

UNITAT 12

PROGRAMACIÓ GRÀFICA

EXEMPLES

PROGRAMACIÓ
CFGs DAW

Autors:

Joan Vicent Cassany – jv.cassanycoscolla@edu.gva.es

Revisat per:

2022/2023

Llicència



CC BY-NC-SA 3.0 ES **Reconeixement – No Comercial – Compartir Igual (by-nc-sa)** No es permet un ús comercial de l'obra original ni de les possibles obres derivades, la distribució de les quals s'ha de fer amb una llicència igual a la que regula l'obra original. **NOTA:** Aquesta és una obra derivada de l'obra original realitzada per Carlos Cacho i Raquel Torres.

Exemple 01

Creació d'una finestra per a representar un rellotge. Podrem avançar i retrocedir hores, minuts i segons polsant botons.

El Meu Rellotge



Hores	Minuts	Segons
Avançar	Avançar	Avançar
Retrocedir	Retrocedir	Retrocedir

Hora	Minut	Segon
10	15	37

CREACIÓ DEL PROJECTE

Creem un projecte amb la següent configuració.

- Categoria: *Java with Ant/JavaFX*
- Projecte: *JavaFX FXML Application*
- Project Name: *UF12Exemple01*
- Project Location i Project Folder: revisarem que siguin correctes
- JavaFX Platform: revisarem que siga la que hem creat per al JDK 19 de Java FX
- FXMLName: podem deixar el que ens ve per defecte FXMLDocumen
- Create Aplicagtion Class: estarà seleccionat i amb el nom per defecte

Verifiquem que se'ns han creat els tres arxius:

- FXMLDocument.fxml
- FXMLDocumentController.java
- UF12Exemple01.java

Afegim les llibreries de JavaFX:

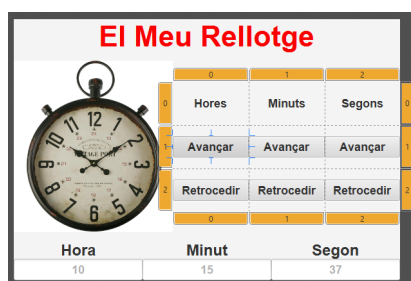
- Botó dret sobre carpeta Libreries
- Triem *Add Library*
- En Available Libreries cercarem la que crearem quan ferem la configuració de NetBeans (*JavaFX* o el nom que assignarem)
- Verificarem que dins de la carpeta Libreries s'han afegit les llibreries.

CONFIGURACIÓ DE LA VISTA O INTERFÍCIE GRÀFICA

Fem doble clic sobre l'arxiu *FXMLDocument.fxml* i s'obrirà **Scene Builder** per a començar a fer el disseny.

- Eliminem tot el contingut que hi ha en el panel de Disseny (o en Documents) per a que quede la zona de treball buida.
- Triem el contenidor: *BorderPane*
- En el panel **Inspector** (o directament en la zona de treball arrossegant) ajustem la mesura del contenidor. Per exemple 600 d'ample per 400 d'alt.
- Veurem en **Document/Hierarchy** que el contenidor *BorderPane* té 5 zones per defecte que corresponen al centre i 4 laterals. Cadascuna d'aquestes zones sols admet dins un element.
- En el top posem un *Label* (etiqueta) per al títol i li posem de nom "*El Meu Rellotge*" i configurarem en **Inspector/Properties** el tipus de lletra, mesura, color, etc. La deixem centrada en el desplegable **Alignement**.
- Fem que ocupe tota la zona on està. Anem a **Inspector/Layout** i en **Max Width** triem *MAX_VALUE*. Veurem com l'etiqueta ocupa tota la zona Top i el text queda centrat.

- En la zona de l'esquerra inserim el control *ImageView*. En **Inspector/Properties** en l'apartat **Image** cercarem la imatge d'un rellotge que ens haurem descarregat i posat en la carpeta del nostre projecte. Ajustem la dimensió de la imatge(per exemple 400x200).
- En la zona centre posem un contenidor de quadricula *GridPane* al que li afegim una columna addicional (boto dret sobre la capçalera d'una columna) per a que quede una graella de 3x3.
- En la primera fila posem 3 etiquetes (*Hores*, *Minuts*, *Segons*) i en les altres dues files posem 6 botons (baix de cada etiqueta posarem *Avançar*, *Retrocedir*).



- Utilitzant la tecla *Shift* anem seleccionant totes les etiquetes i els botons per a posar-los propietats comuns al mateix temps. En **Inspector/Layout** farem que d'amplària s'ajusten al pare (MAX_VALUE) com férem amb el títol. A la lletra li donarem les característiques que vulguem i les centrarem.
- En la zona inferior posem un contenidor de quadricula *GirdPane* de 2 files per 3 columnes. En la primera fila posem tres etiquetes o *Label* (*Hora*, *Minut*, *Segon*) i en la segona fila tres camps de text o *TextField* que contindran els valors variables dels camps. Donem propietats a tot, tal i com ho hem fet abans. Als *TextField* els podem posar un **Prompt Text** per a que es veja el aspecte que tindrà el contingut. A més, desmarquem la propietat **Editable** per a que no es puga escriure sobre aquests.



Anem al menú **Preview/Show Preview in Window** i per veure com ens ha quedat.

CONFIGURACIONS PER AL CONTROLLER EN SCENE BUILDER

Per poder gestionar tots aquests elements des del *Controller* haurem de donar-los nom.

Existeix una sèrie de prefixes recomanat per a nomenar aquests controls gràfics.

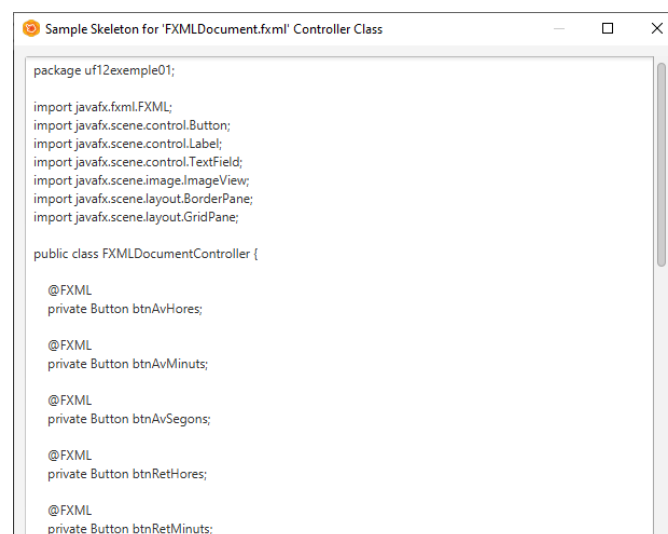
Tipo Objeto / Clase	Prefijo	Ejemplo
Label	lbl	lblHora
TextField	txt	txtHora
Button	cmd	cmdHora
	btn	btnHora
CheckBox	chk	chkSoltero
RadioButton	opt	optMasculino
ComboBox	cbo	cboCiudad
ListBox	lst	lstProducto

Anem a **Inspector/Control** i veurem que tots els controls tenen un **id** on els podem assignar un nom. Ho farem per a tots els controls gràfics que tenim.

Si anem a **Document/Controller** veurem tots els controls amb el seu id.

Anem a associar la nostra interfície gràfica i tots els controls al arxiu controlador que tenim creat en **Netbeans**. Ho trobarem en **Document/Controller/ControllerClass** que al desplegar-lo ens permet seleccionar el nostre arxiu controlador *FXMLDocumentController*.

Des del menú superior de Scene Builder podem veure la llista de handlers que ens inclourà en el nostre controller. Seleccionarem **View/Show Sample Controller Skeleton** i veurem la llista de controls que tenim, que de moment estan buits.



```

package uf12exemple01;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.GridPane;

public class FXMLDocumentController {

    @FXML
    private Button btnAvHores;

    @FXML
    private Button btnAvMinuts;

    @FXML
    private Button btnAvSegons;

    @FXML
    private Button btnRetHores;

    @FXML
    private Button btnRetMinuts;
  
```

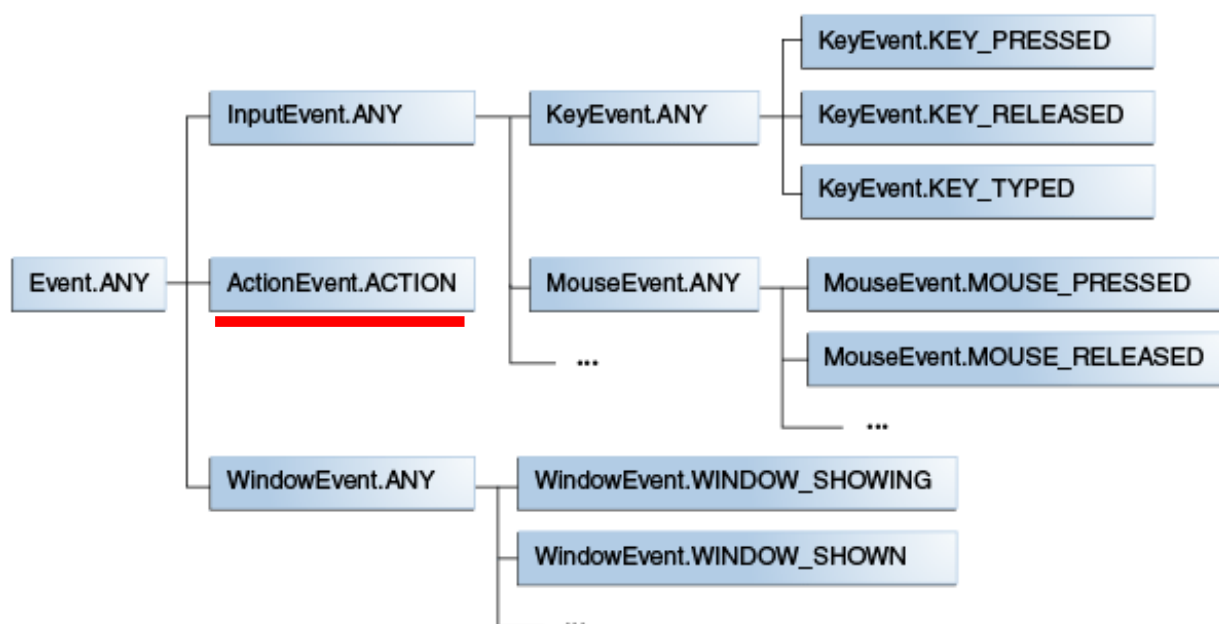
Salvem les modificacions que hem fet des del menú **File/Save**. Al fer-ho podem anar a **NetBeans** a l'arxiu *FXMLDocument.fxml*, polsem botó dret i triem **Edit**. Açò ens permetrà veure el codi FXML que s'ha generat.

NOTA: Es convenient que cada cert temps salvem les modificacions que hem fet.

Quan es realitza una acció en la nostra interfície gràfica (per exemple polsar un botó) es produeix un *event*. Eixe *event* és un objecte de la classe **Event** que provocarà una cridada a un mètode (handler) que es troba en la nostra classe Controller, que haurem de codificar en funció de les accions que volem que es realitzen.

En el següent diagrama podem veure l'estructura d'*events*, hi ha *event* de més baix nivell, específics per a quan hi ha interaccions amb el teclat, amb el ratolí, etc. I altres *events* de més alt nivell com **Action**. S'haurà de consultar en l'API com funciona cadascun d'aquests *events*.

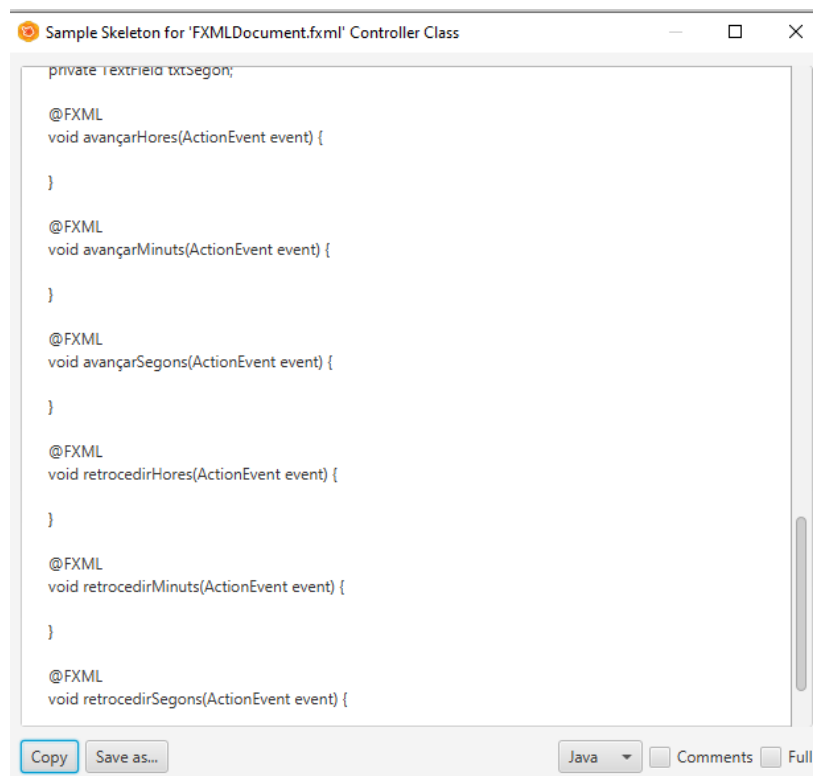
<https://docs.oracle.com/javase/8/javafx/api/toc.htm>



Els controls sobre els que podem interactuar (en nostre cas els botons) necessiten identificar l'*event* front al que el nostre programa ha de respondre. Per tant, anem a **Inspector/Code** i a definir diversos *events* per al control que tenim seleccionat.

Per simplicitat en l'exemple (si ens desplacem cap a ball veurem que hi ha molts), ens quedarem sols amb l'*event* més genèric que apareix al inici **On Action**. En aquesta casella escriurem el nom del mètode que vulguem que s'execute al interactuar amb el botó. Per exemple "*avançarHores*" per al botó Avançar de Hores.

Seleccionarem **View/Show Sample Controller Skeleton** i veurem que ja apareixen aquests mètodes.



Finalment, marquem la casella **Full** per a que s'incloguen totes les importacions que hi ha al principi de l'esquelet i el *void initialize()* del final, i pulsarem **[Save As]**. En proposarà el nom de l'arxiu del nostre Controller i el salvarem.

COMPLETAR CODIFICACIÓ DEL CONTROLLER EN NETBEANS

Ara ja podem obrir en Controller des de **NetBeans** i veure com s'ha incorporat tota aquesta informació.

Per a deixar l'arxiu Controller preparat per a que comencem a fer la codificació de cada mètode sols ens queda crear-nos una variable privada de la nostra classe principal per a que emmagatzeme l'objecte del nostre programa.

NOTA: *El codi de la nostra classe Rellotge es troba més endavant, així que haurem d'haver-nos creat aquesta classe en el nostre projecte abans de continuar.*

Tornant al nostre Controller, inclourem les instruccions d'instanciació de la nostra classe. Inclourem al principi per a localitzar-ho fàcilment:

```
// Definim una instància privada de la classe Rellotge
private Rellotge rellotge;
```

Ara creem un mètode per a actualitzar el valors dels camps en la vista. El posarem al final dels mètodes i abans de del mètode *void initialize ()*, per a localitzar-lo fàcilment. Els mètodes *getxxx* formen part de la nostra classe, els mètodes *setText* ens els proporciona JavaFX.

```
// Mètode per a actualitzar la vista
private void actualitzarVista(){
    String hores= String.valueOf(rellotge.getHora());
    String minuts= String.valueOf(rellotge.getMinut());
    String segons= String.valueOf(rellotge.getSegon());
    txtHora.setText(hores);
    txtHora.setText(minuts);
    txtHora.setText(segons);
}
```


Dins del mètode *void initialize ()*, al final d'aquest per a localitzar-ho fàcilment:

```
// Crear la instància de la classe Rellotge  
rellotge = new Rellotge();  
actualitzarVista();
```

Per tancar aquest apartat ja sols ens queda codificar cadascun dels mètodes que apareixen en el controller. Completarem el mètode *avançarHores (Activation Event)* i la resta de mètodes es codificaran de forma similar.

```
void avançarHores(ActionEvent event) {  
    rellotge.avançarHora();  
    actualitzarVista();  
}
```

NOTA: A partir d'ara hem d'anar en compte si fem modificacions que afecten al Controller en Scene Builder ja que si salvem des de ací podem perdre totes les modificacions fetes des de NetBeans.

PROGRAMA PRINCIPAL

Si anem a la classe principal *UF12Exemple01.java* vorem la declaració

```
public class UF12Exemple01 extends Application
```

Es a dir que la nostra classe hereta de la classe **Application** que és una classe de JavaFX.. D'aquesta classe s'hereta el mètode **start**.

Introduïrem dos conceptes nous: escenes (**Scene**) i escenaris (**Stage**).

- Les escenes es mostren dins dels escenaris i cada pantalla és una escena.

El que estem fent és crear un objecte carregador (*loader*) a partir de la nostra interfície gràfica i ho carregarà en una instància (*root*) de **Parent**, o contenidor pare, que és la classe principal de la qual hereten tots els contenidors. Després, aquest **Parent** (*root*) s'utilitzarà per a crear una escena (*scene*) que finalment s'associarà amb l'escenari.

Finalment, el programa mostra l'escenari

```
stage.show()
```

CLASSE RELLOTGE

```
package uf12exemple01;
/**
 * Classe Rellotge
 */
public class Rellotge {

    private int hora;
    private int minut;
    private int segon;

    // Constructor
    public Rellotge() {
        this.hora = 0;
        this.minut = 0;
        this.segon = 0;
    }

    // Mètodes per a mostrar hora, minuts y segons
    public int getHora() {
        return this.hora;
    }

    public int getMinut() {
        return this.minut;
    }
}
```

```
public int getSegon() {
    return this.segon;
}

// Mètodes per a canviar hora, minuts y segons
public void avançarHora() {
    if (this.hora<23){
        this.hora++;
    } else {
        this.hora=0;
    }
}

public void retrocedirHora() {
    if (this.hora>0){
        this.hora--;
    } else {
        this.hora=23;
    }
}

public void avançarMinut() {
    if (this.minut<59){
        this.minut++;
    } else {
        this.minut=0;
        avançarHora();
    }
}

public void retrocedirMinut() {
    if (this.minut>0){
        this.minut--;
    } else {
        this.minut=59;
        retrocedirHora();
    }
}
```

```
public void avançarSegon() {  
    if (this.segon<59){  
        this.segon++;  
    } else {  
        this.segon=0;  
        avançarMinut();  
    }  
}  
  
public void retrocedirSegon() {  
    if (this.segon>0){  
        this.segon--;  
    } else {  
        this.segon=59;  
        retrocedirMinut();  
    }  
}  
}
```