ja hiljem kirjutatakse need CSS poolt üle. (Oracle, 2013)

JavaFX CSS ei toeta CSS-i asetamise omadusi, näiteks *float, position, overflow* ja *width.* Mõnedel JavaFX *scene graph* objektidel on siiski toetatud CSS *padding* ja *margins* omadused. Kõik teised asetamisega seotud aspektid on JavaFX programmikoodi poolt määratud. Lisaks ei ole tuge ka HTML põhistele CSS elementidele, näiteks *Table,* sest JavaFX pole samaväärset konstruktorit. (Oracle, 2013)

# 3. JavaFX paigaldamine

JavaFX kasutamiseks on vaja veenduda, et arvutis oleks olemas Java JDK 8+ ja Eclipse, koos e(fx)clipse laiendusega. JDK on vajalik Java ja ka JavaFX rakenduste programmeerimiseks. Eclipse on Java IDE (*integrated development environment)*, mis teeb rakenduste loomise lihtsamaks, pakkudes projekti haldamiseks erinevaid tööriistu ja koodi kirjutamisel erinevaid soovitusi andes. e(fx)clipse laiendus laseb Eclipse kasutades JavaFX rakendusi luua ilma, et peaks loodud projekte ise koguaeg konfigureerima JavaFX jaoks.

## 3.1. JDK

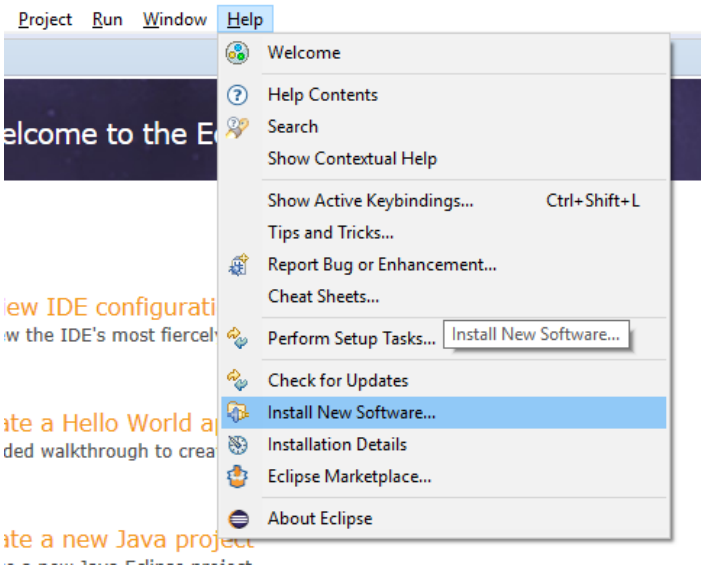
JDK saab alla laadida lehelt:
http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
Laadides alla vastava JDK tuleb paigaldamiseks järgida juhiseid.

## 3.2. Eclipse

Eclipse saab alla laadida lehelt: https://www.eclipse.org/downloads/
Kui Eclipse on alla laetud, siis tuleb paigaldamiseks järgida juhiseid.

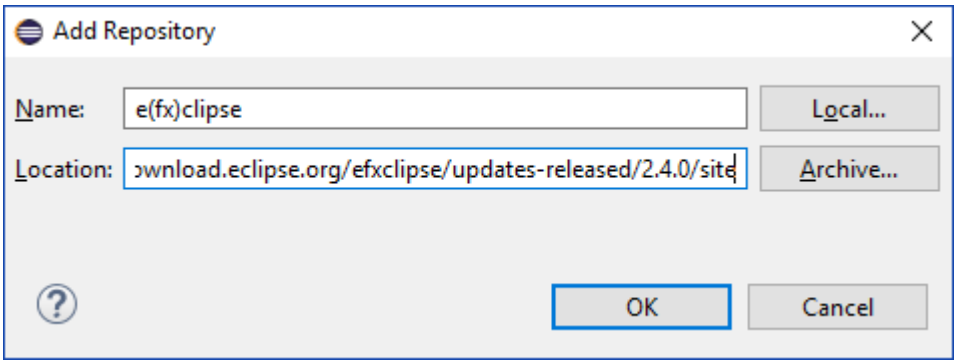
## 3.3. e(fx)clipse

e(fx)clipse paigaldamiseks tuleb käivitada Eclipse ja valida ülevalt menüü ribalt: *Help->Install New Software…*




*Joonis 1Eclipse Install new software...*

Järgmisena tuleb valida *Add…*, *Name:* e(fx)clipse ja *Location:* http://download.eclipse.org/efxclipse/updates-released/2.4.0/site ja valida OK.




*Joonis 2Uue asukoha lisamine*

Peale seda lisandub kaks uut varianti: e(fx)clipse – install ja e(fx)clipse – single components ja valida tuleks e(fx)clipse – install tehes linnuke selle variandi ette ja valides alt *Next* >

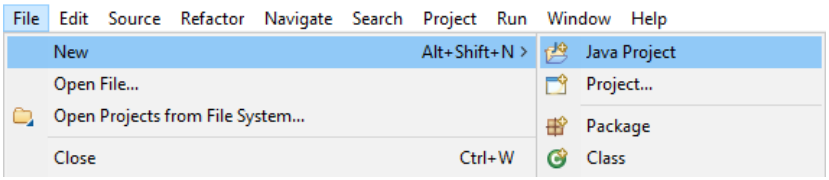

*Joonis 3e(fx)clipse – install*

Peale laadimist antakse ülevaade, mille kinnitamiseks tuleb valida *Next* > ja järgmisel lehel nõustuda tingimustega ja valida *Finish.* Peale e(fx)eclipse allalaadimist tuleb teade, et Eclipse tuleb taaskäivitada.

# 4. JavaFX näiterakenduse loomine Eclipse-is

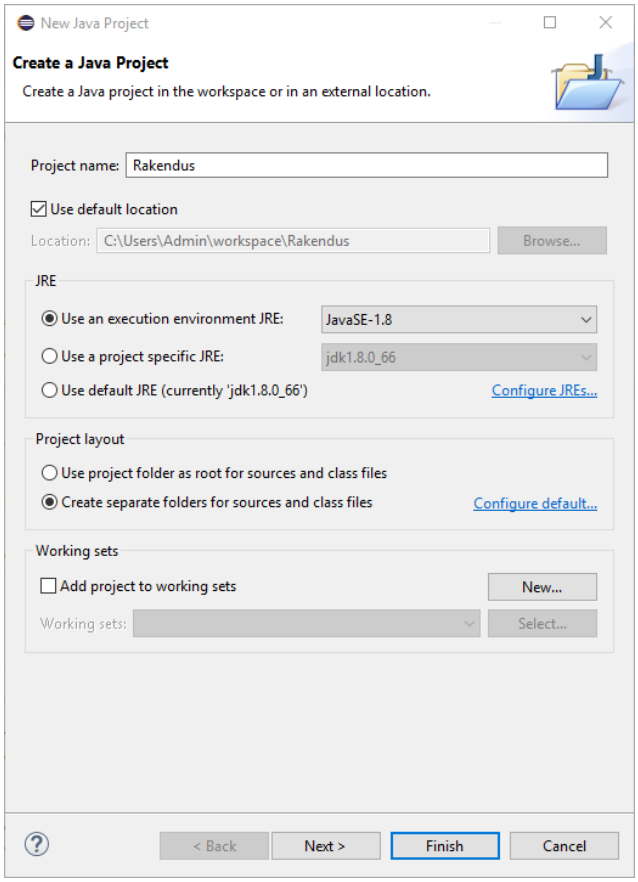
## 4.1. Projekti ülesseadmine

JavaFX projekti loomiseks tuleb kõigepealt käivitada Eclipse ja valida *File -> New -> Java Project.*



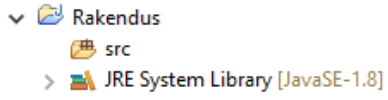

*Joonis 4 Java projekti loomine*

Järgmisena tuleb sisestada projekti nimi ja valida alt *Finish.*



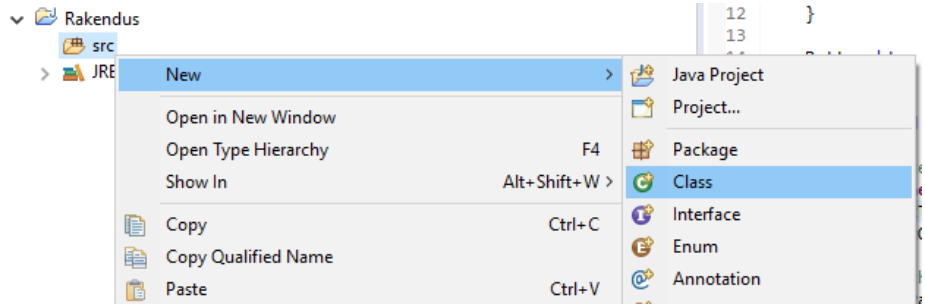

*Joonis 5 Java projekti nime valimine ja kinnitamine*

Sellepeale luuakse tühi projekt, kus saab oma rakendust arendama hakata.



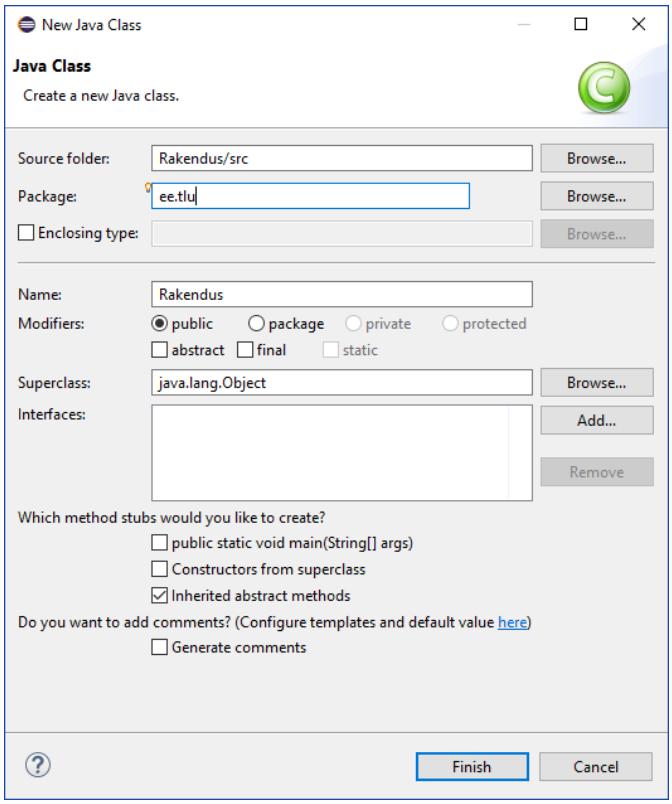

*Joonis 6 Java tühi projekt*

Järgmiseks tuleb luua uus Klass (*Class)*, kuhu saab hakata programmi koodi kirjutama. Selleks tuleb teha parem klõps *src* peal ja sealt valida *New -> Class.*




*Joonis 7 Uue klassi loomine*

Kuvatavas aknas tuleb määrata klassile nimi ja soovitatav oleks ära vahetada ka *package* ja valida *Finish*, mille peale luuakse ja avatakse loodud Klass.




*Joonis 8 Klassi nime ja package määramine*

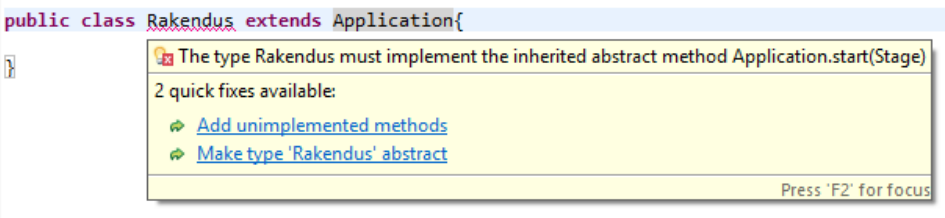
## 4.2. Näiterakenduse loomine ja JavaFX sisseehitatud *Stylesheet* kasutamine

Loodav näiterakendus koosneb aknast ja ühest nupust. Nupu vajutamine muudab kasutatavat sisseehitatud *stylesheet*-i, vaikimisi kasutatav *stylesheet* on *modena.css* ja vajutades nuppu võetakse kasutusele *caspian.css*.

Pärast projekti ja klassi loomist avaneb loodud klass, lisame *package* alla vajalikud *import*-id.

```
package ee.tlu;

import javafx.application.*;
import javafx.stage.*;
import javafx.scene.*;
import javafx.scene.layout.*;
import javafx.scene.control.*;

public class Rakendus {

}
```

Lisame *public class Rakendus* järele *extends Application*, mille peale tuleb *Rakendus* alla punane joon, liikudes hiirega *Rakendus* peale kuvatakse kaks valikut ja valida tuleb *Add unimplemented methods*, mille peale luuakse automaatselt *Override* meetod.




*Joonis 9 Add unimplemented methods*

Loome ka *main* meetodi, et programmi käivitada ja lisamine sinna sisse *launch(args);*

```
public class Rakendus extends Application{

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

	@Override
	public void start(Stage primaryStage) throws Exception {
		// TODO Auto-generated method stub

	}

}
```

Deklareerime *main* meetodi järel uue nupu (*Button)* ja loome uue nupu *Override* meetodi sisse, lisame nupul kuvatava teksti ja määrame nupule tegevuse. *Override* meetodi seest võib ära kustuda *TODO* rea.

```
Button nupp;

	@Override
	public void start(Stage primaryStage) throws Exception {
		//Loome nupu, lisame teksti ja tegevuse
		nupp = new Button();
		nupp.setText("Stylesheet on Modena");
		nupp.setOnAction(e -> buttonClick());
	}
```

Nüüd tuleks ka luua *buttonClick()* meetod, et nupu vajutamine midagi ka teeks. Selleks, et meetod automaatselt luua tuleb liikuda hiirega *buttonClick()* peale, mis on peale *nupp.setOnAction* sulgude sees ja valida *Create method 'buttonClick()'.*

*Override* meetodis peale nupu loomist, teeme uue *BorderPane* ja lisame oma loodud nupu selle keskele, ning lisame loodud *BorderPane* uude stseeni (*Scene).*

```
//Loome uue BorderPane ja lisame oma nupu keskele
BorderPane pane = new BorderPane();
pane.setCenter(nupp);
//Lisame loodud BorderPane uude stseeni
Scene scene = new Scene(pane, 640, 480);
```

Peale seda valime stseeni mida kuvada, lisame sellele pealkirja ja lõpetuseks kuvame stseeni välja.

```
//Valime kuvatava stseeni, lisame pealkirja ja kuvame stseeni
primaryStage.setScene(stseen);
primaryStage.setTitle("Näiterakendus");
primaryStage.show();
```

Järgmiseks liigume edasi loodud *buttonClick()* meetodi juurde. Kustutame sealt seest ära *TODO* rea ja *return null;* rea. Määrame ka ära, et meetod ei tagastaks objekti (*Object*) vaid oleks *void*. Lisame meetodi sisse ühe *if* ja ühe *else* lause.

```
private void buttonClick() {
		if()
		{

		}
		else
		{

		}
	}
```

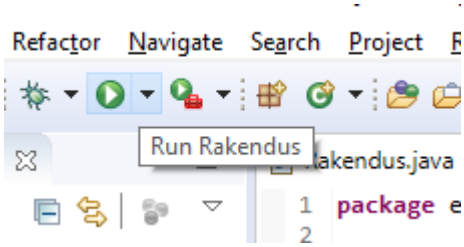
Lisame *if* lausele tingimuse, et kui nupu tekst on võrdne „Stylesheet on Modena“, siis muudame nupul teksti ja vahetame ära ka *stylesheet*-i.

```
if(nupp.getText() == "StyleSheet on Modena")
	{
		nupp.setText("Stylesheet on Caspian");
		Application.setUserAgentStylesheet(STYLESHEET_CASPIAN);
	}
```

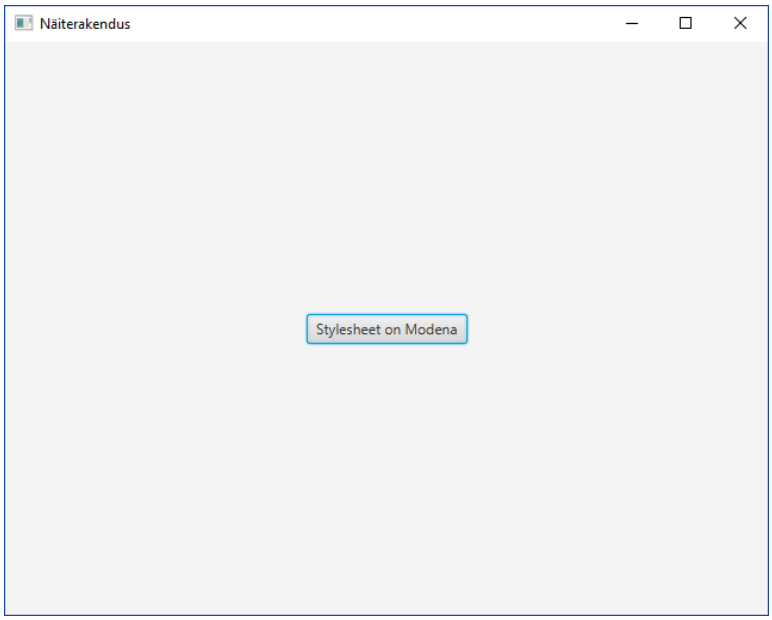
Teeme sama ka *else* lausega.

```
else
	{
		nupp.setText("Stylesheet on Modena");
		Application.setUserAgentStylesheet(STYLESHEET_MODENA);
	}
```

Peale seda on rakendus valmis ja saab käivitada ülevalt menüüst valides *Run Rakendus*, mille peale käivitatakse rakendus ja nupule vajutades saab vahetada *styleheet*-i edasi ja tagasi.




*Joonis 10 Rakenduse käivitamine*



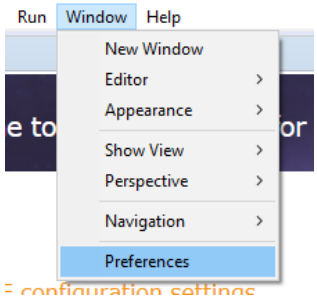

*Joonis 11 Valmis rakendus*

# 5. *Scene Builder* abil keerulisema JavaFX rakenduse loomine

## 5.1. *Scene Builder rakenduse paigaldamine ja Eclipse konfigureerimine*

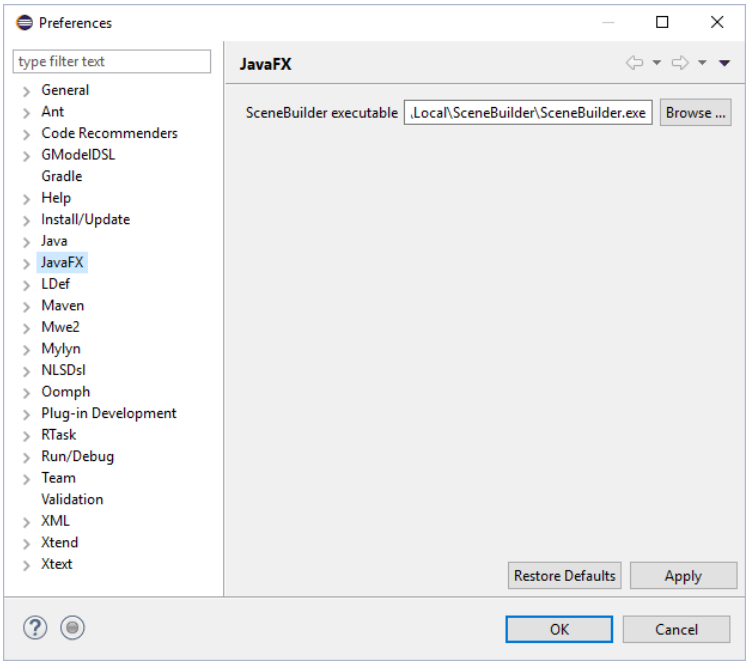
Et suurema stseeni tegemine oleks lihtsam ja kiirem, siis võtame kasutusele programmi *Scene Builder*. *Scene Builder*-i saab alla laadida lehelt: http://gluonhq.com/labs/scene-builder/

Kui *Scene Builder* on alla laetud, siis tuleb paigaldamiseks järgida juhiseid. Peale *Scene Builder*-i paigaldamist tuleb määrata *Eclipse*-s *Scene Builder*-i asukoht. Selleks tuleb *Eclipse*-s valida *Window -> Preferences.*




*Joonis 12 Eclipse preferences*

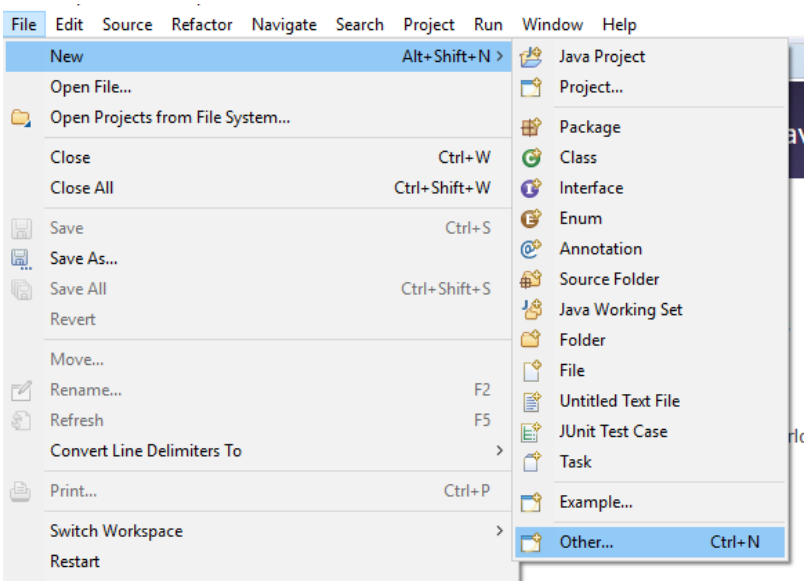
Järgmiseks tuleb kõrvalmenüüst valida *JavaFX* ja seal määrata ära *SceneBuilder executable* asukoht *Browse…* nuppu vajutades. *SceneBuilder.exe* peaks asuma kaustas *C:\Users*Kasutajanimi*\AppData\Local\SceneBuilder*. Peale asukoha määramist vajutada *Apply* ja siis *OK.*




*Joonis 13 Scene Builder asukoha määramine*

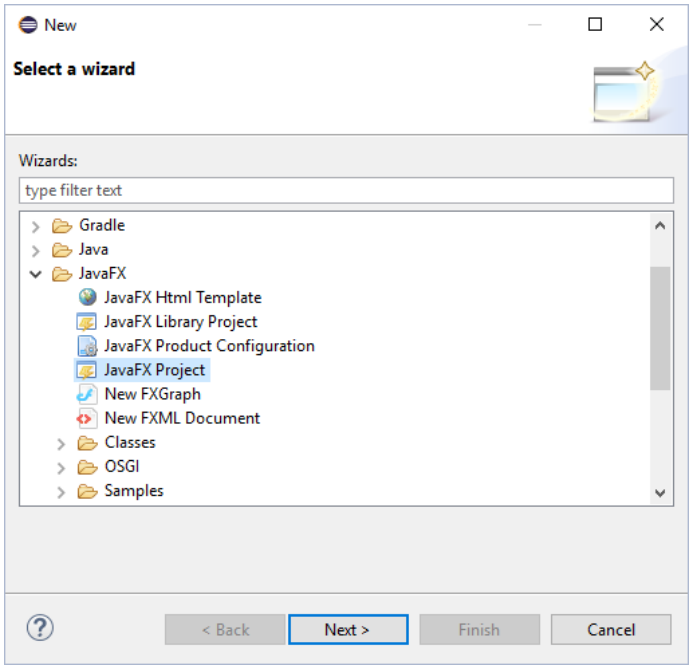
## 5.2. JavaFX projekti ja rakenduse loomine

Loome seekord *Eclipse*-s Java projektki asemel kohe uue JavaFX projekti, mis teeb palju asju automaatselt kohe ära, mida esimese näiterakenduse juures ise kirjutama pidi. Selleks tuleb valida *Eclipse*-s *File -> New -> Other…*



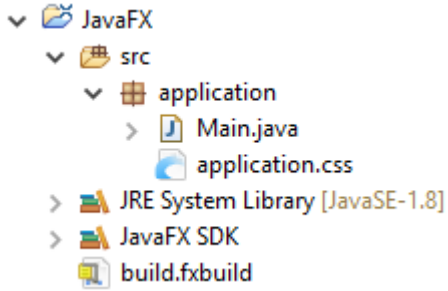

*Joonis 14 JavaFX projekti loomine*

Edasi tuleb valida *JavaFX* ja sealt alt *JavaFX Project* ning valida *Next >*



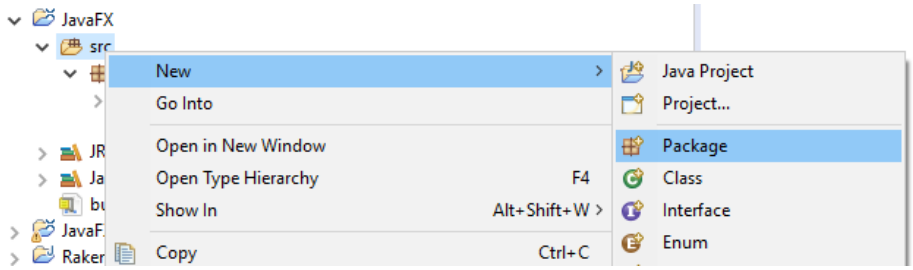

*Joonis 15 JavaFX projekti loomine 2*

Järgmisena tuleb ära määrata *Project Name* ja valida *Finish*, nii nagu tehtud siin: (**Joonis 5)**. Luuakse JavaFX projekt, koos *main* klassi ja tühja *stylesheet-iga.*



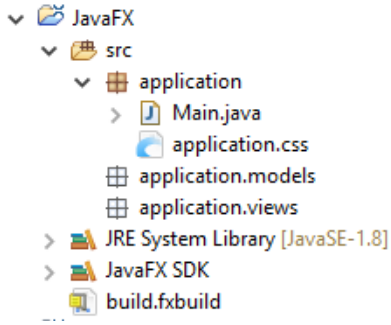

*Joonis 16 JavaFX projekt koos Main klassi ja stylesheet-iga*

Loome *src* alla kaks uut *package*-it, selleks tuleb projektis *src* peal parem klikk teha ja valida *New -> Package.*




*Joonis 17 Uue package loomine*

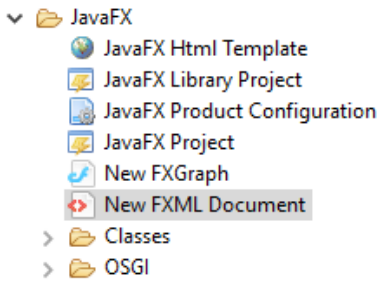
Esimesena loome uue *package*, sisestades *Name* alla *application.views* ja valime *Finish*. Selle *package* alla loome pärast *SceneBuilder*-it kasutades rakendusele vajalikud stseenid.
Teiseks loome uue *package*, sama moodi nagu eelmise aga nimeks paneme *application.models.* Sinna tuleb pärast rakenduses kasutatavate andmete hoidmiseks vajalik klass luua.




*Joonis 18 Projekt peale package-ite loomist*

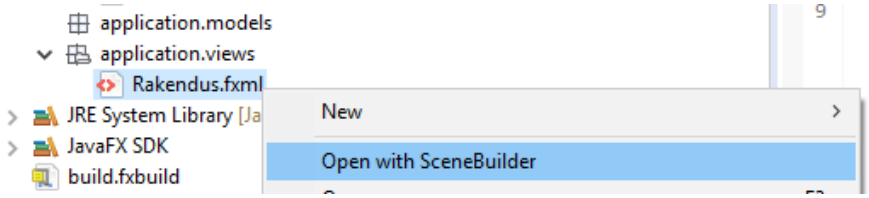
Selleks, et *SceneBuilder*-iga kasutajaliidest tegema hakata, loome *application.views* alla uue *FXML* faili, mis hoiab rakenduse kasutajaliidese osa programmi koodist eraldi. Selleks tuleb teha *application.views package* peal parem klikk ja valida *New -> Other…*

Avatavas aknas tuleb valida JavaFX ja selle alt *New FXML Document* ning siis *Next >*



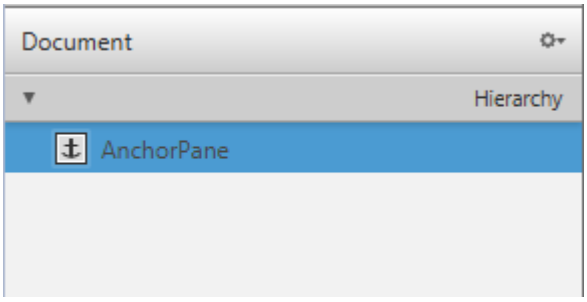

*Joonis 19 New FXML Document*

Järgmisel lehel määrame rakenduse nimeks kirjutades *Name* alla *Rakendus, RootElement* valime *Anchor Pane* ning vajutame *Finish*, mille peale luuakse *FXML* fail. Avame selle faili *Scene Builder*-iga, selleks teeme loodud *Rakendus.FXML* peal parem klikk ja valime *Open with SceneBuilder*, mille peale avatakse *Scene Builder*-is *Rakendus.FXML*.

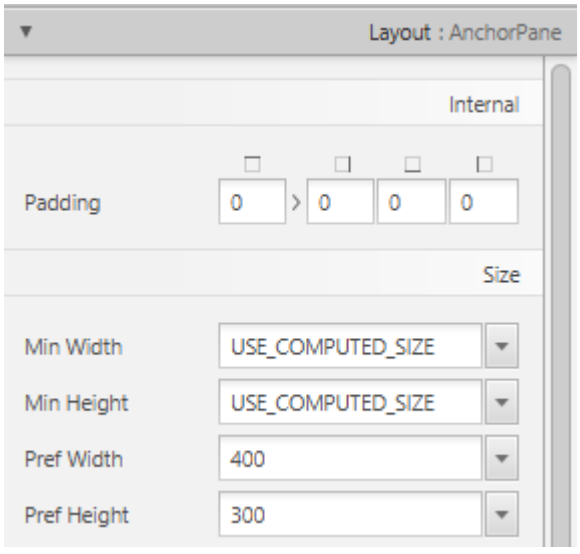



*Joonis 20 FXML faili avamine Scene Builder-iga*

Valime *Scene Builder*-is vasakult poolt *Document* ja *Hierarchy* alt *AnchorPane* ja muudame paremal pool *Layout* alt ära *Pref Width* 400 ja *Pref Height* 300.

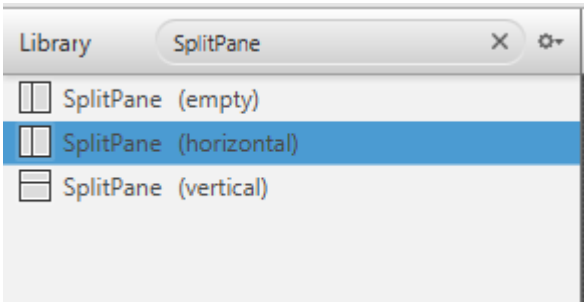



*Joonis 21 AnchorPane valimine*

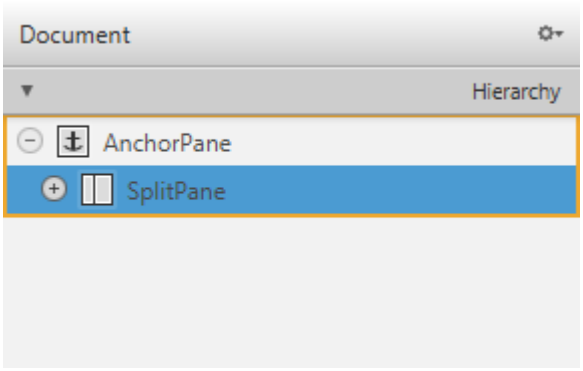



*Joonis 22 AnchorPane suuruse muutmine*

Järgmiseks lisame *AnchorPane* alla *SplitPane (horizontal)*, otsides ülevalt selle otsingukasti järgi üles ja siis tõstes selle hiirega *AnchorPane* peale.

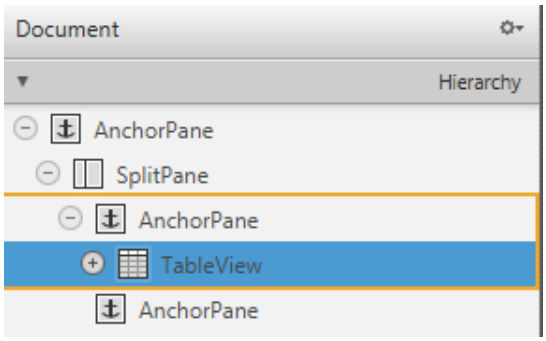



*Joonis 23 SplitPane otsimine*



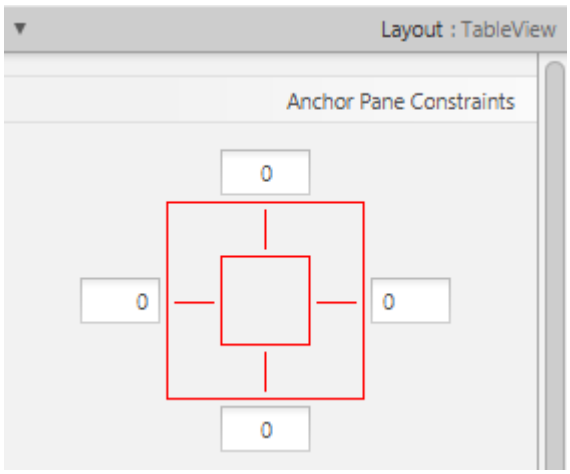

*Joonis 24 SplitPane AnchorPane sees*

Teeme *SplitPane* peal parema kliki ja valime *Fit to Parent*, sellega katab *SplitPane* ära kogu *AnchorPane* ala. Järgmiseks otsime otsingukastist üles *TableView* ja tõstame selle *SplitPane* all olevas esimesse (vasakusse) *AnchorPane.*



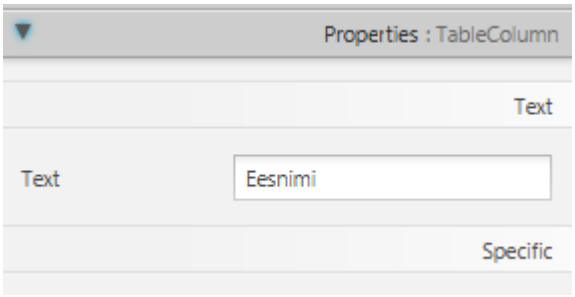

*Joonis 25 TableView paigutus*

Valime *TableView* ja paremal pool menüüs *Layout* all sarnaselt nagu (**Joonis 22),** muudame seal kõik *Anchor Pane Constraints* küljed 0.




*Joonis 26 Achor Pane Constraints muutmine*

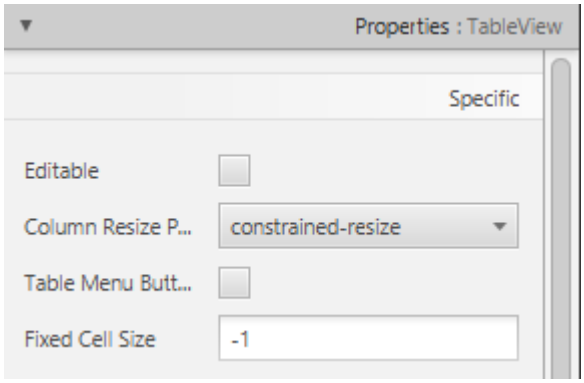
Muudame *TableView* all oleva *TableColumn C1* teksti ära, selleks valime *TableView* alt *TableColumn C1* ja paremal menüüs valime *Layout* asemel *Properties* ja muudame seal ära *Text* välja ja kirjutame sinna *Eesnimi.*




*Joonis 27 TableColumn tekstiväärtuse muutmine*

Teeme sama ka *TableColumn C2* aga tekstiks paneme sinna *Perekonnanimi.*

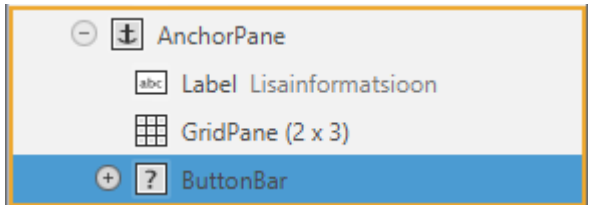
Hetkel ei mahu perekonnanime tulp täies mahus ära vasakusse *AnchorPane*, selleks tuleb valida *TableView* ja paremal pool menüüs *Properties* all valida *Column Resize Policy – constrained-resize.*




*Joonis 28 Column Resize Policy määramine*

Nüüd lisame *SplitPane* alumisse (paremasse) *Anchor* pane uue *Label*-i, selleks otsime otsingukastist *Label* ja tõstame selle alumise *AnchorPane* peale. Samamoodi nagu tegime ka *TableView* endaga (**Joonis 26),** muudame ära ka *Label Anchor Pane Constraints*, määrates ülemiseks ja vasakuks väärtuseks 5. Sarnaselt nagu muutsime ära *TableColumn (*Joonis 27*)* teksti muudame ära ka *Label* teksti ja paneme väärtuseks *Lisainformatsioon.*

Loome alumisse (paremasse) *AnchorPane* ka veel ühe *GridPane* ja ühe *ButtonBar (FX8).*

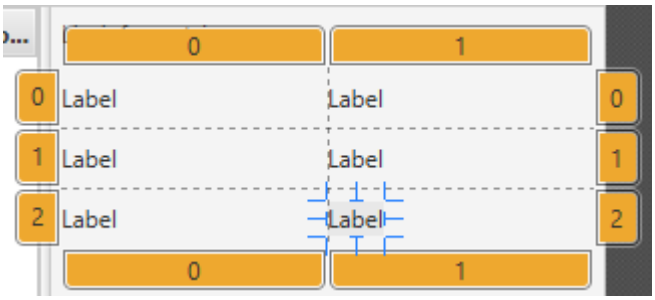



*Joonis 29 Loodud GridPane ja ButtonBar*

Valime *GridPane* ja muudame paremal menüüs *Layout* alt ära selle *Anchor Pane Constraints*, määrates ülemiseks väärtuseks 30, vasakule ja paremale 5. Teeme sama ka lisatud *ButtonBar*-iga, määrates selle paremaks ja alumiseks *Anchor Pane Constraints* väärtuseks 5.

Muudame ära nupu tekstiväärtuse, valides nupu ja paremalt poolt menüüst *Properties* alt *Text* lahter (**Joonis 27**), nupu teksti väärtuseks paneme *Kustuta.*

Nüüd lisame igasse *GridPane* lahtrisse *Label*, selleks otsime otsingukastist üles *Label* ja liigutame selle otse eelvaate peal olevatesse lahtritesse.




*Joonis 30 Kõik GridPane Label-id*

Muudame esimeses veerus olevate *Label* tekstiväärtused ära, esimeseks paneme *Sugu,* teiseks *Pikkus* ja kolmandaks *Kaal.*


*Joonis 31 GridPane Label-ite väärtused*

Salvestame loodud *FXML* faili valides ülevalt menüüst *File -> Save*

Läheme tagasi *Eclipse* juurde ja avame projektis *application package* all *Main.java* klassi ja kustutame *Override* meetodi seest *try* ja *catch* osa ära.

```
@Override
	public void start(Stage primaryStage) {

	}
```

Lisame olemasolevate *import*-ide alla veel:

```
import javafx.fxml.FXMLLoader;
import javafx.scene.layout.AnchorPane;
```

Deklareerime *Main* klassi alguses ühe *private* muutuja *Stage*.

```
private Stage primaryStage;
```

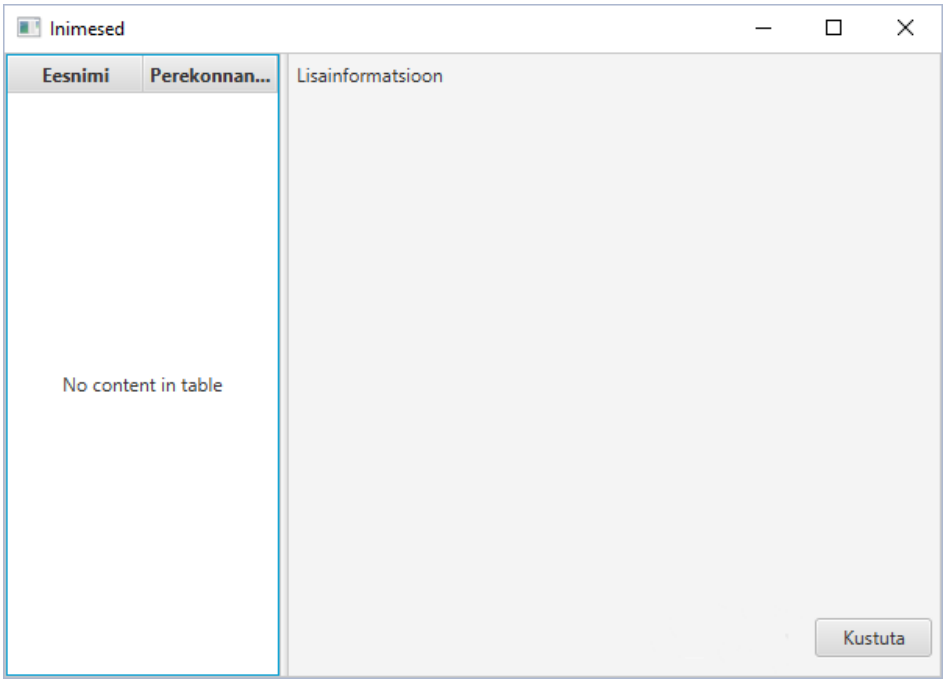
Loome peale *Override* meetodit uue meetodi, mis laeb meie loodud *FXML* faili, loob uue *BorderPane*, asetab *FXML* abil loodud elemendid *BorderPane* sisse, loob uue stseeni ja lõpuks kuvab stseeni välja.

```
public void rakenduseVaade(){
		FXMLLoader loader = new FXMLLoader();
		loader.setLocation(Main.class.getResource("views/Rakendus.fxml"));
		AnchorPane rakendus;
		try {
			BorderPane pane = new BorderPane();
			Scene stseen = new Scene(pane, 600, 400);
			rakendus = (AnchorPane) loader.load();
			pane.setCenter(rakendus);

			primaryStage.setScene(stseen);
			primaryStage.show();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
```

Lisame *Override* meetodi sisse koodi, mis määrab ära rakenduse pealkirja ja kuvab välja eelnevalt loodud rakenduse vaate.

```
@Override
	public void start(Stage primaryStage) {
		this.primaryStage = primaryStage;
		this.primaryStage.setTitle("Inimesed");

		rakenduseVaade();
	}
```

Peale seda saab rakenduse käivitada vajutades käivitamise nuppu üleval menüüs (**Joonis 10**).




*Joonis 32 Käivituv rakendus*

Selleks, et uusi inimesi tabelis hoida loome oma *application.models package* alla uue Java klassi, mille nimeks paneme *Inimene.* (**Joonis 7**).

Lisame klassile vajalikud *import*-id.

```
import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
```

Deklareerime klassi alguses ära vajalikud muutujad.

```
public class Inimene {

	private final StringProperty eesNimi;
	private final StringProperty perekonnaNimi;
	private final StringProperty sugu;
	private final IntegerProperty pikkus;
	private final IntegerProperty kaal;

}
```

Loome deklareeritud muutujate järele uue konstruktori.

```
public Inimene(String eesNimi, String perekonnaNimi, String sugu, Integer pikkus,
Integer kaal){
		this.eesNimi = new SimpleStringProperty(eesNimi);
		this.perekonnaNimi = new SimpleStringProperty(perekonnaNimi);
		this.sugu = new SimpleStringProperty(sugu);
		this.pikkus = new SimpleIntegerProperty(pikkus);
		this.kaal = new SimpleIntegerProperty(kaal);
	}
```

Loome eesnime kohta 3 meetodit, esimene tagastab väärtuse, teine määrab väärtuse ja kolmas tagastab *Property*.

```
public String getEesNimi(){
	return eesNimi.get();
}

public void setEesNimi(String eesNimi){
	this.eesNimi.set(eesNimi);
}

public StringProperty eesNimiProperty(){
	return eesNimi;
}
```

Teeme sama ka perekonnanime ja soo kohta.

Pikkuse ja kaalu meetodid tulevad ka sarnaselt, aga nendes on kasutusel *integer*-id.

```
//Pikkus
	public Integer getPikkus(){
		return pikkus.get();
	}

	public void setPikkus(Integer pikkus){
		this.pikkus.set(pikkus);
	}

	public IntegerProperty pikkusProperty(){
		return pikkus;
	}

//Kaal
	public Integer getKaal(){
		return kaal.get();
	}

	public void setKaal(Integer kaal){
		this.kaal.set(kaal);
	}

	public IntegerProperty kaalProperty(){
		return kaal;
	}
```

Läheme tagasi *Main.java* klassi juurde. Lisame deklareeritud muutujate järele veel ühe muutuja *private ObservableList.*

```
private ObservableList<Inimene> inimesteAndmed =
FXCollections.observableArrayList();
```

Lisame ka juurde kaks uut *importi*

```
import application.models.Inimene;
import javafx.collections.ObservableList;
```

Loome peale deklareeritud muutujaid uue konstruktori ja lisame selle sees mõned inimesed.

```
public Main(){
		inimesteAndmed.add(new Inimene("Juku", "Tammik", "Mees", 172, 70));
		inimesteAndmed.add(new Inimene("Kati", "Kuusik", "Naine", 161, 55));
		inimesteAndmed.add(new Inimene("Kalle", "Maasik", "Mees", 186, 83));
	}
```

Peale seda konstruktorit loome uue meetodi, mis tagastaks inimeste andmed.

```
public ObservableList<Inimene> tagastaInimesteAndmed(){
		return inimesteAndmed;
	}
```

Selleks, et loodud informatsiooni rakenduse tabelis kuvada tuleb luua *application.views package* alla uus klass *InimesteKontroller (***Joonis 7***).*

Lisame loodud klassi uued *import*-id.

```
import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import application.Main;
import application.models.Inimene;
```

Järgmiseks deklareerime muutujad, siin peab muutujate ja ka meetodite ette kirjutama *@FXML*, sest see tagab *FXML* failile õigused ligipääseda *private* väljadele ja meetoditele.

```
@FXML
private TableView<Inimene> inimesteTabel;
@FXML
private TableColumn<Inimene, String> eesNimeTulp;
@FXML
private TableColumn<Inimene, String> perekonnaNimeTulp;

@FXML
private Label eesNimeLabel;
@FXML
private Label perekonnaNimeLabel;
@FXML
private Label suguLabel;
@FXML
private Label pikkusLabel;
@FXML
private Label kaalLabel;
```

Peale deklareeritud muutujatid viitame *Main* klassile ja loome tühja konstruktori.

```
private Main mainKlass;

public InimesteKontroller(){

	}
```

Lähtestame *InimesteKontroller*-i.

```
@FXML
private void initialize(){
	eesNimeTulp.setCellValueFactory(CellData ->
CellData.getValue().eesNimiProperty());
	perekonnaNimeTulp.setCellValueFactory(CellData ->
CellData.getValue().perekonnaNimiProperty());
}
```

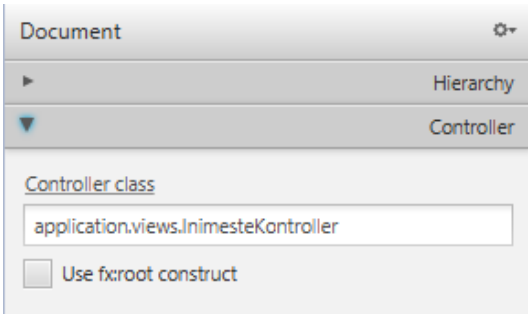
Peale seda lisame meetodi, mis lisab *Main* klassis olevast *ObservableList*-ist väärtused rakenduse tabelisse.

```
public void setMainApp(Main mainKlass){
	this.mainKlass = mainKlass;

	inimesteTabel.setItems(mainKlass.tagastaInimesteAndmed());
}
```

Läheme tagasi *Main* klassi juurde ja ühendame loodud *InimesteKontroller*-i *Main* klassiga. Lisame *rakenduseVaade()* meetodi sisse 2 rida koodi, peale *pane.setCenter(rakendus)* rida..

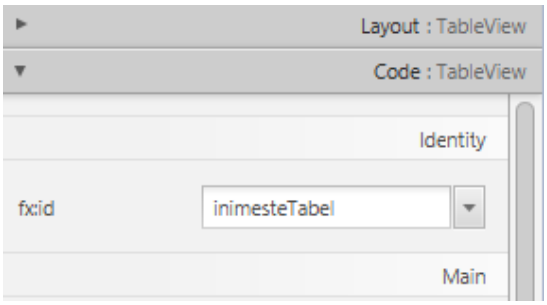
```
InimesteKontroller kontroller = loader.getController();
kontroller.setMainApp(this);
```

Lisame järgmisena *Rakendus.fxml* failile loodud *InimesteKontroller*-i. Selleks avame *application.views* all *Rakendus.fxml* faili *SceneBuilder*-iga (**Joonis 20**). Valime vasakult *Hierarchy* asemel *Controller* menüü ja lisame *Controller class* lahtrisse oma *InimesteKontroller*-i.




*Joonis 33 Kontrolleri lisamine SceneBuilder-is*

Läheme uuesti tagasi *Hierarchy* juurde ja valime seal *TableView*. Paremal menüüs valime *Code* ja seal muudame ära *fx:id*.

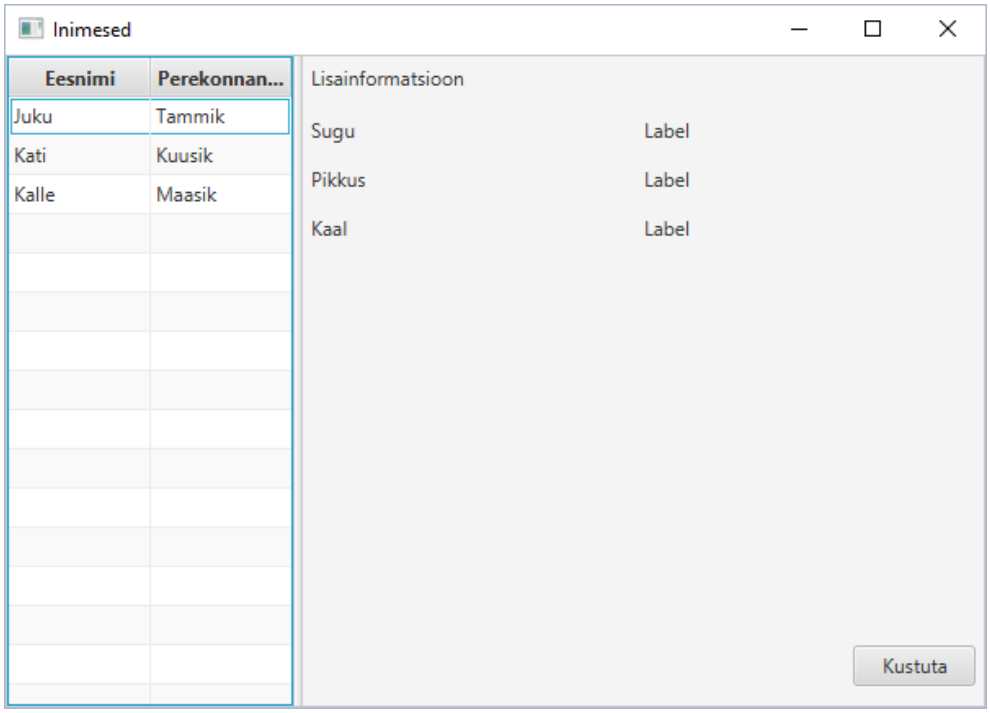



*Joonis 34 TableView fx:id muutmine*

Muudame ära ka vasakul *TableView* all olevad tulbad, selleks tuleb valida tulp ja muuta selle *fx:id* paremal *Code* all ära. Esimese tulba väärtuseks tuleb *eesNimeTulp* ja teise tulba väärtuseks *perekonnaNimeTulp.*

Järgmiseks tuleb *GridPane* teises tulbas olevate *Label*-ite *fx:id* vastavalt ära muuta. Esimese rea *Label fx:id* väärtus on *suguLabel,* teise rea oma *pikkusLabel* ja kolmanda rea oma *kaalLabel.*

Kui nüüd rakendus käivitada, siis kuvatakse tabelis *Main* klassis loodud inimesed.



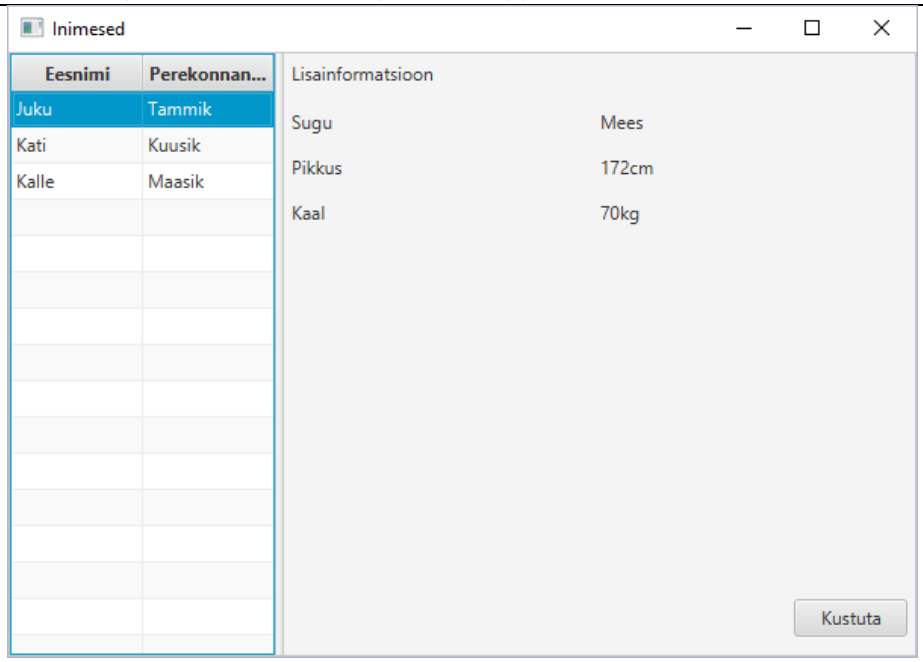

*Joonis 35 Rakendus koos tabeli väärtustega*

Selleks, et inimese valimisel kuvataks tema kohta paremal lisainformatsioon, peab avama *application.views* alt *InimesteKontroller* ja lisama sinna meetodi, mis kuvab välja lisainformatsiooni.

```
private void kuvaLisainfo(Inimene inimene){
	if (inimene != null) {
		suguLabel.setText(inimene.getSugu());
		pikkusLabel.setText(String.valueOf(inimene.getPikkus() + "cm"));
		kaalLabel.setText(String.valueOf(inimene.getKaal() + "kg"));
	}else{
		suguLabel.setText(" ");
		pikkusLabel.setText(" ");
		kaalLabel.setText(" ");
	}
}
```

*InimesteKontrolleri*-s oleva *initialize* meetodi lõppu tuleb lisada ka 2 rida.

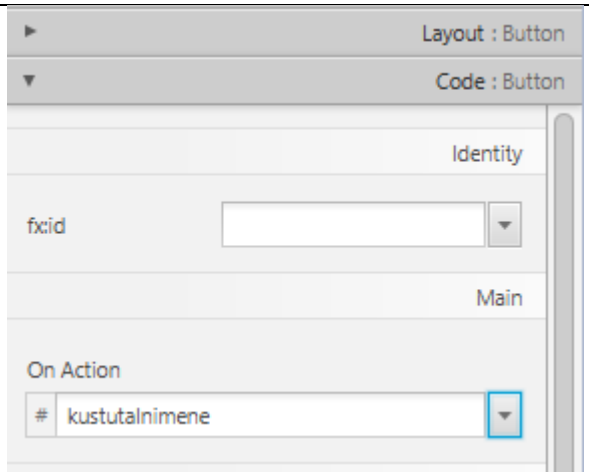
```
kuvaLisainfo(null);

inimesteTabel.getSelectionModel().selectedItemProperty().addListener((observable,
oldValue, newValue) -> kuvaLisainfo(newValue));
```




*Joonis 36 Rakendus koos kuvatava lisainformatsiooniga*

Tabelist kustutamiseks tuleb luua uus meetod ja lisada *SceneBuidler*-is *Kustuta* nupule loodud meetod.

```
@FXML
private void kustutaInimene(){
	int valitudInimene = inimesteTabel.getSelectionModel().getSelectedIndex();
	inimesteTabel.getItems().remove(valitudInimene);
}
```




*Joonis 37 Kustuta nupule meetodi lisamine*

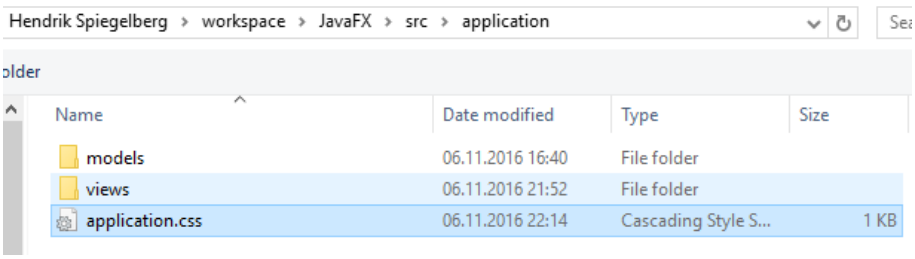
# 6. Loodud rakenduse kujundamine CSS kasutades

## 6.1. *Stylesheet*-i ühendamine rakendusega

*Stylesheet*-i saab ühendada rakendusega mitut moodi. Üheks võimaluseks on seda läbi koodi lisada, selleks tuleb avada *Main.java* klass ja seal leida *rakenduseVaade()* meetodi sees üles rida, kus luuakse stseen. Selle järele tuleb lisada üks rida koodi, mis võtab kasutusele projektis oleva *application.css.*

```
stseen.getStylesheets().add("application/application.css");
```

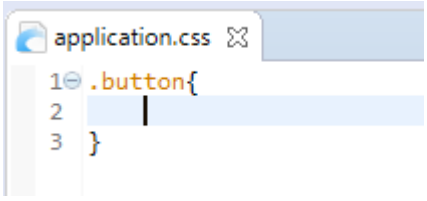
*Stylesheet*-i saab lisada ka vaatele läbi *SceneBuilder* rakenduse. Selleks tuleb avada *application.views package* alt *Rakendus.fxml SceneBuilderis* (**Joonis 20**). Siis tuleb vasakult poolt valida *Hierarchy* alt kõige ülemine *AnchorPane.* Kui see on valitud, siis paremalt poolt menüüst tuleb lisada *Properties* alt uus *stylesheet,* vajutades selleks seal olevat + nuppu ja liikuda kohta, kus *stylesheet* asub, hetkel peaks asukohaks olema: *C:\Users*kasutaja*\workspace\JavaFX\src\application.* Kui *stylesheet* on leitud, siis tuleb see avada vajutades *Open.* Kui *stylesheet* on lisatud, siis tuleb *Rakendus.fxml* ka ära salvestada, selleks valida *File -> Save.*




*Joonis 38 Application.css asukoht failisüsteemis*

## 6.2. *Styleheet*-i muutmine

Algselt projekti luues tekitati *application package* sisse tühi *application.css* fail. Avame selle faili ja lisame esimesena sinna vajaliku, et muuta rakenduses olev *Kustuta* nupu taust punaseks. Selleks on vaja luua klassi *selector: .button,* mis valib kõik rakenduses olevad nupud.




*Joonis 39 .button selector*

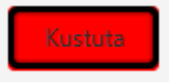
JavaFX rakenduste puhul algavad kõik *selector*i sees olevad reeglid prefiksiga *-fx-*. Selleks, et muuta nuppudel taustavärv punaseks on vaja *selector*i sisse lisada:

```
-fx-background-color: red;
```


*Joonis 40 Kustuta nupp peale taustavärvi muutmist*

Järgmiseks loome ühe pseudo-klassi *selector*i: *.button:hover*. Selle selektori sees olevaid reegelid kasutatakse siis, kui hiirega nupule peale liikuda. Loome *selector*i sisse reegli, mis muudab nupule ümber musta ümarate nurkadega piiri.

```
.button:hover {
    -fx-border-color: black, black;
    -fx-border-width: 3, 3;
    -fx-border-style: solid;
    -fx-border-radius: 4, 4;
}
```

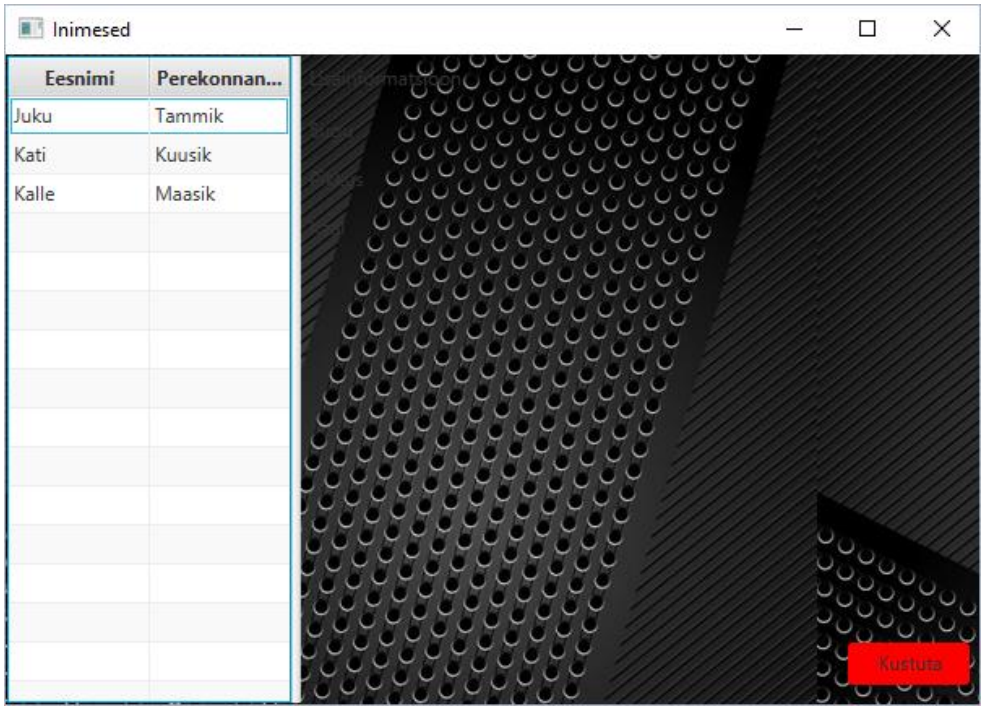



*Joonis 41 Nupu hover pseudoklassi muudatus*

Muudame järgmisena ära *SplitPane* tausta, asendades selle taustapildiga. Selleks tuleb luua uus pseudo-klassi *selector*: *.split-pane:horizontal* ja selle sisse lisada reegel:

```
-fx-background-image: url(img/metal.jpg);
```

*Url* järele sulgude sisse võib kirjutada kas kohaliku pildi asukoha või veebiaadressi.

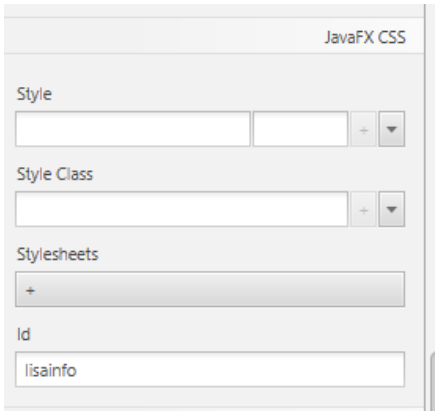



*Joonis 42 SplitPane taust asendatud pildiga*

Peale pildi lisamist muutus *Labelite* tekst loetamatuks, teeme teksti uuesti nähtavamaks. Muudame kõik *labelid* ära, selleks loome uue *selectori: .label* ja muudame labelite tekstivärvi ja teksti *fonti*.

```
.label {
	-fx-text-fill: white;
	-fx-font-family: "Arial";
}
```

Muudame *Lisainformatsiooni labe*-it. Selleks, et muutuks ainult *Lisainformatsiooni label*, lisame talle uue *id*. Selleks, et *id* lisada, tuleb avada *Rakendus.fxml SceneBuilder*-iga. Seal tuleb *Hierarchy* all olles valida *SplitPane* alumise *AnchorPane* seest *Label* ja muuta paremas menüüs *Properties -> JavaFX CSS* alt ära *Id*, määrame *Id* väärtuseks *lisainfo*.




*Joonis 43 Id lisamine SceneBuilder-is*

Lisame *application.css* alla uue *id selector*i: #lisainfo. Id *selector* valib ja muudab ainult neid elemente, mille *id* on sama, mis *selectoris* määratud. Muudame ära *Lisainformatsiooni labeli* teksti värvi, teksti suuruse ja muudame teksti *Boldiks*.

```
#lisainfo {
	-fx-font-size: 14pt;
	-fx-font-weight: bold;
}
```


*Joonis 44 Labelid peale nende stiili muutmist*

# Kokkuvõte

Käesolevas töös on antud ülevaade *JavaFX* ja *CSS* ning sellest, kuidas nad koos toimivad. Töö alguses on lühike sissejuhatus *JavaFX* ja *CSS.* Loodud on juhendid tarkvara paigaldamiseks, mis on vajalikud, *JavaFX* rakenduste loomiseks.

Lisaks on loodud kaks näiterakendust koos nende loomise juhendiga. Esimene rakendus näitab *JavaFX* põhiomadusi ja kasutab *JavaFX* vaikimisi saadaval *stylesheeti ja* teine kasutab oma loodud *stylesheeti*.

Teise rakenduse kujundamiseks on töö käigus loodud *stylesheet ja see JavaFX* rakendusega ühendatud. *Stylesheet*-i muutmiseks on loodud juhend, mis näitab ära, kuidas *CSS* kasutamine loodud *JavaFX* rakenduste puhul käib.

# Kasutatud kirjandus


1. Lowe, D. (2015). *JavaFX for dummies*. Hoboken, New Jersey: John Wiley & Sons, Inc.

2. Pouncey, I., & York, R. (2011). *Beginning CSS : Cascading Style Sheets for Web Design (3)*. Wrox.

3. Oracle. (2013). JavaFX CSS Reference Guide. Loetud aadressil https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html

4. Pawlan, M., & Castillo, C. (2013). JavaFX Overview Release 2.2.21, (April), 1–10. Loetud aadressil http://docs.oracle.com/javafx/2/overview/jfxpub-overview.pdf