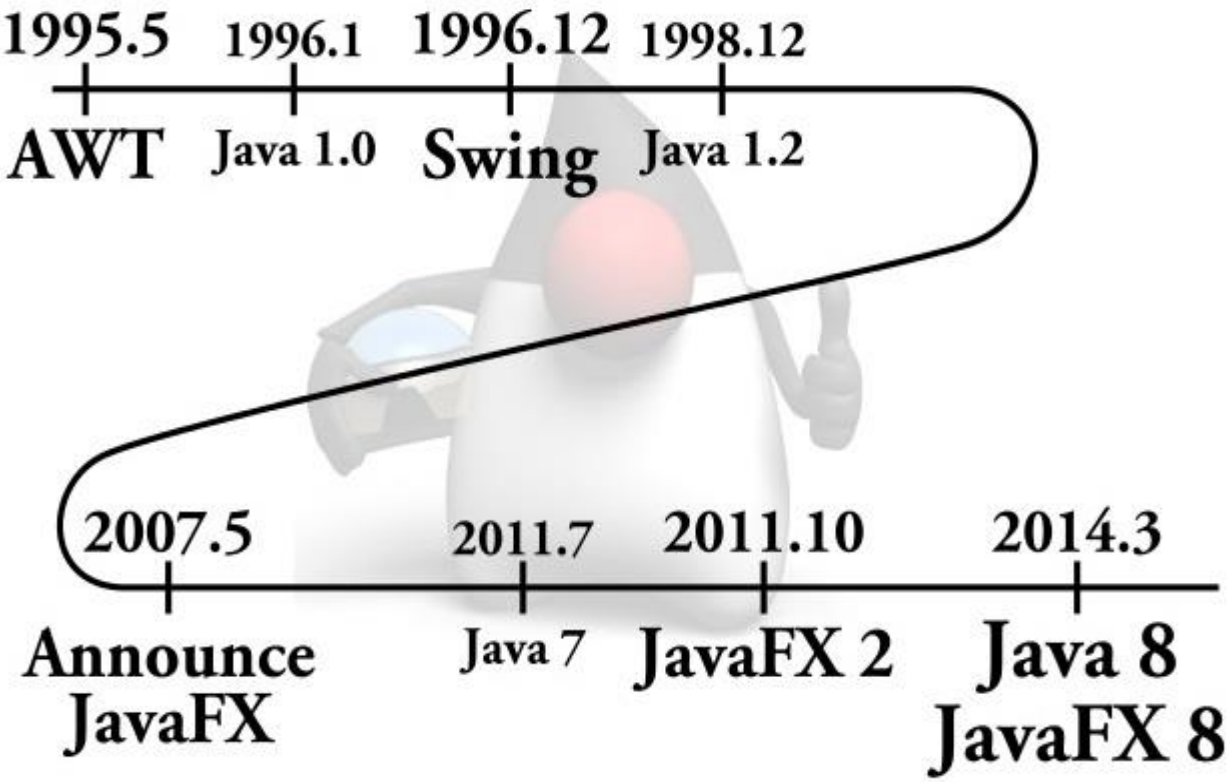
Cours IHM-1
JavaFX

1 - Introduction
Concepts de base

Java GUI – Bref historique [1]

- Chronologie des principales étapes :



Java GUI – Bref historique [2]

- A l'origine du langage *Java*, les interfaces graphiques étaient créées en utilisant la **librairie AWT** (`java.awt`)
  - Composants "lourds" *(heavyweight)* basés sur ceux de la machine cible
  - Difficulté de créer des applications multiplateformes *(write once, run anywhere)*, lourdeur
- Rapidement, la **librairie *Swing*** (`javax.swing`) est venu compléter (et partiellement remplacer) la librairie AWT
  - Composant "légers" *(lightweight)* dessinés par la librairie
  - *Pluggable Look&Feel*
    - ⇨ L&F multiplateforme *(Metal)*
    - ⇨ L&F imitant (plus ou moins bien) ceux des OS spécifiques *(Windows, OS X, …)*
- ***JavaFX 1*** a tenté, sans grand succès, de remplacer *Swing*
  - Essentiellement basé sur un (nouveau) langage de script *(JavaFX Script)*
  - Vaine tentative pour concurrencer *Flex* (basé sur *Flash* et *MXML*)

Java GUI – Bref historique [3]

- Une refonte importante du *toolkit* a pris en compte les critiques formulées et a conduit à une nouvelle mouture : ***JavaFX 2***
- Caractéristiques principales :
  - Abandon du langage de script
  - Choix de deux modes : interfaces basées sur du code *Java* (API) et/ou sur un langage descriptif utilisant une syntaxe XML : **FXML**
  - Création d'un outil interactif ***Scene Builder*** pour créer graphiquement des interfaces et générer automatiquement du code FXML
  - Utilisation possible de feuilles de styles CSS pour adapter la présentation sans toucher au code (créer des thèmes, des *skins*, etc.)
  - Application du modèle de conception *(design pattern) Builder* avec un chaînage de méthodes *(Fluent API)*

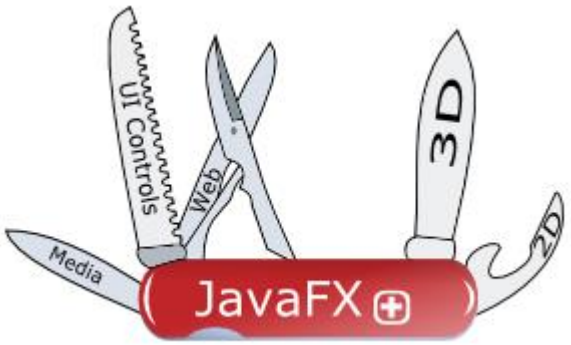
Java GUI – Bref historique [4]

- Avec la sortie de *Java 8*, une nouvelle version baptisée ***JavaFX 8*** a été développée :
  - Intégration dans la distribution de la plateforme standard *Java (JDK, JRE)*
    - ⇨ Plus de librairie externe à télécharger et à référencer
  - ***Scene Builder 2*** : nouvelle version de l'outil d'édition graphique de GUI (*FXML*)
  - Cohabitation améliorée avec les composants *Swing*
  - Prise en compte des nouveaux concepts introduits en *Java 8* et notamment les *expressions lambda* et les *streams*
    - ⇨ Abandon des *Builders (deprecated)*
  - Ajout d'une nouvelle API pour gérer l'impression
  - Ajout de nouveaux composants riches *(DatePicker, TreeTableView, …)*
  - Gestion des écrans tactiles *(TouchEvent, GestureEvent, …)*
  - Amélioration des librairies graphiques 2D et 3D
  - Ajout d'un outil de *packaging* pour simplifier le déploiement des applications
  - *JavaFX* devient le standard officiel pour le développement des interfaces des applications *Java*

Potentiel de *JavaFX* [1]

- *JavaFX* étant le résultat de développements récents, il bénéficie de **concepts modernes** qui en font un framework intéressant pour la réalisation d'applications dans des domaines très divers.

- *JavaFX* est très bien doté pour développer des **interfaces riches** en relation avec des données stockées dans des bases de données ou accessibles au travers de serveurs d'informations.
- Sa riche **librairie graphique** 2D et 3D lui donne également un intéressant potentiel dans des domaines variés :
  - Représentations graphiques
  - Animations graphiques
  - Modélisation (CAD, …)
  - Applications multimédia
  - Réalité virtuelle et augmentée
  - Jeux



Potentiel de *JavaFX* [2]

- La possibilité de **découpler le design graphique** (grâce à l'outil *Scene Builder* et à FXML) permet de déléguer la conception graphique de l'interface à un spécialiste (UI designer) qui n'a pas l'obligation de connaître et maîtriser le langage *Java*.
- L'application possible de **feuilles de style CSS** renforce encore cette séparation entre le design graphique et les traitements qui seront effectués à l'aide de code *Java*.
- Différents **composants complexes** sont disponibles et permettent, avec un minimum d'effort, de créer des applications riches :
  - Effets visuels (ombrages, transitions, animations, …)
  - Graphiques 2D *(charts)*
  - Navigateur web *(WebKit)*
  - Images, audio, vidéo *(media player)*

I/F déclaratives vs procédurales [1]

- La plateforme *JavaFX* offre deux techniques complémentaires pour créer les interfaces (I/F) graphiques des applications :
  - **Manière déclarative**
    - ⇨ En décrivant l'interface dans un fichier FXML (syntaxe XML)
    - ⇨ L'utilitaire graphique *Scene Builder* facilite la création et la gestion des fichiers FXML
    - ⇨ L'interface peut être créée par un designer (sans connaissance *Java*, ou presque...)
    - ⇨ Séparation entre présentation et logique de l'application (MVC)
  - **Manière procédurale**
    - ⇨ Utilisation d'API pour construire l'interface avec du code *Java*
    - ⇨ Création et manipulation dynamique des interfaces
    - ⇨ Création d'extensions et variantes (par héritage)
    - ⇨ Homogénéité des sources de l'application
- Il est possible de mélanger les deux techniques au sein d'une même application (l'API `javafx.fxml` permet de faire le lien entre les deux).

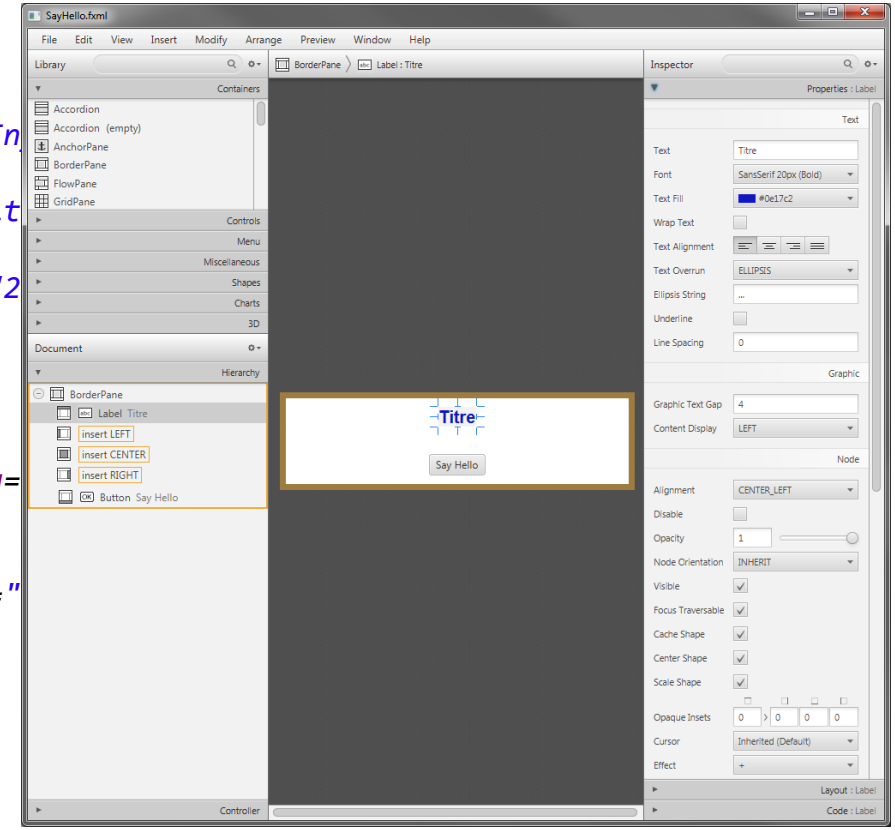
I/F déclaratives vs procédurales [2]

- Technique déclarative (fichier FXML, *Scene Builder* et résultat).

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import javafx.scene.text.*?>
<?import javafx.scene.control.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>

<BorderPane maxHeight="-Infinity" maxWidth="-In
   <top>
      <Label id="title" fx:id="title" text="Tit
         <font>
            <Font name="SansSerif Bold" size="2
         </font>
      </Label>
   </top>
   <bottom>
      <Button fx:id="btnHello" mnemonicParsing=
   </bottom>
   <padding>
      <Insets bottom="10.0" left="10.0" right="
   </padding>
</BorderPane>
```



Déploiement [1]

- Une application *JavaFX* peut être déployée (mise à disposition des utilisateurs) de différentes manières :
  - **Installée localement** comme une application autonome *(standalone/desktop application)*
    - ⇨ Semblable à une application native
    - ⇨ La machine virtuelle *Java* peut être intégrée ou non dans l'exécutable (`.exe` ou `.jar`)
  - **Installée sur un serveur** et intégrée dans une page web
    - ⇨ Lancée depuis un navigateur, en cliquant sur un lien ou sur un autre élément actif de la page
    - ⇨ Lancée automatiquement dès qu'une page est chargée
    - ⇨ Utilise la technique *Java Web Start* (fichier JNLP + descripteur de déploiement XML)
    - ⇨ Une fois téléchargée, l'application peut également être lancée hors-connexion (mise en cache local)

Déploiement [2]

- Différents efforts de développement sont en cours pour permettre de déployer des applications *JavaFX* sur des **terminaux mobiles** (smartphones et tablettes *Android*, *iOS*, …) ainsi que sur des systèmes embarqués (*Raspberry Pi*, …).
- La communauté javafxports.org est notamment active dans ce domaine assez prometteur qui permettrait, à partir d'un code unique, de générer de réelles **applications multiplateformes** (*standalone* et mobiles) comportant des interfaces riches.

Projets connexes

- *JavaFX* devrait, à terme, être totalement publié en *open-source* (ce n'est que partiellement le cas) dans le cadre du projet *OpenJFX*.
- De nombreux projets contribuent à enrichir l'écosystème *JavaFX.*
- Parmi les principaux (et les plus dynamiques) on peut mentionner :
  - ***ControlsFX*** : Projet open-source destiné à offrir des composants supplémentaires *(controls)* de qualité fxexperience.com/controlsfx
  - ***JFXtras*** : Projet open-source destiné à fournir aux développeurs des éléments utiles dans leur vie de tous les jours et qui manquent dans la version de base de *JavaFX* jfxtras.org
  - ***DataFX*** : Projet open-source destiné à faciliter la collaboration entre une application *JavaFX* et un système de gestion des données (DB, …) javafxdata.org
  - ***TestFX*** : Librairie pour automatiser le test des applications *JavaFX* github.com/TestFX

Documentation [1]

- Quelques références web utiles (à mettre dans vos bookmarks) :
  - **Tutoriel officiel** *Oracle* docs.oracle.com/javase/8/javase-clienttechnologies.htm

  - ***FX-Experience*** : Blog géré par des experts du domaine *(news, demos, …)* fxexperience.com
  - Autre **blog** dédié à différentes thématiques *JavaFX* guigarage.com
  - **Communauté des développeurs** du projet open-source *OpenFJX* (qui est un sous-projet de *OpenJDK*) javafxcommunity.com
  - **API** *JavaFX (Javadoc)* docs.oracle.com/javase/8/javafx/api

Attention : *On trouve, sur le web, encore passablement de documentation et de code JavaFX en version 1.x. Des changements majeurs sont intervenus dans les versions 2.x et 8.x. Tout ce qui date d'**avant 2012** doit être considéré avec beaucoup de prudence et de circonspection.*

Documentation [2]

- Quelques livres :
  - **Pro JavaFX 8 -** A Definitive Guide to Building Desktop, Mobile, and Embedded Java Clients
    James Weaver, Weiqi Gao, Stephen Chin, Dean Iverson, Johan Vos, Adrian Chin
    Apress, 2014 ISBN: 978-1430265740
  - **JavaFX 8 - Introduction by Example**
    Carl Dea et Mark Heckler
    Apress, 2014 ISBN: 978-1430264606
  - **Mastering JavaFX 8 Controls**
    Hendrik Ebbers
    McGraw-Hill Professional - Oracle Press, 2014 ISBN: 978-0071833776

Programmation Java

- Avant d'aborder les concepts principaux de *JavaFX* il est nécessaire de présenter quelques éléments de programmation *Java* qui sont utiles voire nécessaires pour développer des applications avec des interfaces graphiques.
- Il s'agit notamment des notions suivantes :
  - Types énumérés (`enum`)
  - Généricité (classes, interfaces et méthodes génériques)
  - Structures de données prédéfinies *(Collections)*
  - Annotations (`@...`)
  - Expressions lambda
  - Références de méthodes
  - Streams
- Ces notions seront sommairement présentées au chapitre suivant qui constitue une petite parenthèse technique de programmation *Java*, indispensable pour aborder la suite.