جامعة طرابلس
كلية تقنية المعلومات

# البرمجة المرئية
# Visual Programming

ITSE423

إعداد: حسن علي حسن إبراهيم
المحاضر: فاطمة علي بن الأشهر

**المحاضرة الأولى**

## المراجع وتوزيع الدرجات

- **Book**: Learn JavaFX 8_ Building Experience and Interfaces with Java 8 Apress 2015.

| البرامج | الدرجات |
|---|---|
| Java Development Kit 8 -JDK<br>NetBeans IDE 8.0 or later | ▪ Midterm Exam 30%<br>▪ Assignments 20%<br>▪ Final Exam 40%<br>▪ Lab Exam 10% |

## مواضيع المادة الدراسية

- مقدمة : ما هي JavaFX
- ما هو Node - Scene - Stage
- الأشكال ثنائية الأبعاد 2D (Line- Rectangle - Circle ...)
- التخطيط Layout Pane
- أوراق الأنماط المتتالية CSS
- الالوان والتأثيرات Colors و Effects
- تصميم الواجهات FXML
- التعامل مع الاحداث Events Handling
- تصميم الواجهات Scene Builder
- التعامل مع قاعدة البيانات Connection to DataBase
- عناصر واجهة المستخدم Controls (UI Control)
- التعامل مع المخططات Charts
- الاشكال ثلاثية الابعاد 3D Shapes
- الرسومات المتحركة Animations

## مواضيع المحاضرة

- مقدمة : ما هي **JavaFX**
- ميزات **JavaFX**
- تاريخ **JavaFX**
- تهيئة بيئة التطوير
- تجهيز بيئة العمل لتطوير البرامج بلغة جافا
- تحميل و تنصيب أدوات جافا
- تركيبة JavaFX
- الشكل العام لأي برنامج مكتوب بلغة جافا
- دورة حياة البرامج في JavaFX
- طريقة بناء البرامج في JavaFX

## مقدمة

- ما هي JavaFX؟
- JavaFXعبارة عن مجموعة من الحزم الرسومات والوسائط التي تمكن المطورين من تصميم وإنشاء واختبار وتصحيح ونشر تطبيقات تعمل باستمرار عبر منصات متنوعة.
- هي مكتبة ضخمة تم تطويرها لبناء تطبيقات تحتوي واجهة مستخدم. تمت كتابة JavaFX بلغة Java، والتي تمكنك من الاستفادة من جميع الميزات الموجودة في Java.
- باستخدام هذه المكتبة يمكنك تطوير تطبيقات تعمل على مختلف أنظمة التشغيل و مختلف الأجهزة مثل الكمبيوتر, الهاتف, الاجهزة المرئية.
- JavaFX جعلت المطورين قادرين على بناء تطبيقات قوية بسرعة و سهولة لأنها توفر تقنيات جاهزة للمبرمج يمكنه استخدامها في البرامج بدون أي تعقيد.



## ميزات JavaFX

### FXML -1

- FXML هي لغة توصيف Markup Language مبنية في الأساس على لغة XML Extensible Markup Language لغة التوصيف الموسعة تم تصميمها خصيصاً لتستخدم في بناء واجهة المستخدم للتطبيق باستخدام المكتبة JavaFX.
- توجد طريقتين لتصميم واجهة المستخدم الأولى FXML والثانية Java code.
- يمكن لمصممي الرسوم تخصيص مظهر وأسلوب البرنامج بسهولة من خلال استخدام CSS.

## ميزات JavaFX

### 2- منشئ المشهد Scene Builder

- عند تصميم واجهة المستخدم يمكنك كتابة كود الـ FXML بنفسك أو باستخدام أداة مقدمة من شركة Oracle تسمى الـ Scene Builder التي تسمح بإضافة المكونات بسهولة في النافذة.
- برنامج الـ Scene Builder هو برنامج موجود ضمن برنامج الـ Netbeans.



## ميزات JavaFX

### 3- CSS

- لغة CSS Cascading style sheet ورقة الانماط المتتالية تعتبر من أسهل اللغات على الإطلاق و هي تستخدم في تصميم صفحات الويب.
- تتيح JavaFX كتابة كود CSS لتطوير وتصميم واجهة المستخدم بسهولة.

## ميزات JavaFX

**4-** دعم اللمس المتعدد **Multitouch Support**

- **JavaFX** تدعم تقنية اللمس المتعدد بناء على قدرات الجهاز الذي تعمل عليه.
- هذا يعني في حال بناء تطبيق لجهاز يحتوي على شاشة يمكن التحكم بها بواسطة اللمس **Touch Screen** فإنك قادر على تنفيذ أكثر من أمر في نفس الوقت.
- فمثلاً إذا كان المستخدم يضع إصبع واحد على الشاشة تنفذ أمر محدد و إذا كان يضع إصبعين على الشاشة تنفذ أمر آخر أو أمرين في وقت واحد و هكذا.



## ميزات JavaFX

**5-** مكتبة الرسومات المتكاملة **Integrated Graphics library**

- **تحتوي JavaFX** على مكتبات جاهزة تدعم تقنيات رسم الـ **2D** والـ **3D**

**6-** محرك وسائط عالي الأداء **High-Performance Media Engine**

- **JavaFX** تدعم صيغ الصور و الفيديوهات و الملفات الصوتية التي يمكن وضعها في صفحات الويب.
- يمكن بشكل أساسي تشغيل أنواع الملفات التالية في واجهة المستخدم: **.MP4 – MP3 – PNG – GIF – JPG – JPS**

## تاريخ JavaFX

- بدأ مهندس البرمجيات كريس أوليفر **Chris Oliver** أثناء عمله في شركة **See Beyond Technology Corporation** بالعمل على مشروع قام بتسميته **F3** اختصاراً لـ **Form Follows Functions**.
- فكرة هذا المشروع الأساسية كانت توفير إطار جديد لبناء تطبيقات فيها واجهة مستخدم قوية فيها أغلب الأشياء التي قد يحتاجها المبرمج لبناء تطبيقاته.
- في عام **2005** أصبحت الشركة تابعة لشركة **Sun Microsystems** التي بدورها قامت بتغير إسم المكتبة التي بناها كريس إلى **JavaFX** وتولت تطوير هذه المكتبة بشكل مستمر حتى أنها أضافت عليها الكثير من التقنيات الجديدة.
- في عام **2007** تم الإعلان رسمياً عن المكتبة **JavaFX** في مؤتمر **JavaOne**الذي يعقد كل سنة لمناقشة تقنيات الجافا.
- في عام **2008** تم تضمين المكتبة **JavaFX** في برنامج **NetBeans** وبالتالي أصبح المبرمج قادراً على بناء تطبيقات **JavaFX** بكل سهولة من برنامج الـ **NetBeans**.
- في عام **2009** أصبحت شركة **Sun Microsystem** تابعة لشركة **Oracle**.
- المكتبة **JavaFx** تم تطويرها خلال عدة سنوات و تم إنشاء العديد من الإصدارات منها. إصدار المكتبة الحالي في عام **2021** هو **17 في شهر سبتمبر.**

## تهيئة بيئة التطوير

- البرنامج الذي يتم استخدامه لبناء تطبيقات **JavaFX** هما:
  - Java Development Kit 8 - JDK
  - NetBeans IDE 8.0 or later
- **إنتبه:** تم تضمين مكتبة الـ **JavaFX** في الـ **JDK 8** و الإصدارات الأحدث منه.
- إذا تم استخدام الإصدار **JDK** أقدم من الإصدار الثامن فإنك لن تجد المكتبة **JavaFX**، و بالتالي سيظهر لك خطأ تحت كل كود الذي تحاول كتابته لأن الكمبيوتر لم يستطع التعرف على الكود الذي كتبته.

## تجهيز بيئة العمل لتطوير تطبيقات بلغة جافا

- **خطوات تحميل و تثبيت JDK و NetBeans**
- في السابق كان لا بد من تثبيتهما بشكل منفصل و لكن يمكن الآن تثبتهما دفعة واحدة و بكل سهولة.
- شاهد الملف المرفق الذي يبين كيف يتم تحميل و تثبيت الإصدار الثامن من **JDK** و **NetBeans** الملائمين لحاسوبك.

- خطوات إنشاء مشروع جديد و تشغيله في برنامج NetBeans
- في الملف المرفق تم وضع خطوات إنشاء مشروع ( أي برنامج ) جديد في NetBeans.

## تجهيز بيئة العمل لتطوير تطبيقات بلغة جافا

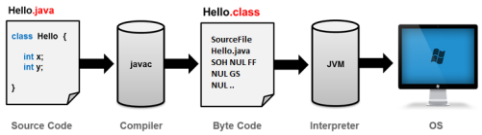
- **حل بعض المشاكل قد تتعرض لها أثناء استخدام برنامج NetBeans**
- **المشكلة الأولى:** أحياناً عندما تقوم بإنشاء مشروع جديد ثم تضغط على أيقونة تشغيل المشروع تجد أنه يظهر لك نتيجة مشروع آخر!!! بمعنى أنه يقوم بتشغيل مشروع آخر و ليس آخر مشروع قمت بكتابته.

- **المشكلة الثانية:** أحياناً بسبب العمل بسرعة تقوم إغلاق إحدى النوافذ المهمة في البرنامج. مثل النافذة التي تظهر فيها كل المشاريع التي قمت بإنشائها أو النافذة التي تظهر فيها نتيجة التشغيل.

- أطلع على الملف المرفق الذي يبين كيف يتم التعامل مع هذه المشاكل.

## تحميل و تنصيب أدوات جافا

- **طريقة عمل برنامج مكتوب بلغة جافا**
- الكود الذي تكتبه على الكمبيوتر لا يعمل بشكل مباشر بل يمر بعدة مراحل تباعاً حتى يعمل تماماً كما في الصورة التالية.
- الكود الذي يتم كتابته يسمى **Source Code** ثم يتم تحويله إلى **Byte Code** بواسطة مترجم لغة جافا **javac** وهذا المترجم يضمن أن الكود الذي تم كتابته يعمل في لغة جافا. بعدها يقوم مفسر لغة جافا **JVM** بتنفيذ الكود على بشكل يلائم نظام التشغيل الذي يستخدمه المستخدم سواء كان **Windows, Linux, MacOS**

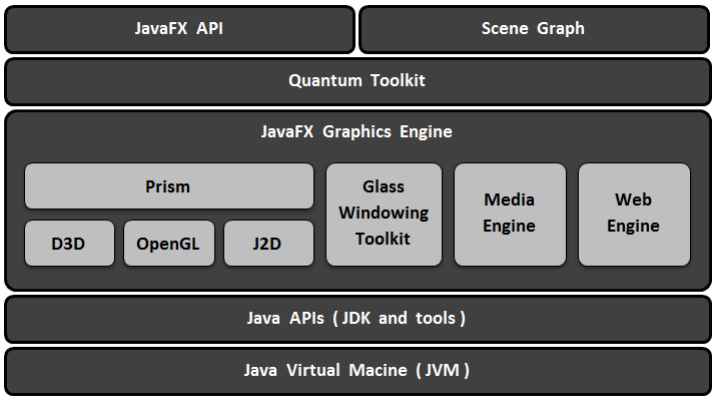



Hassan Ebrahem

15

## تركيبة JavaFX

- الصورة التالية توضح تركيبة architectural مكونات **JavaFX**




Hassan Ebrahem

17

## تركيبة JavaFX

- الـ JavaFX APIs هي الطبقة العليا وهي عبارة عن مكتبات مبنية بلغة الجافا والمتوفرة في JavaFX ومنها:
- javafx.application: تحتوي على مجموعة من الفئات المسؤولة عن دورة حياة التطبيق .
- javafx.stage: تحتوي على الحاوية الرئيسية وتستخدم لإضافة نوافذ عادية أو نوافذ لها إستعمالات مختلفة.
- javafx.animation: تحتوي على فئات المتعلقة بالرسومات المتحركة أي لإضافة مؤثرات على الأشياء.
- javafx.css: تحتوي على فئات تستخدم لتحسين تصميم واجهة المستخدم بكود CSS.

## تركيبة JavaFX

- javafx.event: تحتوي على فئات وواجهات للتعامل مع الاحداث events أي تستخدم للتعامل مع أي تفاعل قد يجريه المستخدم عند استخدام التطبيق.
- javafx.geometry: تحتوي على فئات تستخدم لإضافة أشكال هندسية ثنائية الأبعاد 2D.
- javafx.scene: تستخدم لتحديد نوع المحتوى الذي يمكن إضافته في النافذة و لإضافة محتوى جديد فيها. وتحتوي بداخلها أيضاً على المكتبات التالية:
  **canvas – chart – control – effect – image – input – layout – media – shape – text – transform – web.**
- javafx.fxml: تستخدم لربط ملفات الـ Java بملفات FXML.

## تركيبة JavaFX

- **Scene Graph**
- هو نقطة البداية لبناء تطبيق واجهة المستخدم الرسومية GUI. ويتم بنائها باستخدام API.
- الـ **Scene Graph** عبارة عن مجموعة من الـعقد **Nodes** تم إضافتها بشكل هرمي حتى تكون محتوى النافذة.
- تتضمن أنواع العقد في **Scene Graph** مجموعة عناصر التحكم في واجهة المستخدم UI مثل buttons, text, shapes, images, media
- **Qunatum Toolkit**
- الـ **Qunatum Toolkit** عبارة عن برنامج يجعل جميع الـ **Graphics Engine** متاحة للإستخدام للتطبيقات المبنية بواسطة **JavaFX**.

## تركيبة JavaFX

- **JavaFX Graphics Engine**
- الـ **JavaFX Graphics Engine** عبارة عن البرامج التي تعمل لتولد صورة التطبيق على الشاشة ويحتوي على :
- الـ **Prism** مهمته عرض الأشكال الهندسية التي قد يتم إضافتها في التطبيق سواء كانت **2D** أو **3D** باستخدام البرامج الخاصة بالرسومات مثل Direct X9 للويندوز او OpenGL للماك أو يونكس.
- الـ **Glass Windowing Toolkit** مهمته عرض النوافذ بشكل يتوافق مع نظام التشغيل بالإضافة إلى إدارة الأحداث **Event Queues** مجموعة من الأدوات المسؤولة أيضًا عن إدارة قوائم انتظار الأحداث.
- الـ **Media Engine** مهمته تشغيل الفيديو والملفات الصوتية التي قد يتم إضافتها في التطبيق وتدعم مجموعة صيغ منها FLV – WAV – MP3.
- الـ **Web Engine** مهمته عرض صفحات الويب مثل HTML وتأثير CSS التي قد يتم إضافتها في التطبيق وكذلك التعديل عليها واعادة تحميلها.

## تركيبة JavaFX

- **JDK and Tools**
- هو المكتبات و الأدوات الموجودة في لغة **Java**. حيث أن نظام التشغيل الذي تستخدمه لا يتعرف على لغة جافا وبالتالي نحتاج تحميل و تنصيب أدوات تطوير جافا **Java Development Kit** .
- الـ **JDK** يتضمن أشياء كثيرة مثل:
- الفئات **classes** الجاهزة في جافا.
- الشروحات للفئات والدوال الجاهزة التي تظهر لك أثناء كتابتك للكود والتي تسمى **javadoc**.
- مترجم لغة جافا **javac**.
- مشغل لغة جافا **Java Runtime Environment** الذي يختصر بـ **JRE** والذي بدوره يحتوي على الـ **JVM**.
- بدون الـ **JDK** لن يستطيع نظام التشغيل التعرف على لغة جافا. و بالتالي لن يستطيع تشغيل أي كود أو حتى برنامج جاهز مكتوب بلغة جافا.
- **Java Virtual Machine**
- الـ **JVM**هو البرنامج الذي يقوم بتشغيل البرامج المكتوبة بلغة **Java**.

## الشكل العام لأي تطبيق مكتوب بلغة جافا

- الفئة الأساسية في المشروع يجب أن يكون شكلها كالتالي:

```
public class هنا يوضع إسم الكلاس {

    public static void main(String[] args) {

        هنا يجب أن تضع الأوامر التي ستنفذ عند تشغيل البرنامج

    }

}
```

- في العادة الفئة Class الأساسية في المشروع يتم تسميتها **Main**.
- الكود الذي يتنفذ مباشرةً عند تشغيل التطبيق هو الكود الذي نكتبه بداخل حدود الدالة **main()**.

## الشكل العام لأي برنامج مكتوب بلغة جافا

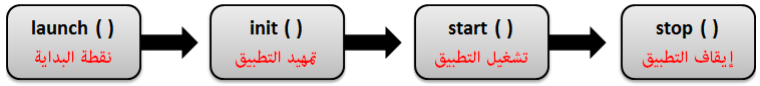
- في المثال التالي قمنا بإنشاء برنامج إسمه **Main** موجود بداخله دالة إسمها **main** مهمتها فقط طباعة الجملة **Hello World!** عند التشغيل.

```
1.  public class Main {
2.
3.      public static void main(String[] args) {
4.
5.          System.out.println("Hello World!");
6.
7.      }
8.
9.  }
```



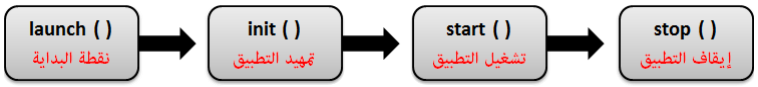
## دورة حياة التطبيق في JavaFX

- الفئة **Application** الموجود في المسار **javafx.application.Application** تحتوي على الدوال **launch() – init() – start() – stop()**
- هذه الدوال تمثل دورة حياة أي التطبيق تم بناؤه في **JavaFX** وهي تستدعى بشكل تلقائي كما في الصورة التالية.



## دورة حياة التطبيق في JavaFX

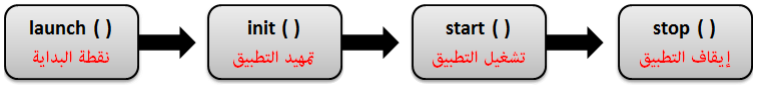
- دورة حياة تطبيق JavaFX تشتمل على ثلاث دوال لدورة الحياة ، وهي:
- Start(): هي دالة نقطة الإدخال حيث تتم كتابة كود المراد تنفيذه بداخلها، والتي تستخدم لإنشاء تطبيق JavaFX.
- Init(): دالة فارغة يمكن تجاوزها، ولا يمكن إنشاء Stage أو Scene بداخلها.
- Stop(): دالة فارغة يمكن تجاوزها، فيها يمكن كتابة كود لإيقاف التطبيق.




Hassan Ebrahem

26

## طريقة بناء التطبيق في JavaFX




- يقوم مشغل برامج جافا **JVM** بتنفيذ الدوال بشكل تلقائي و بنفس الترتيب الذي في الصورة أعلاه.
- الدوال الموجودة في الفئة **Application** يتم استدعائها بشكل تلقائي في كل برنامج مبني بواسطة **JavaFX**.
  - **public void launch()**
- تعتبر نقطة البداية في التطبيق حيث أنها أول دالة تنفذ عند تشغيل التطبيق. عندما يتم تنفيذها تقوم فقط باستدعاء الدالة **init()**.
- **ملاحظة:** إذا لم تقم بتعريف الدالة **main()** في البرنامج سيتم إستدعاء الدالة **launch()** بشكل تلقائي عند تشغيل التطبيق.
- أما إذا تم تعريف الدالة **main()** فسيكون عليك إستدعاء **launch()** بداخلها أثناء التنفيذ.

Hassan Ebrahem

28

## طريقة بناء التطبيق في JavaFX

- public void init() throws Exception

- تعتبر نقطة تمهيد البرنامج حيث أنه يمكن أن يتم لها **Override** في حال أردت أن يتم تنفيذ أي كود في لحظة إنشاء التطبيق.

- **إنتبه:** لا يمكنك كتابة كود إنشاء الواجهة الرئيسية للتطبيق فيها.

- يظهر الإستثناء Exception في حال حدث أي خطأ عندما يتم إستدعاءها.

## طريقة بناء التطبيق في JavaFX

- public void start(Stage primaryStage) throws Exception
  - تعتبر نقطة تشغيل التطبيق يجب أن يتم لها **Override**.
  - هذه الدالة تعتبر المكان الذي يجب أن يتم فيه كتابة كود إنشاء الواجهة الرئيسية للبرنامج. وهي تستدعي بعد أن يتم تنفيذها الدالة **init()**.
  - البارامتر **primaryStage** يمثل الواجهة الرئيسية للتطبيق والتي يمكنك إظهارها بواسطة الدالة **show()**.

- تظهر الإستثناء Exception في حال حدث أي خطأ عندما يتم إستدعاءها.

## طريقة بناء التطبيق في JavaFX

- public void stop()

- تعتبر نقطة النهاية في التطبيق حيث أنه يمكن أن يتم لها **Override** في حال أردت أن يتم تنفيذ أي كود عند الخروج من التطبيق, مثل إغلاق أي شيء كان مفتوحاً من داخل التطبيق, أو حفظ أي عملية قام بها المستخدم قبل الخروج من التطبيق أو اظهار رسالة معينة إلخ.

- تظهر الإستثناء Exception في حال حدث أي خطأ عندما يتم إستدعاءها.



## طريقة بناء التطبيق في JavaFX

- **تذكير** في أي تطبيق **JavaFX** يتم بنائه يجب أن يتم **Override** للدالة **start()** ومن غير الضروري أن يتم **Override** للدوال **stop()** و **init()**.

- المثال التالي يبين كيف يتم إستدعاء الدالة **launch()** من داخل الدالة **main()** والذي يؤدي ذلك إلى إستدعاء الدالة **init()** ثم الدالة **start()** عند تشغيل التطبيق والدالة **stop()** عند الخروج منه.

- **إنتبه:** إذا تم تعريف الدالة **main()** في التطبيق في هذه الحالة يجب إستدعاء الدالة **launch()** بداخلها حتى يتم إستدعاء الدوال **init()** و **start()** و **stop()**.

# مثال على طريقة بناء التطبيق في JavaFX

Main.java

```java
import javafx.application.Application;
import javafx.stage.Stage;

public class Main extends Application {

    //هذه أول دالة سيتم إستدعاءها عند تشغيل البرنامج
    @Override
    public void init() {
        System.out.println("init() method is called");
    }

    // init() سيتم استدعاء هذه الدالة بشكل تلقائي في حال بعد أن يتم إستدعاء الدالة
    @Override
    public void start(Stage primaryStage) {
        System.out.println("start() method is called");
        primaryStage.show();
    }

    // سيتم استدعاء هذه الدالة عند الخروج من التطبيق
    @Override
    public void stop() {
        System.out.println("stop() method is called");
    }

}
```

```
init() method is called
start() method is called
stop() method is called
```

Main.java

```java
import javafx.application.Application;
import javafx.stage.Stage;

public class Main extends Application {

    //هذه أول دالة سيتم إستدعاءها عند تشغيل البرنامج
    // launch() و التي ستقوم بدورها باستدعاء الدالة init() يتم استدعاء هذه الدالة تلقائيا باستدعاء الدالة
    public static void main(String[] args) {
        System.out.println("main() method is called");
        launch();
    }

    @Override
    public void init() {
        System.out.println("init() method is called");
    }

    // init() سيتم استدعاء هذه الدالة بشكل تلقائي في حال تم استدعاء الدالة
    @Override
    public void start(Stage primaryStage) {
        System.out.println("start() method is called");
        primaryStage.show();
    }

    // سيتم استدعاء هذه الدالة عند الخروج من التطبيق
    @Override
    public void stop() {
        System.out.println("stop() method is called");
    }

}
```

```
main() method is called
init() method is called
start() method is called
stop() method is called
```

# طريقة بناء التطبيق في JavaFX

إذا كنت تستخدم بيئة تطوير **IDE** لا تحتوي على جميع الأدوات المتوفرة في **JavaFX** وتحديداً الأداة **JavaFX Launcher** التي تتولى تشغيل برامج **JavaFX** من داخل بيئة التطوير فإنك مجبر على وضع الدالة **()main** في البرنامج حتى لا يظهر لك خطأ مفاده أنه لا يمكن تشغيل البرنامج لأنه لم يتم العثور على الدالة **()main**.

يجب وضع الدالة **()main** في جميع البرامج حتى تعمل جميعها مهما كانت بيئة التطوير التي تعتمد عليها و بدون الحاجة لإجراء أي تعديل على الكود.

## ملخص المحاضرة

- في هذه المحاضرة تحدثنا على JavaFX وبينا بعض مميزاتها وكيف يتم تهيئة بيئة التطوير IDE التي سيتم العمل عليها وتجهيزها، وتم توضيح تركيبة JavaFX وكذلك دورة حياة البرنامج وطريقة بنائه.



## نهاية المحاضرة