

## JavaFX

<https://docs.oracle.com/javase/8/javase-clienttechnologies.htm>

### Aplicații propuse

<https://docs.oracle.com/javase/8/javafx/get-started-tutorial/animation.htm>

<https://docs.oracle.com/javase/8/javafx/get-started-tutorial/form.htm>

<https://docs.oracle.com/javase/8/javafx/get-started-tutorial/css.htm>

<https://www.jetbrains.com/help/idea/javafx.html#run>

<https://www.tutorialspoint.com/javafx/index.htm>

JavaFX este un set de pachete grafice și media care permite dezvoltatorilor să proiecteze, să creeze, să testeze, să depuneze și să implementeze aplicații client care funcționează **constant pe diverse platforme**.

Aplicațiile dezvoltate folosind **JavaFX** pot rula pe diverse dispozitive, cum ar fi computere desktop, telefoane mobile, televizoare, tablete etc.

Pentru a dezvolta **aplicații GUI** folosind limbajul de programare Java, programatorii se bazează pe biblioteci, cum ar fi **setul AWT** și **Swing**. **După apariția JavaFX, programatorii Java pot dezvolta aplicații GUI în mod eficient cu conținut bogat.**

**JavaFX oferă un set bogat de API-uri grafice și media.**

### JavaFX Applications

Aplicațiile JavaFX pot utiliza biblioteci Java API pentru a accesa capacitățile sistemului nativ și pentru a se conecta la aplicații middleware bazate pe server.

Aspectul aplicațiilor JavaFX poate fi personalizat. Foile de stil în cascadă (CSS) separă aspectul și stilul de implementare, astfel încât dezvoltatorii să se poată concentra pe scriere eficientă a codului. Designerii grafici pot personaliza cu ușurință aspectul și stilul aplicației prin intermediul CSS. Dacă aveți un fundal în design web sau dacă doriți să separați interfața cu utilizatorul (UI) și logica back-end, atunci puteți dezvolta aspectele de prezentare ale interfeței de utilizare în limbajul de scripting **FXML** și puteți utiliza codul Java pentru aplicație logică. Dacă preferați să proiectați interfețe de utilizare fără a scrie cod, atunci utilizați **JavaFX Scene Builder**. Pe măsură ce proiectați interfața de utilizare, **Scene Builder** creează un marcaj FXML care poate fi portat într-un mediu de dezvoltare integrat (IDE).

### Disponibilitate

JavaFX 2.2 și versiunile ulterioare sunt complet integrate cu Java SE 7 Runtime Environment (JRE) și Java Development Kit (JDK). Deoarece JDK este disponibil pentru toate platformele desktop majore (Windows, Mac OS X și Linux), aplicațiile JavaFX compilate încapand cu JDK 7 **rulează și pe toate platformele desktop**.

Compatibilitatea multiplatformă permite o experiență de rulare consistentă pentru dezvoltatorii și utilizatorii de aplicații JavaFX.

**Pe pagina de descărcare JDK, puteți obține un fișier zip cu exemple de aplicații JavaFX.** Exemplele de aplicații oferă multe exemple de cod și fragmente care arată prin exemplu cum să scrieți aplicații JavaFX.

### Caracteristici

JavaFX este o bibliotecă Java care constă din clase și interfețe care sunt scrise în cod nativ Java. API-urile sunt concepute pentru a fi o alternativă prietenoasă la limbajele Java Virtual Machine (Java VM), precum JRuby și Scala.

**FXML și Scene Builder.** FXML este un limbaj de marcare declarativ bazat pe XML pentru construirea unei interfețe cu utilizatorul aplicației JavaFX. Un designer poate codifica în FXML sau poate folosi **JavaFX Scene Builder** pentru a proiecta interactiv interfața grafică cu utilizatorul (GUI). Scene Builder generează marcaj FXML care poate fi portat într-un IDE unde un dezvoltator poate adăuga logica de afaceri.

**WebView.** O componentă web care utilizează tehnologia WebKitHTML pentru a face posibilă încorporarea paginilor web într-o aplicație JavaFX. JavaScript care rulează în WebView poate apela API-uri Java, iar API-urile Java pot apela JavaScript care rulează în WebView.

**Interoperabilitate swing.** Aplicațiile Swing existente pot fi actualizate cu noi caracteristici JavaFX, cum ar fi redarea conținutului media bogat în grafică și conținut Web încorporat.

**Controale UI și CSS încorporate.** JavaFX oferă toate controalele majore ale UI necesare pentru a dezvolta o aplicație cu funcții complete. Componentele pot fi personalizate cu tehnologii Web standard, cum ar fi CSS.

**API-ul Canvas.** API-ul Canvas permite desenarea direct într-o zonă a scenei JavaFX care constă dintr-un element grafic (nod).

**Suport multitouch.** JavaFX oferă suport pentru operațiuni multitouch, pe baza capacităților platformei de bază.

**Hardware-accelerated graphics pipeline.** Grafica JavaFX se bazează pe conducta de randare a graficelor (Prism). JavaFX oferă o grafică netedă care se redă rapid prin **Prism** atunci când este utilizat cu o placă grafică sau o unitate de procesare grafică (GPU) acceptată. Dacă un sistem nu dispune de unul dintre GPU-urile recomandate acceptate de JavaFX, atunci Prism folosește implicit stiva de software Java 2D.

**Motor media de înaltă performanță.** Conducta media acceptă redarea conținutului multimedia web. Oferă un cadru media stabil, cu latență scăzută, care se bazează pe cadrul multimedia GStreamer.




**Model autonom de implementare a aplicației.** Pachetele de aplicații autonome au toate resursele aplicației și o copie privată a runtime-urilor Java și JavaFX. Sunt distribuite ca pachete native instalabile și oferă aceeași experiență de instalare și lansare ca și aplicațiile native pentru acel sistem de operare. Consultați documentul Implementarea aplicațiilor JavaFX.

### Ce pot construi cu JavaFX?

Cu JavaFX, puteți construi multe tipuri de aplicații care sunt implementate pe mai multe platforme și afișează informații într-o interfață de utilizator modernă de înaltă performanță, care include audio, video, grafică și animație.

Tabelul 1 prezintă imagini cu câteva dintre exemplele de aplicații JavaFX incluse în versiunea JavaFX 2.2.n.

Table 1 Sample JavaFX Applications

Sample Application	Description
 <a href="#">Description of the illustration ensemble-small.gif</a>	<p>JavaFX Ensemble</p> <p>Ensemble provides a gallery of samples that demonstrate various JavaFX features, such as animation, charts, and controls. You can view the running sample, read a description, copy the source code, and follow links to the <a href="#">API documentation</a>.</p>
 <a href="#">Description of the illustration data-app-small.gif</a>	<p>Sales Dashboard (DataApp)</p> <p>DataApp is a client-server application for a fictional global automobile company called Henley Car Sales. Automobile sales are simulated on an EJB server using JavaDB, and the data is available via Derby and a RESTful web service. The client demonstrates a variety of data presentations by using a mix of FXML and JavaFX.</p> <p>Note: The DataApp sample has multiple NetBeans projects and cannot be run without some additional setup. The DataAppReadme.html file and the NetBeans project file are in the src\DataApp directory.</p>
 <a href="#">Description of the illustration swing-interop.gif</a>	<p>SwingInterop Sample</p> <p>This Swing application shows how Swing and JavaFX can be combined. It uses JavaFX components to implement a chart and a simple browser. A JTable Swing component is used for the table.</p>

### JavaFX și JavaFX Scene Builder 2.0

JavaFX este o bibliotecă java ce permite utilizatorilor construirea unor aplicații versatile ce pot rula pe mai multe dispozitive și îmbogățește elementele disponibile pentru dezvoltarea unor aplicații cu o interfață grafică(GUI) față de cele prezente anterior în AWT și Swing, având în plan înlocuirea celor din urmă.

JavaFX 13 – septembrie 2019

- JavaFX 14 – martie 2020

Un aspect foarte important este că API-ul JavaFX definește în **pachetul javafx** clase container pentru ordonarea elementelor. Acestea sunt:

- **AnchorPane** – se pot defini noduri în partea de jos, sus, stînga dreapta și mijlocul containerului
- **BorderPane** – ordonează noduri în jos, sus, stînga dreapta și mijlocul containerului
- **FlowPane** – ordonează elementele vertical sau orizontal în funcție de preferințele definite
- **GridPane** – ordonează elementele într-un tabel definit de programator, numărul de linii și coloane fiind definit de acesta

- HBox – aranjează elementele orizontal pe o singură linie
- VBox – aranjează elementul vertical pe o singură coloană
- Pane – un container simplu definit de utilizator după dimensiunile dorite de acesta
- SplitPane – un container format din două elemente pane, acestea pot fi separate pe verticală sau pe orizontală.

O altă parte importantă a JavaFX sunt controalele disponibile Label, Button, CheckBox, RadioButton, Separator, TextArea, TextField și PasswordField.

Un element nou în aplicațiile GUI este apariția FXML un limbaj de marcare declarativ ce are rolul de a defini o interfață. Totodată apariția aplicației JavaFX Scene Builder ce poate fi integrată în IDE oferă o interfață în care se pot proiecta cu ușurință scenele dorite cu ajutorul opțiunii de drag and drop al elementelor și posibilitatea de a le atribui id-uri pentru stilul CSS și pentru utilizarea acestora în cadrul controlerului Java.

Pentru a dezvolta o aplicație JavaFX trebuie dezvoltat un graf de scene, o structură arborescentă de noduri ce conține elementele interfeței grafice. Nodul rădăcină poate avea doar rolul de părinte și poate avea copii, însă copii acestuia au un părinte și pot avea la rândul lor mai mulți copii.

Structura generală a unei aplicații JavaFX este următoarea: în pachetul său java de bază există o clasă principală ce este derivată din `javafx.application.Application`, un fișier FXML și o clasă controler atașată fișierului FXML ce tratează evenimentele petrecute în scena respectivă.

Printre avantajele dezvoltării unei aplicații JavaFX FXML se numără accesul cu ajutorul importurilor la toate clasele Java din JDK-ul de bază, astfel în tratarea evenimentelor există o multitudine de metode ce pot fi folosite în funcție de nevoile dezvoltatorului.

JavaFX Scene Builder este o aplicație ce se poate integra în mediul de lucru al dezvoltatorului (NetBeans, Eclipse, IntelliJ) și permite dezvoltarea unei interfețe de utilizator fără a fi nevoie ca dezvoltatorul să scrie totul cu ajutorul unei interfețe drag and drop. Aceasta poate să deschidă fișiere FXML chiar din spațiul de lucru odată integrată sau poate crea fișiere noi FXML și apoi poate face legătura între controler și acestea din urmă. Astfel JavaFX Scene Builder permite crearea cu ușurință a unei interfețe oferind dezvoltatorului opțiunea de a edita anumite elemente manual.

[https://docs.oracle.com/javase/8/javafx/get-started-tutorial/fxml\\_tutorial.htm](https://docs.oracle.com/javase/8/javafx/get-started-tutorial/fxml_tutorial.htm)

- **Scene Builder** - JavaFX furnizează o aplicație numită Scene Builder. La integrarea acestei aplicații în IDE-uri precum Eclipse și NetBeans, utilizatorii pot accesa o interfață de proiectare drag and drop, care este utilizată pentru a dezvolta aplicații FXML (la fel ca Swing Drag & Drop și DreamWeaver Applications).
- <https://docs.oracle.com/javase/8/javafx/get-started-tutorial/css.htm>

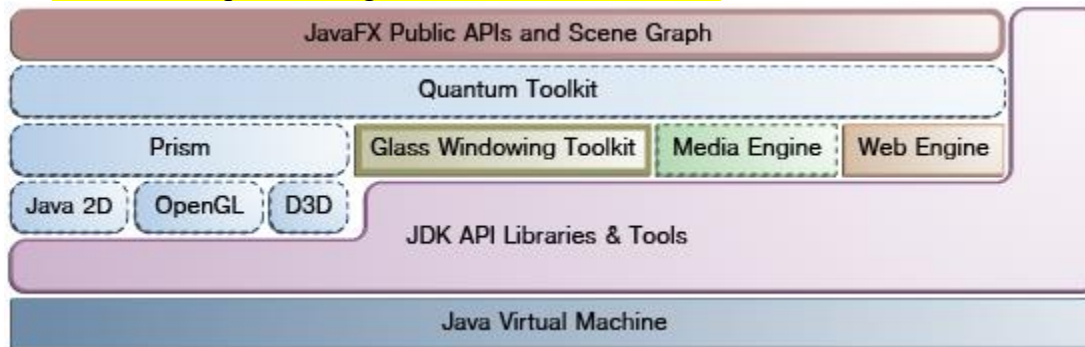
Aplicațiile JavaFX pot fi rulate:

- direct pe o anumită mașină, unde se află “instalată” (accesând fișierul .jar care conține programul în sine) - o astfel de opțiune este preferată în cazul în care aplicația nu are nevoie de acces Internet sau mașina pe care rulează nu dispune de o astfel de conexiune;
- prin intermediul unui navigator (eng. browser), fiind integrată în cadrul unei pagini Internet, interacțiunea cu elementele paginii Internet putând fi realizată prin intermediul limbajului JavaScript;
- prin descărcare de la o anumită locație (dacă aplicația JavaFX este găzduită pe un server web), rulând apoi pe mașina respectivă (modul WebStart);
- ca program autonom, folosind copii private ale mediilor de rulare Java și JavaFX.

### Arhitectura JavaFX

Platforma JavaFX dispune de o arhitectură bazată pe o stivă de componente (transparentă pentru programator), constituind motorul ce rulează codul propriu-zis (Quantum Toolkit). Acesta cuprinde:

1. un motor grafic performant (Prism)
2. un sistem de ferestre, de dimensiuni mici (Glass)
3. un motor pentru redarea conținutului multimedia,
4. un motor pentru integrarea conținutului Internet.



### Graful de Scene

Implementarea unei aplicații JavaFX implică proiectarea și dezvoltarea unui graf de scenă (*eng.* Scene Graph), structură ierarhică de noduri ce conține elementele vizuale ale interfeței grafice cu utilizatorul, care poate trata diferite evenimente legate de acestea și care poate fi redată.

Un element din graful de scenă (= un nod) este identificat în mod unic, fiind caracterizat printr-o clasă de stil și un volum care îl delimitează. Fiecare nod are un părinte (cu excepția nodului rădăcină), putând avea nici unul, unul sau mai mulți copii. De asemenea, pentru un astfel de element pot fi definite efecte (estompări sau umbre), opacitate, transformări, mecanisme de tratare a diferitelor evenimente (care vizează interacțiunea cu utilizatorul) precum și starea aplicației.

Spre deosebire de Swing sau AWT (Abstract Window Toolkit), **JavaFX conține pe lângă mecanisme de dispunere a conținutului, controale, imagini sau obiecte multimedia și primitive pentru elemente grafice (ca fi texte sau figuri geometice cu care se pot crea animații, folosind metodele puse la dispoziție de API-urile javafx.animation).**

API-ul `javafx.scene` permite construirea următoarelor conținuturi:

- **noduri:** forme 2D și 3D, imagini, conținut multimedia și conținut Internet, text, controale pentru interacțiunea cu utilizatorul, grafice, containere;
- **stări:** transformări (poziționări și orientări ale nodurilor), efecte vizuale;
- **efecte:** obiecte care modifică aspectul nodurilor (mecanisme de estompare, umbre, reglarea culorilor).

### Mecanisme de Dispunere a Conținutului

Controalele din graful de scenă pot fi grupate în containere sau panouri în mod flexibil, folosind mai multe mecanisme de dispunere a conținutului (*eng.* layout).

API-ul JavaFX definește mai multe clase de tip container pentru dispunerea elementelor, în pachetul `javafx.scene.layout`:

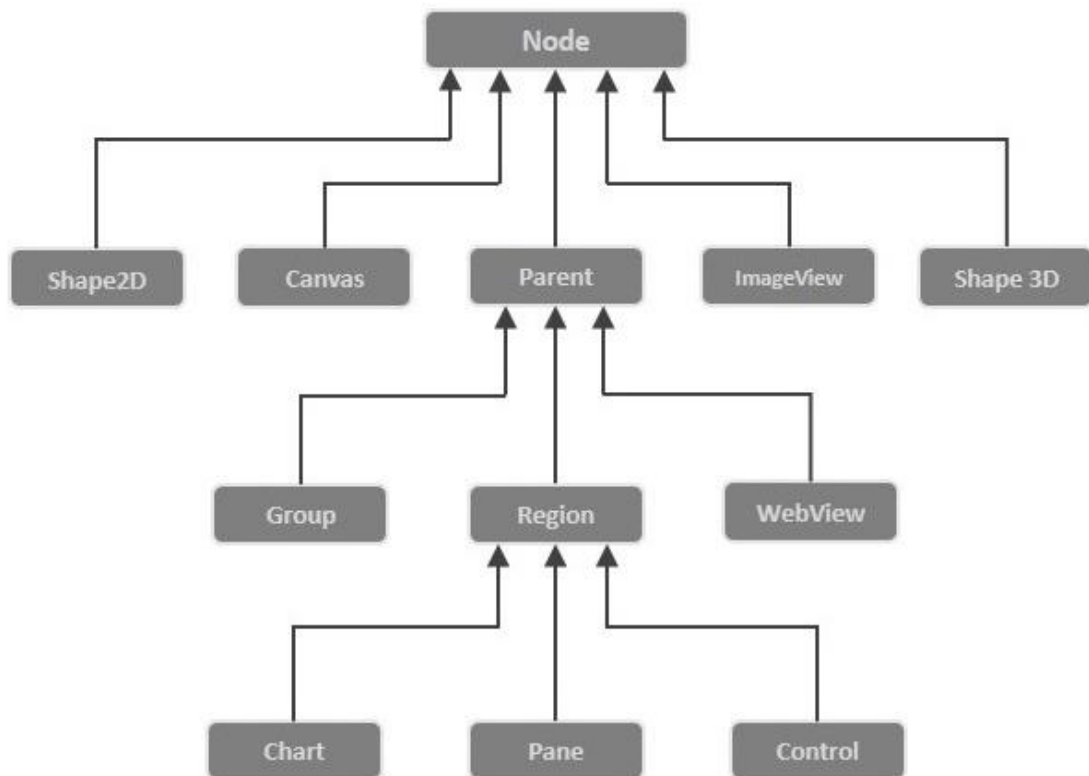
- **BorderPane** dispune nodurile conținute în regiunile de sus, jos, dreapta, stânga sau centru;
- **HBox** își aranjează conținutul orizontal pe un singur rând;
- **VBox** își aranjează conținutul vertical pe o singură coloană;
- **StackPane** utilizează o stivă de noduri afișând elementele unele peste altele, din spate către față;
- **GridPane** permite utilizatorului să își definească un tabel (format din rânduri și coloane) în care să poată fi încadrate elementele conținute;
- **FlowPane** dispune elementele fie orizontal, fie vertical, în funcție de limitele specificate de programator (lungime pentru dispunere orizontală, respectiv înălțime pentru dispunere verticală);

- **TilePane** plasează nodurile conținute în celule de dimensiuni uniforme;
- **AnchorPane** oferă programatorilor posibilitatea de a defini noduri ancoră (referință) în funcție de colțurile de jos / sus, din stânga / dreapta sau raportat la centrul containerului sau panoului.

Diferitele moduri de dispunere pot fi imbricate în cadrul unei aplicații JavaFX pentru a se obține funcționalitatea dorită.

**WebView** - Acest nod gestionează motorul web și afișează conținutul acestuia.

## Ierarhia claselor de noduri din JavaFX



O scenă JavaFX este reprezentată de clasa **Scene** a pachetului **javafx.scene**. Puteți crea o Scenă prin instanțierea acestei clase așa cum se arată în următorul bloc cod.

```
Scene scene = new Scene(root);  
Scene scene = new Scene(root, 600, 300);
```

În timpul instanțierii, este obligatoriu să treceți obiectul rădăcină către constructorul clasei scene.

Clasa Stage a pachetului **javafx.stage** este containerul oricărei aplicații JavaFX și oferă o fereastră pentru aplicație.

Un obiect din clasa Stage a pachetului **javafx.stage** este transmis ca parametru al metodei **start()** a clasei **Application**.

Folosind acest obiect, puteți efectua diverse operații pe scenă. Ca de exemplu:

Setați titlul scenei folosind metoda **setTitle()**.

Atașați obiectul **scene** la scena folosind metoda **setScene()**.

Afișați conținutul scenei folosind metoda **show()**

```
//Setting the title to Stage.  
primaryStage.setTitle("Sample application");
```

```
//Setting the scene to Stage
```

```
primaryStage.setScene(scene);
```

```
//Displaying the stage  
primaryStage.show();
```

### Ciclul de viață al aplicației JavaFX

Clasa JavaFX Application are trei metode de ciclu de viață, care sunt :

**start()** - Metoda punctului de intrare în care urmează să fie scris codul grafic JavaFX.

**stop()** - O metodă goală care poate fi suprascrisă, aici puteți scrie logica pentru a opri aplicația.

**init()** - O metodă goală care poate fi suprascrisă, dar nu puteți crea stage or scene în această metodă.

În plus față de acestea, oferă o metodă statică numită **launch()** pentru a lansa aplicația JavaFX.

Deoarece metoda **launch()** este statică, trebuie să o apelați dintr-un context static (principal în general). Ori de câte ori se lansează o aplicație JavaFX, vor fi efectuate următoarele acțiuni (în aceeași ordine).

Este creată o instanță a clasei de aplicație.

Metoda **Init()** este apelată.

Metoda **start()** este apelată.

Lansatorul așteaptă ca aplicația să se termine și apelează metoda **stop()**.

### Închiderea aplicației JavaFX

Când ultima fereastră a aplicației este închisă, aplicația JavaFX este terminată implicit. Puteți dezactiva acest comportament trecând valoarea booleană „False” metodei statice **setImplicitExit()**.

Puteți închide o aplicație JavaFX în mod explicit folosind metodele **Platform.exit()** sau **System.exit(int)**.



## EX1

```
/**
 *
 * @author 5info
 */
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class EX1 extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception {
        //creating a Group object
        Group group = new Group();

        //.. Crearea unei scene
        Scene scene = new Scene(group ,600, 300);

        //setting color to the scene
        scene.setFill(Color.BROWN);

        //Setting the title to Stage.
        primaryStage.setTitle("Aplicatia 1 JavaFX");

        //Adding the scene to Stage
        primaryStage.setScene(scene);

        //Displaying the contents of the stage
        primaryStage.show();
    }
    public static void main(String args[]){
        launch(args);
    }
}
```

## Ex1\_1

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Group;
import javafx.scene.control.Button;
import javafx.scene.Scene;

public class Ex1_1 extends Application {

    public static void main(String[] args) {

        Application.launch(args);
    }
}
```

```

public void start(Stage primaryStage) {

    primaryStage.setTitle(" JavaFx Exemplan 1_1");

    Button btn = new Button("JavaFx Button");
    btn.setLayoutX(100);
    btn.setLayoutY(100);

    Button btn2 = new Button("JavaFx Button 2");
    btn2.setLayoutX(300);
    btn2.setLayoutY(100);

    Group group = new Group();
    group.getChildren().add(btn);
    group.getChildren().add(btn2);

    Scene scene = new Scene(group, 700, 200);
    primaryStage.setScene(scene);
    primaryStage.show();
}
}

```

Hello.java

```

/**
 *
 * @author 5info
 */
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class Hello extends Application {

    @Override
    public void start(Stage primaryStage) {
        Button btn = new Button();
        btn.setText("Say 'Hello World'");
        btn.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
                System.out.println("Hello World!");
            }
        });

        StackPane root = new StackPane();
    }
}

```

```

root.getChildren().add(btn);

Scene scene = new Scene(root, 300, 250);

primaryStage.setTitle("Hello World!");
primaryStage.setScene(scene);
primaryStage.show();
}

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

```

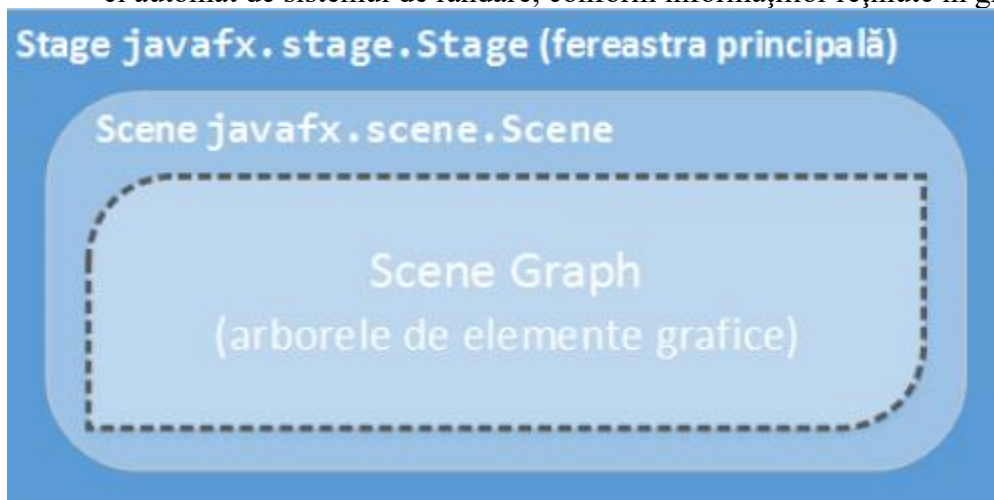
### Structura Generală a unei Aplicații JavaFX

O aplicație JavaFX trebuie să aibă o clasă principală, derivată din `javafx.application.Application`.

Metoda `start()` a acesteia trebuie să fie suprascrisă, primind ca unic parametru un obiect de tip `javafx.stage.Stage`, fereastra principală a aplicației, pentru care se indică titlul, scena și vizibilitatea. Această metodă este apelată în mod automat la execuția clasei principale.

Containerul interfeței cu utilizatorul se exprimă prin clase de tip:

- `javafx.stage.Stage` - fereastra principală;
- `javafx.scene.Scene` - are asociat graful de noduri menținând un model intern al elementelor grafice din cadrul aplicației; la fiecare moment de timp, acesta știe ce obiecte să afișeze, care zone ale ecranului trebuie actualizate și cum să realizeze acest proces într-un mod eficient; metodele de desenare nu sunt apelate manual de utilizator, ci automat de sistemul de randare, conform informațiilor reținute în graful de noduri.

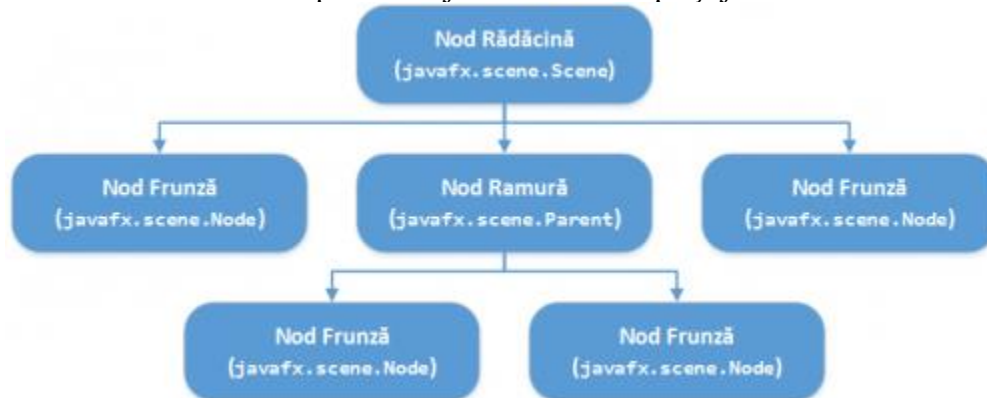


Fiecare element din cadrul unei scene este exprimat sub forma **unui nod dintr-un graf (arbore)** în care elementul rădăcină este reprezentat de containerul din care acestea fac parte.

Pachetul `javafx.scene` definește clase aferente celor trei tipuri de noduri care pot apărea în graful (arborele) de scene:

- Scene: containerul de bază pentru întregul conținut (nod rădăcină);
- Parent: clasă abstractă pentru nodurile ce pot avea copii (nod ramură); implementări ale acestei clase sunt Control, Group, Region și WebView;

- Node: clasă abstractă pentru toate elementele grafice, folosită în special pentru nodurile care nu pot avea copii (nod frunză); clasele ce pot fi folosite în acest scop sunt definite mai cu seamă în pachetele `javafx.scene.shape` și `javafx.scene.text`.



Aceste clase de bază definesc funcționalități importante care vor fi moștenite de subclasele lor, inclusiv ordinea de desenare, vizibilitatea, compunerea transformărilor sau suportul pentru stiluri CSS.

### Componente ale Interfeței cu Utilizatorul Controale Grafice

#### Label

- `public void setText(String text)`
- `public void setGraphic(Node graphic)`
- `setTextAlignment()`
- `setTextFill()`
- `setWrapText()`
- `setTextOverrun()`
- `setRotate()`
- `setTranslate()`
- `setScale()`

#### Button

- `setOnAction()`
- `setEffect()`
- `addEventHandler()`
- `setStyle()`
- `setStyleClass()`

#### RadioButton

- `setToggleGroup()`
- `setSelected()`
- `requestFocus()`
- `isSelected()`

#### CheckBox

- `setSelected()`
- `setIndeterminate()`

#### ListView

- `setItems()`
- `setCellFactory()`
- `setPrefHeight()`

- setPrefWidth()
- setOrientation()
- getSelectionModel().getSelectedIndex()
- etSelectionModel().getSelectedItem()
- getFocusModel().getFocusedItem()
- setSelectionMode()

### **TableView**

Setter și getter pentru modelul de date.

### **Separator**

- setOrientation()
- setMaxWidth()
- setValignment()
- setHalignment()

### **ColorPicker**

- get.Value()
- set.Value()

### **Tipurile de meniu sunt:**

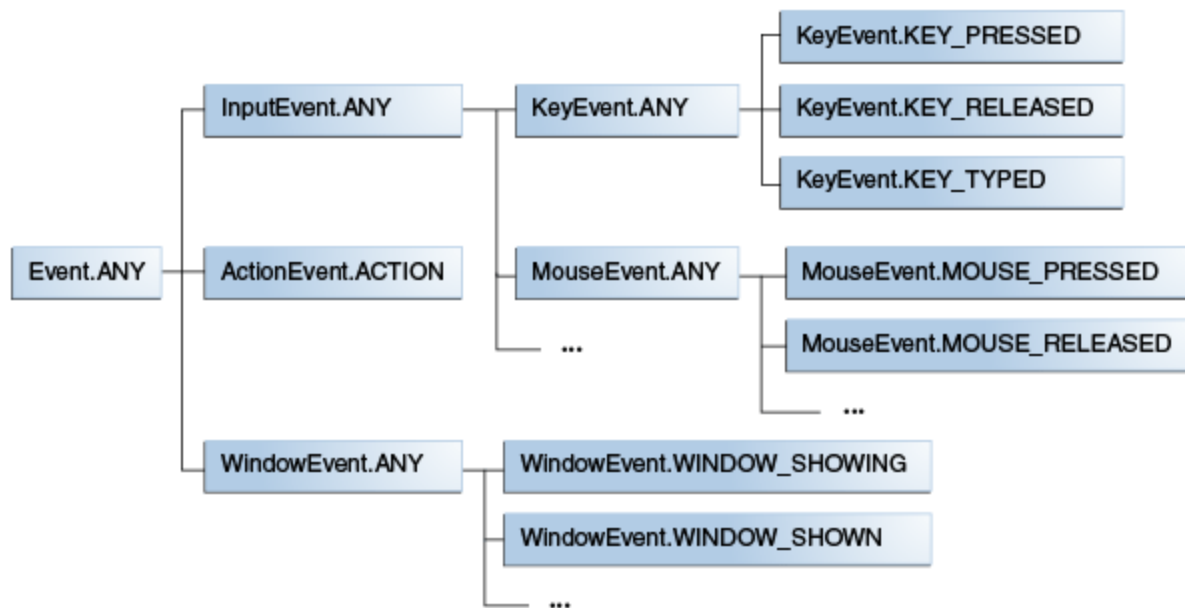
- MenuBar
  - MenuItem
  - Menu
  - CheckMenuItem
  - RadioMenuItem
  - CustomMenuItem
    - SeparatorMenuItem
- ContextMenu

### **Metodele pentru meniu sunt:**

- add()
- addAll()
- getMenus()

### **Modurile de dispunere a elementelor într-o interfață javaFX:**

- GridLayout
- CardLayout
- BorderLayout



## Ex2

```

import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.shape.Line;
import javafx.stage.Stage;

```

```

public class Ex2 extends Application{
    @Override
    public void start(Stage stage) {
        //Creating a line object
        Line line = new Line();

        //Setting the properties to a line
        line.setStartX(100.0);
        line.setStartY(150.0);
        line.setEndX(500.0);
        line.setEndY(150.0);

        Button b1 = new Button();
        b1.setText("Ex3");
        b1.setLayoutX(100);
        b1.setLayoutY(50);

        //Creating a Group
        Group root = new Group(line, b1);

        //Creating a Scene
        Scene scene = new Scene(root, 600, 300);
    }
}

```

```

//Setting title to the scene
stage.setTitle("Sample application");

//Adding the scene to the stage
stage.setScene(scene);

//Displaying the contents of a scene
stage.show();
}
public static void main(String args[]){
    launch(args);
}
}

```

### Ex3

```

import javafx.application.Application;
import javafx.collections.ObservableList;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.text.Font;
import javafx.scene.text.Text;

public class Ex3 extends Application {
    @Override
    public void start(Stage stage) {
        //Creating a Text object
        Text text = new Text();

        //Setting font to the text
        text.setFont(new Font(45));

        //setting the position of the text
        text.setX(50);
        text.setY(150);

        //Setting the text to be added.
        text.setText("Welcome JavaFX");

        //Creating a Group object
        Group root = new Group();

        //Retrieving the observable list object
        ObservableList list = root.getChildren();

        //Setting the text object as a node to the group object
        list.add(text);

        //Creating a scene object
        Scene scene = new Scene(root, 600, 300);
    }
}

```

```

//Setting title to the Stage
stage.setTitle("JavaFx ex3");

//Adding scene to the stage
stage.setScene(scene);

//Displaying the contents of the stage
stage.show();
}
public static void main(String args[]){
    launch(args);
}
}

```

#### Ex4

```

import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.shape.LineTo;
import javafx.scene.shape.MoveTo;
import javafx.scene.shape.Path;

public class Ex4 extends Application {
    @Override
    public void start(Stage stage) {
        //Creating a Path
        Path path = new Path();

        //Moving to the starting point
        MoveTo moveTo = new MoveTo(108, 71);

        //Creating 1st line
        LineTo line1 = new LineTo(321, 161);

        //Creating 2nd line
        LineTo line2 = new LineTo(126, 232);

        //Creating 3rd line
        LineTo line3 = new LineTo(232, 52);

        //Creating 4th line
        LineTo line4 = new LineTo(269, 250);

        //Creating 4th line
        LineTo line5 = new LineTo(108, 71);

        //Adding all the elements to the path
        path.getElements().add(moveTo);
        path.getElements().addAll(line1, line2, line3, line4, line5);
    }
}

```



```
//Creating a Group object
Group root = new Group(path);

//Creating a scene object
Scene scene = new Scene(root, 600, 300);

//Setting title to the Stage
stage.setTitle("Drawing an arc through a path");

//Adding scene to the stage
stage.setScene(scene);

//Displaying the contents of the stage
stage.show();
}
public static void main(String args[]){
    launch(args);
}
}
```