מעבדה מספר 8

המשך היכרות עם JavaFX, תוכנית הכוללת Graphical user interface (GUI)

כפי שלמדנו במעבדה 3 - JavaFX היא קבוצה של חבילות גרפיות ומדיה המאפשרת למפתח תוכנה לתכנן, לכתוב ולבדוק אפליקציות עם ממשק משתמש נוח וידידותי שיכול לרוץ על פלטפורמות שונות.

מרכיבי תוכנה גרפית

כתיבת קוד ליצירת ממשק גרפי כוללת שלושה מרכיבים:

- מרכיב 1 - בחירת אלמנטים מסוגים שונים (כפתורים, רשימות, שדות טקסט, כותרות) שיופיעו על המסך/מסכי התוכנית.
- מרכיב 2 - הארגון הדו־ממדי של האלמנטים ופריסתם על המסך (בתחילת התוכנית ולאחר שינוי גודל החלון ע"י המשתמש/סגירתו/פתיחתו מחדש).
- מרכיב 3 - ההתנהגות הדינמית של האלמנטים בתגובה לפעולות של המשתמש/ת ("אירועים": הקלדה, הקלקה, גרירה).

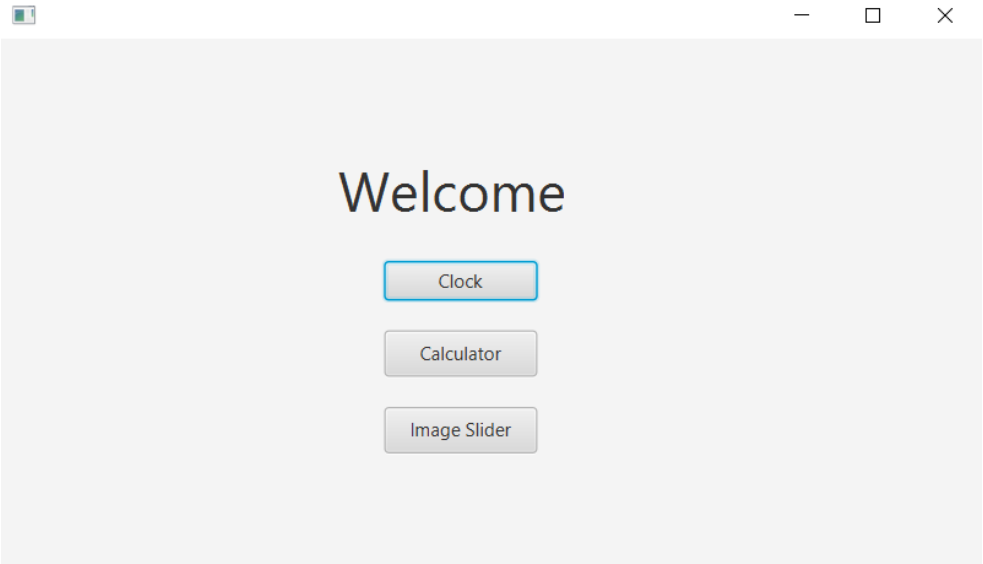
היום במעבדה נלמד על:

1) הוספת שעון רץ.
2) מעבר בין מסכים.
3) הצגת תמונות בעזרת ImageView.
4) הוספת תפריט.
5) הטענת אובייקט מוכן לPane (שימושי עבור Reuse באובייקט מוכן).
6) הוספת תיבת בחירה (comboBox)
7) הצגת חלוניות התראה.

במהלך המעבדה הנוכחית נכין ביחד תוכנית קטנה המכילה שעון דיגיטלי, לוח שנה וגלריית תמונות.

ראשית ניצור פרויקט חדש כפי שלמדנו במעבדה 3.

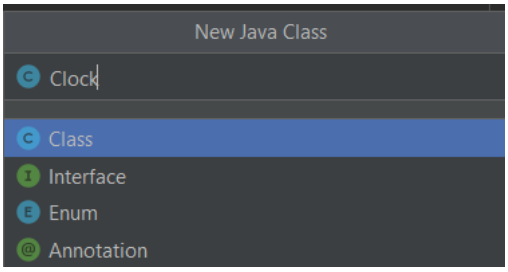
לאחר מכן נייצר את המסך הראשי – מסך שיכיל 3 כפתורים: מחשבון, שעון דיגיטלי וגלריית תמונות.

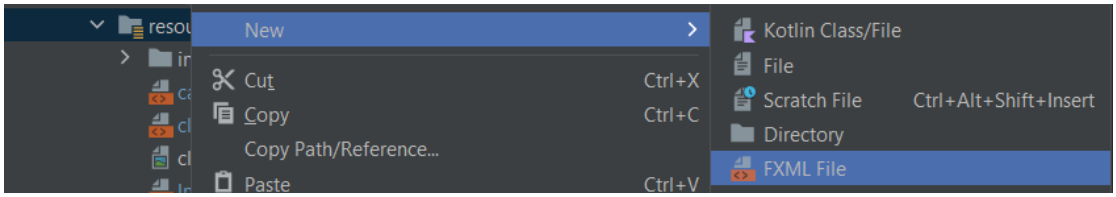


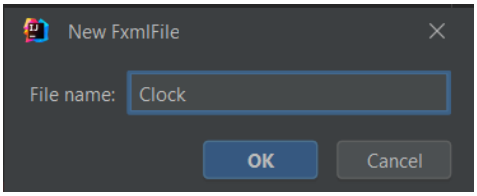
הוספת שעון רץ:

נוסיף לתוכנית שלנו שעון רץ.

על מנת להוסיף שעון רץ לפרויקט נוסיף 2 קבצים – קובץ java וקובץ fxml עבור הcontroller של המסך:





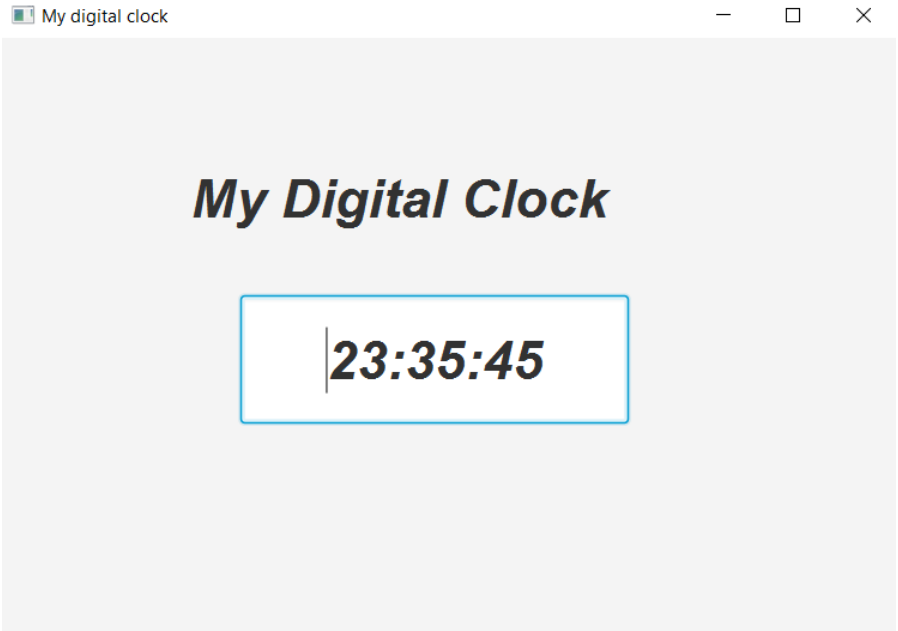


נוסיף למסך תיבת טקסט (textField) ואז נוסיף את שורות הקוד הבאות למתודה initialize של השעון:

```
DateTimeFormatter dtf = DateTimeFormatter.ofPattern("HH:mm:ss");
Timeline clock = new Timeline(new KeyFrame(Duration.ZERO, e -> {
    LocalTime currentTime = LocalTime.now();
    times.setText(currentTime.format(dtf));
}),
        new KeyFrame(Duration.seconds(1))
);
clock.setCycleCount(Animation.INDEFINITE);
clock.play();
```

הסבר: https://www.demo2s.com/java/javafx-timeline-tutorial-with-examples.html

```
18  public class Clock {
        2 usages
19      @FXML
20      private TextField times;
21
22
23      void initialize() throws IOException {
24
25          DateTimeFormatter dtf = DateTimeFormatter.ofPattern("HH:mm:ss");
26          Timeline clock = new Timeline(new KeyFrame(Duration.ZERO, e -> {
27              LocalTime currentTime = LocalTime.now();
28              times.setText(currentTime.format(dtf));
29          }),
30                  new KeyFrame(Duration.seconds( v: 1))
31          );
32          clock.setCycleCount(Animation.INDEFINITE);
33          clock.play();
```



הוסף לפרויקט שלנו שעון.

מעבר בין מסכים:

נוסיף לפרויקט מעבר בין מסכים.

על מנת לעשות זאת ראשית נוסיף למחלקה App 2 מתודות חדשות:

```
public static void setWindowTitle(String title) {
   appStage.setTitle(title);
}

public static void setContent(String pageName) throws IOException {
   Parent root= loadFXML(pageName);
   scene = new Scene(root);
   appStage.setScene(scene);
   appStage.show();
}
```

הסבר:

setWindowTitle(String title) – מתודה שמגדירה את כותרת החלון לString ששלחנו לה.

setContent(String filename) – מתודה שמגדירה את תוכן החלון בהתאם לקובץ fxml שאת שמו שלחנו לה (אין צורך להוסיף fxml. בסוף שם הקובץ כי הוא מתווסף במתודה עצמה).

כעת נשתמש בקוד הבא בכדי לעבור בין המסכים:

```
Platform.runLater(() -> {
    setWindowTitle("title");
    try {
        setContent("/filename");
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

הסבר:

Platform.runLater() – מתודת עזר שמגדירה הרצה מאוחרת יותר של הקוד המוגדר בתוכה. אנו משתמשים בה בכדי לעדכן את המסך של JavaFX Application. היתרון העיקרי של שימוש בPlatform.runLater() הוא שהריצה מתבצעת על application thread ולכן הקוד יתבצע בביטחה על ה -GUI, ללא צורך בסינכרוניזציה.

להסבר נוסף: https://www.demo2s.com/java/javafx-platform-runlater-runnable-runnable.html

https://math.hws.edu/javanotes/c12/s2.html

את הקוד הנ"ל נוסיף למתודה חדשה שנוסיף לApp שתדאג למעבר בין המסכים. נדאג להחליף את title לשם המסך שאליו אנו עוברים ובמקום filename נרשום את שם הfxml שמשויך למסך שאליו אנו רוצים לעבור.

לדוגמה המתודה שמטפלת במעבר מסכים תראה כך:

```
public static void switchScreen (String screenName){
  switch (screenName){
    case "clock":
      Platform.runLater(() -> {
        setWindowTitle("My Clock");
        try {
          setContent("/Clock");
        } catch (IOException e) {
          e.printStackTrace();
        }
      });
      break;
    case "Image Gallery":
....
}
```

בעזרת הקוד הנ"ל דאג שכאשר נלחץ על הכפתור של השעון במסך הראשי אנו נעבור למסך המתאים.

הצגת תמונות בעזרת ImageView:

JavaFX מספקת לנו אובייקט נוח להצגת תמונות שנקרא ImageView.

ImageView – אובייקט שמציג לנו תמונה שהטענו לו בעזרת המחלקה Image. בעזרת Image אנו יכולים להגדיר את הנתיב של התמונה, את הגודל הרצוי שלה, את השקיפות ועוד.

הסבר: https://www.educba.com/javafx-imageview/

איך אנו מבצעים זאת?:

- על מנת להוסיף מצגת תמונות לפרויקט נוסיף 2 קבצים – קובץ java וקובץ fxml עבור הצגת התמונות.
- נוסיף לקובץ fxml שלנו אובייקט מסוג imageView.

❖ נלך לcontroller שמשוייך לfxml בו הוספנו את imageView ונוסיף את השורות הבאות לinitialize:

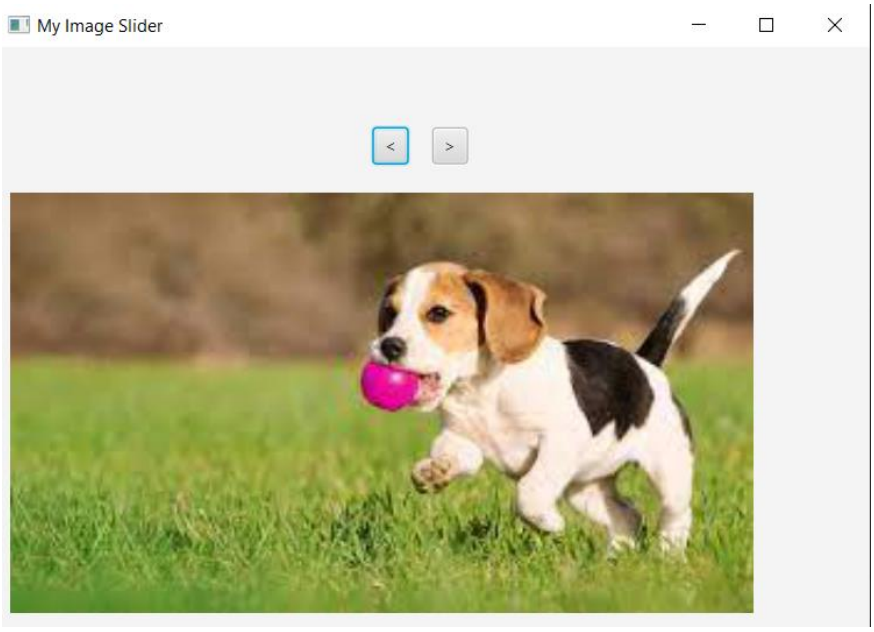
```
try {

    images = new Image[numOfImages];
    for (int i = 0; i < numOfImages; i++) {
        File file = new
File("C:\\Users\\User\\IdeaProjects\\JFXApps\\src\\main\\resour
ces\\images\\"+i+".jpg");
        images[i] =  new Image(file.toURI().toString());
    }
    imageView.setImage(images[j]);
} catch (Exception e) {
    e.printStackTrace();
}
```

הסבר:
אנו בעצם טוענים למערך מסוג Image את התמונות הנמצאות בתיקייה images ע"י ריצה על המערך הנ"ל והוספת תמונה אחת כל פעם ע"י שליחת הנתיב של התמונה לבנאי של המחלקה Image בכדי ליצור אובייקט חדש מסוג Image שישמר במערך. לאחר מכן אנו מגדירים לimageView להראות את התמונה הראשונה במערך.

שימו לב: לבנאי של File שלחתי את הנתיב של התמונה שרציתי להוסיף למערך התמונות. אני השתמשתי בתיקיית תמונות ששמרתי תחת resources בפרויקט הנוכחי ואת התמונות שמרתי כך ששמם הוא מספר שלם בין 0 ל14 לצורך נוחות של טעינת התמונות. למצגת שלכם הוסיפו איזה תמונות שתרצו ע"י נתינת הנתיב של התמונה המבוקשת. הוספתי לכם במודל תחת המעבדה את התיקיה שבה אני השתמשתי למקרה שתעדיפו להשתמש בה.

בכדי לעבור לתמונה הבאה/הקודמת נוסיף 2 כפתורים עבור מעבר שמאלה או ימינה.

נוכל אם נרצה גם לשנות את גודל התמונה, השקיפות שלה, הבהירות שלה וכו'.



הוסף לפרויקט שלנו גלריית תמונות בעזרת ImageView. דאג להוסיף לפחות 5 תמונות לגלריה.
כמו כן הוסף כפתורים למעבר בין תמונות.
בנוסף דאג שכאשר נלחץ על הכפתור של גלריית התמונות במסך הראשי אנו נעבור למסך המתאים.

הוספת תפריט:

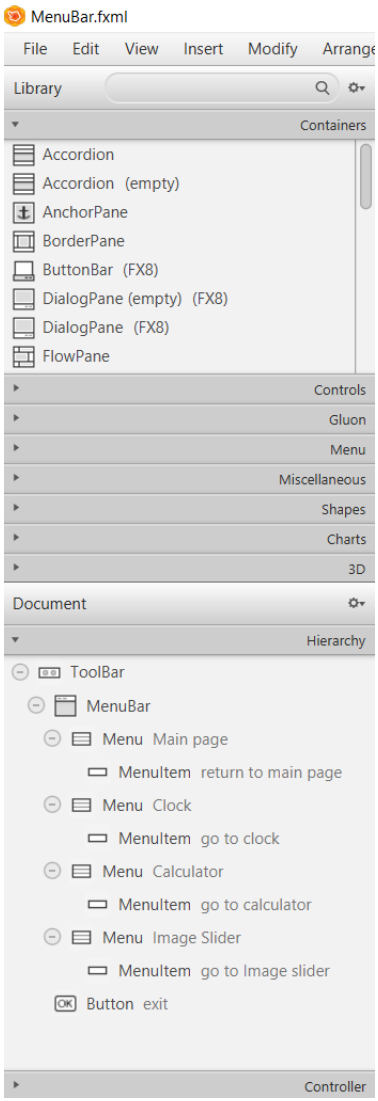
JavaFX מספקת לנו מבחר אובייקטים מסוג תפריט להגדרה נוחה של תפריט לבחירתנו.

נלמד היום על 2 מהם:

- **MenuBar** - תפריט המופיע בחלק העליון של החלון בדרך כלל ומכיל MenuItem לבחירה קלה של אפשריות שונות שהגדרנו לשימוש המשתמש (Intellij למשל אתם יכולים לראות דוגמה לתפריט מסוג זה בחלק העליון של התוכנה).

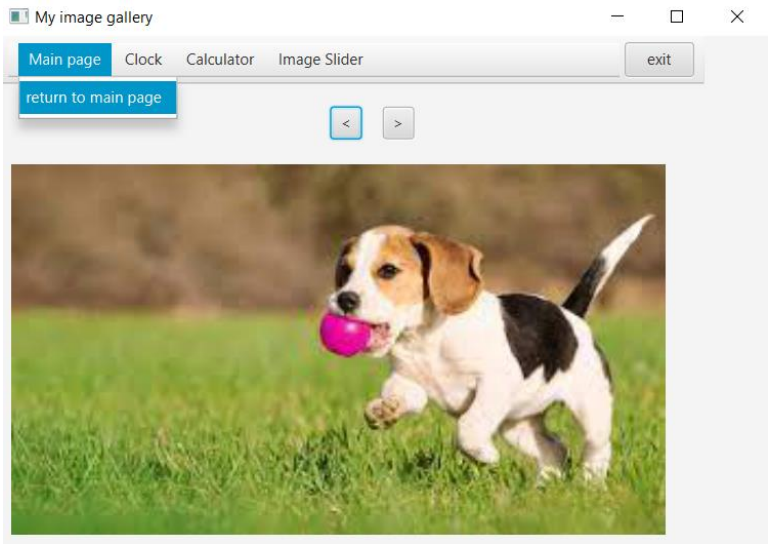
כדי להשתמש באובייקט מסוג זה נוסיף MenuBar וכמה Menu שנרצה לתפריט. בתוך כל Menu כזה נוסיף MenuItem שעבורם נגדיר את פעילות התוכנה כאשר אנו לוחצים עליהם (לדוגמה נוכל להוסיף Calculator,ImageSlider,Clock בשביל מעבר חלק בין האפשרויות הללו).

הסבר: http://tutorials.jenkov.com/javafx/menubar.html



עבור הוספת MenuBar בצע את השלבים הבאים:

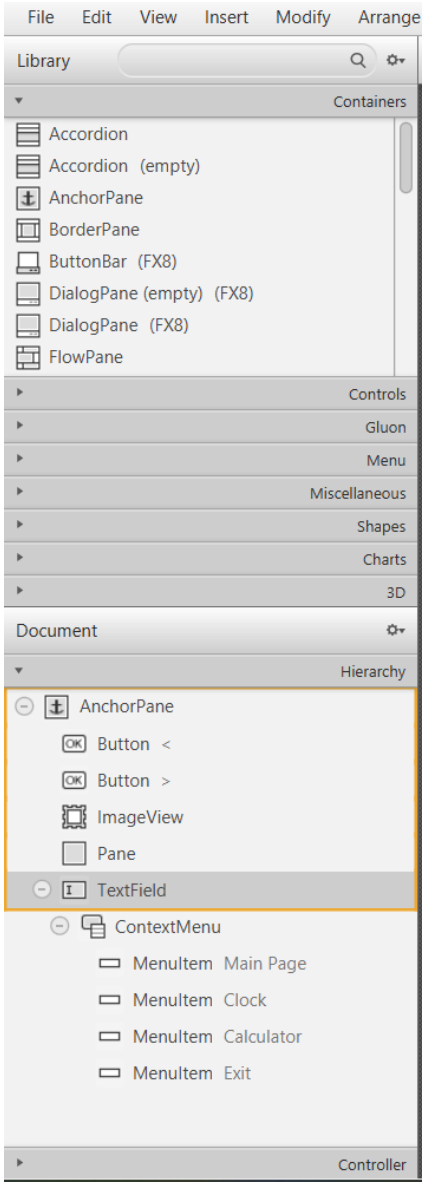
- ❖ צור קובץ fxml חדש שיקרא Menu או משהו בסגנון ובו תגדיר את התפריט ע"י הוספת אובייקט מסוג MenuBar שאליו נוסיף את התפריטים הנדרשים.
- ❖ צור קובץ java שישמש כcontroller של הMenuBar הזה שבו נגדיר את פעילות הmenuBar ומה נרצה שיקרה עבור לחיצות על האפשרויות השונות (לדוגמה מעבר בין מסכים).



- **ContextMenu** - תפריט חבוי שמופיע כאשר אנו לוחצים על המקש הימני בעכבר ומכיל בתוכו MenuItem לבחירה קלה של אפשריות שונות שהגדרנו לשימוש המשתמש (בIntellij למשל אתם יכולים לראות דוגמה לתפריט מסוג זה כאשר אנו לוחצים על המקש הימני בעכבר).

  כדי להשתמש באובייקט מסוג זה נוסיף ContextMenu ואז בתוכו נוסיף MenuItem שעבורם נגדיר את פעילות התוכנה כאשר אנו לוחצים עליהם (לדוגמה נוכל להוסיף Calculator,ImageSlider,Clock בשביל מעבר חלק בין האפשרויות הללו).

  הסבר: http://tutorials.jenkov.com/javafx/contextmenu.html



עבור הוספת ContextBar בצע את השלבים הבאים:

- ❖ פתח את קובץ הfxml שבו אתה מעוניין להוסיף את התפריט.
- ❖ הוסף textArea בגודל המסך עם opacity=0 (כי אנחנו לא רוצים לראות את תיבת הטקסט. היא משמשת לנו רק לצורך הוספת התפריט לתוכה מכיוון שcontextMenu ניתן להוסיף רק לתוך אובייקטים).
- ❖ הוסף את הContextMenu בתוך הtextArea ואליו הוסף כמה MenuItem שתרצה.
- ❖ הוסף לקובץ java שמשמש כcontroller של המסך בו אנו נמצאים שורות קוד בכדי להגדיר את פעילות הcontextMenu ומה נרצה שיקרה עבור לחיצות על האפשרויות השונות (לדוגמה מעבר בין מסכים).

הוסף תפריט באחד הדרכים לפרויקט שלנו.

הטענת אובייקט מוכן לPane:

כשנרצה להשתמש באובייקט מוכן בתוכנית שלנו נוכל להוסיף Pane במקום שבו נרצה להוסיף את האובייקט המוכן ונטען לתוכו את האובייקט. אופציה זאת תקל עלינו לעשות שימוש חוזר באובייקט שאנו הגדרנו/שמישהו אחר הגדיר עבורנו. בתוכנית שלנו נשתמש באובייקט מוכן פעמיים:

- נוסיף את המחשבון שיצרנו במעבדה 3.
- נוסיף את התפריט שהכנו בסעיף הקודם לכל המסכים.

נלך למתודת ה-initialize של המסכים שאליהם אנו רוצים להוסיף את התפריט ונוסיף את השורות הבאות:

```
Parent menuBarParent = org.example.App.loadFXML("/Menubar");
MenubarPane.getChildren().clear();
MenubarPane.getChildren().add(menuBarParent);
```

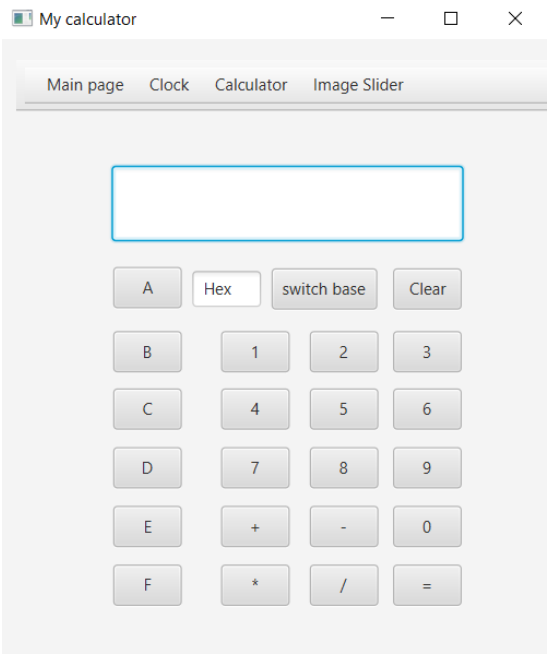
באופן דומה נטען את המחשבון למסך של המחשבון:

```
package org.example.Calculator;

import javafx.fxml.FXML;
import javafx.scene.Parent;
import javafx.scene.layout.Pane;

public class CalculatorController {

    @FXML
    private Pane pane;

    @FXML
    void initialize() {
        //add code here
    }

}
```

להסבר נוסף על Pane:

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/Pane.html

הוסף לכל המסכים בפרויקט שלנו את התפריט שיצרנו והוסף את המחשבון לפרויקט.

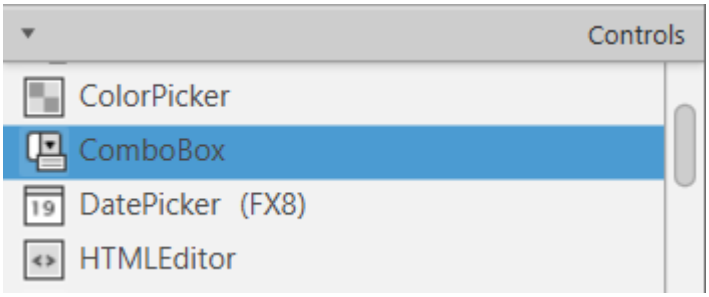


הוספת תיבת בחירה (ComboBox):

תיבת הבחירה מאפשרת למשתמש לבחור אפשרות אחת מתוך רשימה קיימת.

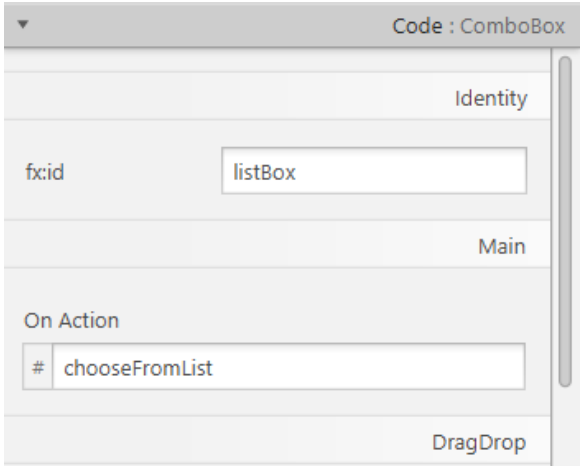
איך טוענים רשימה?

<u>שלב 1:</u> יש לפתוח את ה- Controls בחלק השמאלי העליון של ה־SceneBuilder ולגרור את תיבת הבחירה למסך התצוגה ולמקמם במקום הרצוי.

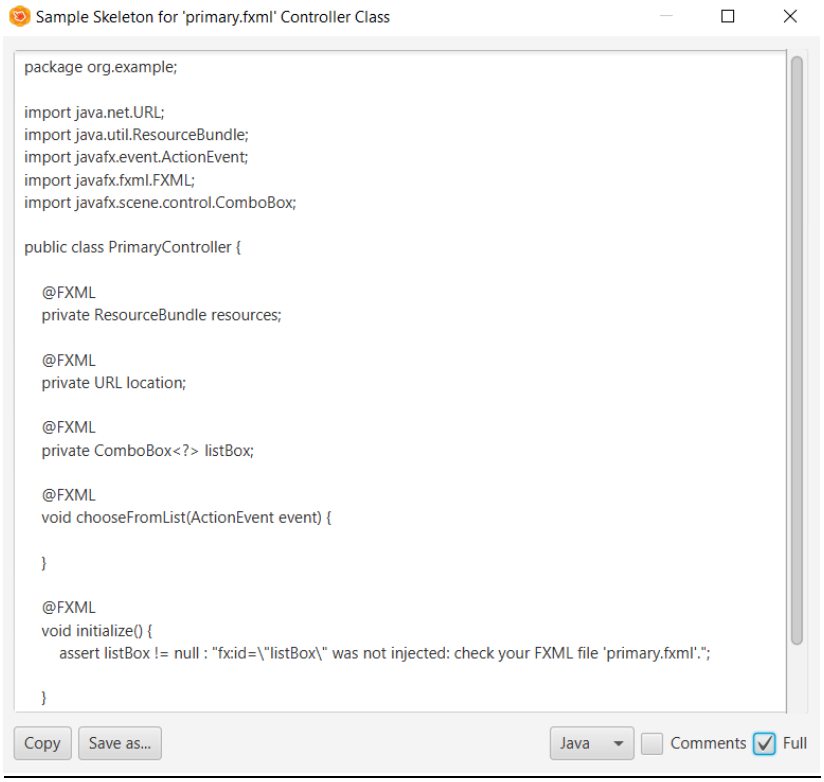


כאשר ממקמים את אלמנטי ה־UI ניתן להבחין בקווים אדומים המופיעים ונעלמים. קווים אלה מאפשרים למקם בקלות אלמנטים ביחס לאלמנטים אחרים או למרכז/ליישר את האלמנטים על המסך.

<u>שלב 2:</u> יש לבחור בכפתור תחת ה־Hierarchy section בצד ימין של ה־SceneBuilder ולפתוח את החלק שנקרא Code, המציג את תכונות הקוד המתייחסות לכפתור. יש לקבוע את fx:id ל־"listBox" ואת On Action ל-"chooseFromList".



שלב 3: יש לבחור באופציה **View->Show Sample Controller Skeleton** בתפריט הראשי של ה־SceneBuilder. כתוצאה מכך יוצג הקוד הראשוני של מחלקת ה־controller. בתחתית החלון בחרו "Comments" ואל תבחרו "Full". אחר כך העתיקו את הקוד ע"י לחיצה על "Copy" והעבירו אותו למחלקה הרצויה ב־IntelliJ.



להלן קוד לדוגמה:

```
/**
 * Sample Skeleton for 'primary.fxml' Controller Class
 */
package org.example;
```

```
import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.ComboBox;

public class PrimaryController {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private ComboBox<String> listBox;

    @FXML
    void chooseFromList(ActionEvent event) {

    }

    @FXML
    void initialize() {
        assert listBox != null : "fx:id=\"listBox\" was not injected:
check your FXML file 'primary.fxml'.";

    }

}
```

שלב 4: תוספת הקוד הנדרשת תהיה בinitialize וב chooseFromList:

ראשית בהגדרה של הComboBox נגדיר איזה סוג אובייקט הרשימה מקבלת (הרשימה אינה יכולה לקבל טיפוסים פרימיטיבים).

Initialize() – המתודה שמאתחלת את המסך. נוסיף את השורות הבאות למתודה כדי לאתחל את הרשימה של הComboBox:

```
listBox.getItems().add(first);
listBox.getItems().add(second);
```

וכו' בכדי להוסיף את האובייקטים לרשימה.

chooseFromList(ActionEvent event) – מתודה זו תיקרא עם קבלת האירוע ותטפל בבחירה מתוך הרשימה. נוסיף את השורה הבאה למתודה:

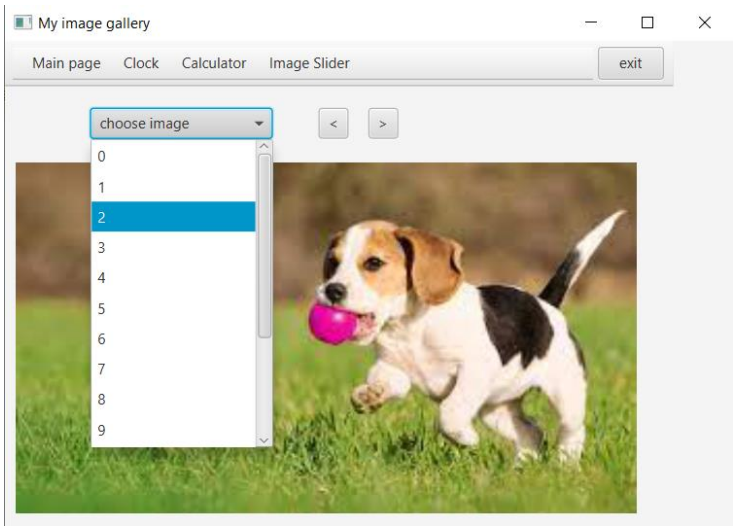
```
String chosen = listBox.getSelectionModel().getSelectedItem();
```

לאחר שורה זו המשתנה chosen יהיה שווה בעצם למחרוזת שנבחרה מהרשימה.

פונקציות שימושיות נוספות לעבודה עם רשימות של ComboBox:

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/ComboBox.html

הוסף לגלריית התמונות שלנו רשימה על מנת שהמשתמש יוכל לבחור את התמונה הרצויה.



הצגת חלוניות התראה:

נוסיף לפרויקט חלונית התראה עבור שגיאות במחשבון.

על מנת לעשות זאת נוסיף את המתודה הבאה לcontroller של המחשבון:

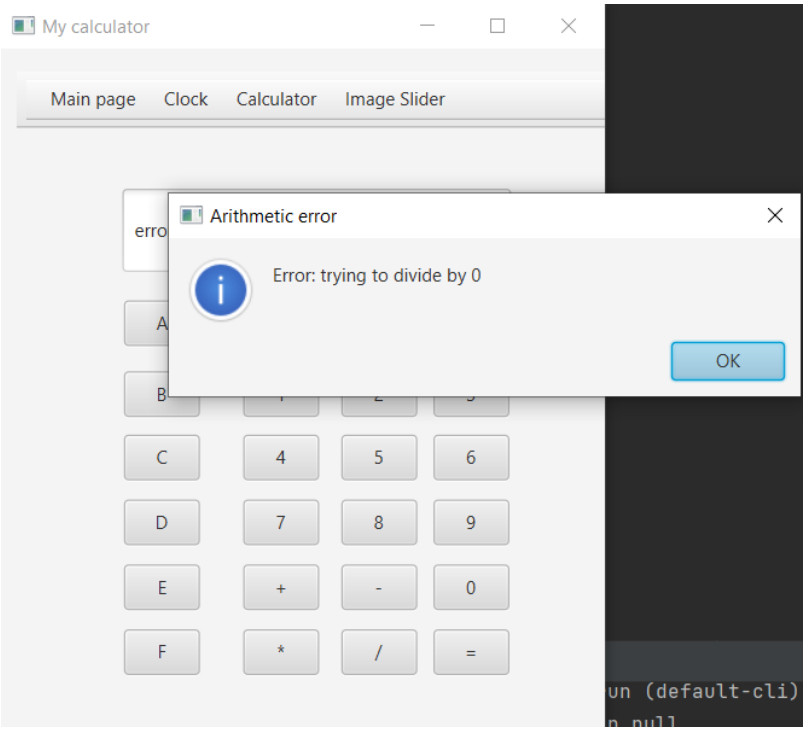
```
public void showAlert(String title, String head) {
    Platform.runLater(new Runnable() {
        public void run() {
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setTitle(title);
            alert.setHeaderText(null);
            alert.setContentText(head);
            alert.showAndWait();
        }
    });
}
```

הסבר:

Alert – תת מחלקה של מחלקת Dialog. משמשת להצגת התראות שהוגדרו מראש המספקות למשתמש מידע נחוץ.

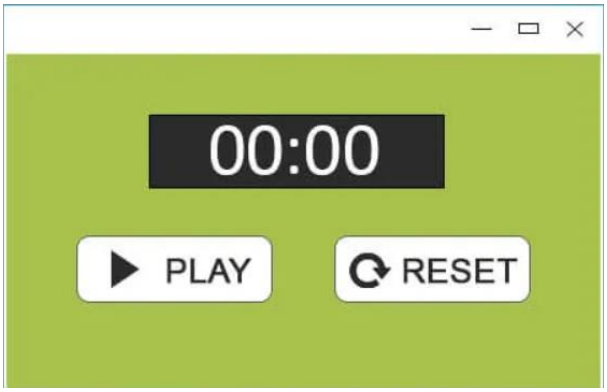
להסבר נוסף: https://www.geeksforgeeks.org/javafx-alert-with-examples/

הוסף לפרויקט שלנו התראות עבור שגיאות אפשריות שונות בשימוש במחשבון.

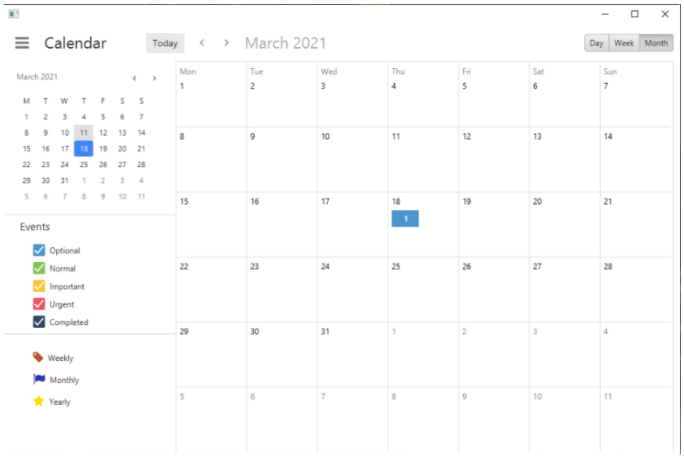


פעולות נוספות לדוגמה שנוכל להוסיף לתוכנית שלנו:

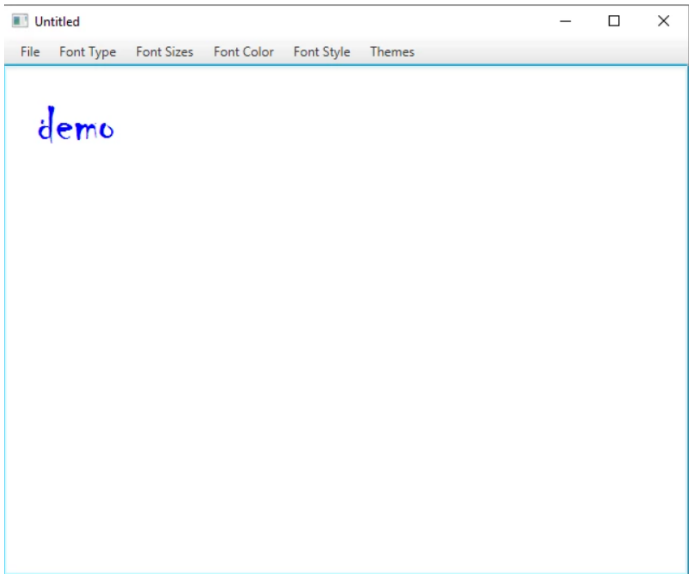
❖ טיימר - https://www.educba.com/javafx-timer/



- לוח שנה - https://github.com/JKostikiadis/JFXCalendar



- פנקס - https://github.com/akumar23/notepad-javafx



- ועוד....

ע ב ו ד ה נ ע י מ ה!