Développement de programme dans un environnement graphique **Automne 2022** **Cégep Limoilou** **Département d'informatique** **Professeur :** **Martin Simoneau**	**Formatif 2** *Callback* Formatage Dialogue simple Intro aux boutons

Objectifs

- Pratiquer la programmation avec des *callback*
  - Classe interne
  - Classe anonyme
  - Fonction lambda
- Utiliser un bouton *JavaFX* et lui associer un comportement
- Ouvrir un dialogue simple.

À remettre :

- À faire seul.

Contexte :

- Remettre votre projet sur Léa
- Cet exercice sert à assimiler les concepts qui seront nécessaires pour le TP2.
- Nous allons utiliser les callbacks pour :
  - ajouter des comportements aux différents boutons
  - associer une procédure de création de scène à un bouton.

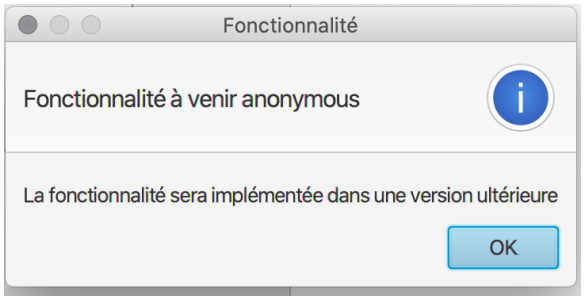
Callback simple

- Assurez-vous que votre projet utilise la librairie Azul Zulu JDK 17 qui est disponible sur Teams.
- Dans la classe ***Classexterne***
  - Faites en sorte que cette classe soit une sorte de *SimpleCallBack.MethodeI*.
  - Implémenter la méthode *fait* pour qu'elle affiche:
    - `méthode normale de ClasseExterne "`
      suivi de la valeur reçu en paramètre
  - Cette classe servira à faire notre premier callback.
- Dans la classe ***SimpleCallBack***
  - On fait les appels de tous les call back
    - Appelez la méthode *afficheCallBack* sur l'instance de app en lui envoyant une nouvelle instance de la classe *ClassExterne*.
    - Appelez encore *afficheCallBack* mais cette fois en utilisant une classe anonyme.
    - Appelez encore *afficheCallBack* mais cette fois en utilisant une closure.
    - Appelez encore *afficheCallBack* mais cette fois en utilisant une référence directe sur la méthode *dooda* de l'instance *app* avec l'opérateur ::
    - Appelez encore *afficheCallBack* mais cette fois en utilisant une référence directe sur la méthode *staticDooda* de la classe *SimpleCallBack* avec l'opérateur *:: .*
  - On va utiliser des variables pour faire les manipulations de référence sur des callbacks

- Commencez par placer une instance appropriée de callback dans chacune des variables (*anonyme*, *closure* et *dooda*)
- Dans la *switch* qui suit affectez à la variable *methodeChoisie*, l'instance qui correspond au numéro saisi par l'utilisateur
    - 1 -> *anonyme*
    - 2 -> *closure*
    - 3-> *reference*
- Appelez *afficheCallBack* en lui envoyant la méthode choisie.

Dialogue et gestionnaire de disposition

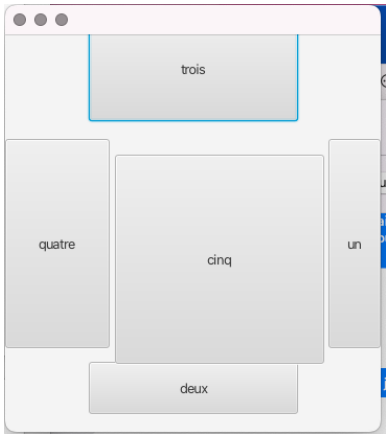
- Programmez la méthode *showAlert* pour qu'elle fasse apparaître un dialogue modal simple (classe *Alert*)
    - Le dialogue doit afficher le titre et l'en-tête reçu en paramètre.
    - Le contenu du dialogue est toujours *"La fonctionnalité sera implémentée dans une version ultérieure".*
    - *Il devrait ressembler à :*



Essayez votre dialogue dans la méthode *start*. Essayez votre dialogue dans la méthode *main* (ça ne marchera pas… pourquoi ?)

- Création d'une scène avec une grille
    - Dans la méthode *Formatif2.**setGridScene**()*, utilisez une gestionnaire *GridPane* pour obtenir la scène suivante:

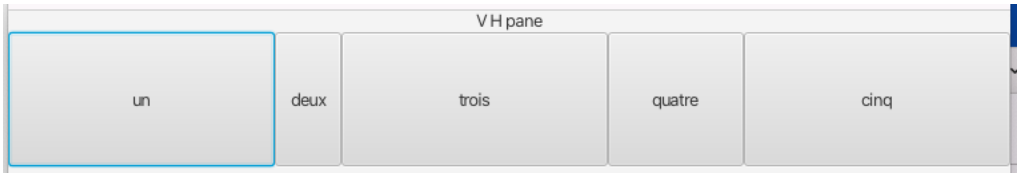
    - Vous devez installer le gestionnaire comme root de l'objet Scene qui est retourné par la méthode.
- Création d'une scène avec des bordure tout le tour de la fenêtre
    - Dans la méthode *Formatif2.**setBorderScene**()*, utilisez une gestionnaire *BorderPane* pour obtenir la scène suivante:



- Vous devez installer le gestionnaire comme root de l'objet *Scene* qui est retourné par la méthode.

- Création d'une scène avec des gestionnaire *VBox* et *HBox*
  - Dans la méthode *Formatif2.**setVHScene**()*, utilisez des gestionnaires *VBox et HBox* pour obtenir la scène suivante:

  - Le titre est un simple *Label*.
  - Utilisez les propriétés de gestion de taille ***Hgrow*** ou ***Vgrow*** pour que la répartition de l'espace lorsque l'on agrandit la fenêtre soit:



    - Le bouton2 n'agrandit pas
    - Le bouton4 n'agrandit que s'il reste la place
    - Les autres boutons agrandissent aussitôt que possible.
  - Vous devez installer le gestionnaire comme root de l'objet *Scene* qui est retourné par la méthode.

Callback et comportements de bouton

- Nous allons associer un comportement à des boutons à l'aide de la classe interne *InnerAction*.
  - Modifiez cette dernière pour qu'elle devienne un *EventHandler<ActionEvent>*
    - Ajoutez la méthode *handle* requise
    - Cette méthode appelle la méthode *showAlert* que nous avons codée dans la section précédente.
    - Transmettez la valeur de l'attribut *headerText* de *InnerClass* pour le second paramètre.

- On va programmer la méthode *associeDialogAuxBoutons* qui branche un *callback* de type classe interne.
    - Cette méthode balaye tous les boutons reçus et elle associe à chacun un objet *InnerAction* à l'aide la méthode *setOnAction* de chaque bouton. Attention *InnerAction* n'est pas statique le new est particulier!
    - Appelez la méthode *associeDialogAuxBoutons* en lui passant les 5 boutons.
    - Démarrez l'application, maintenant en appuyant sur les boutons on devrait voir apparaître le dialogue.
- On va programmer la méthode *associeDialogAnonymous*
    - Cette méthode balaye tous les boutons reçus et elle associe à chacun un objet ***anonyme*** à l'aide la méthode *setOnAction* de chaque bouton.
    - La méthode *handle* appelle encore *showAlert*, mais cette fois elle envoie la variable locale *headerText*. Un objet anonyme peut consulter les variables effectivement finales déclarées dans le même bloc qu'elle (autour d'elle). On appelle cela de la capture de variable.
    - Appelez la méthode *associeDialogAnonymous* en lui passant les 4 et 5 boutons.
    - Démarrez l'application, maintenant en appuyant sur les boutons 4 et 5. On devrait voir apparaître le dialogue.

Callback pour associer une scène à un bouton

- Pour le callback avec *closure* nous allons faire quelque chose d'un peu plus complexe. Nous allons faire en sorte qu'en appuyant sur un bouton on change complètement la scène associée au *primaryStage*.
- Remarquez l'interface *CreationScene* dans le bas de la classe *Formatif2*. Elle ne comporte qu'une méthode *setScene* qui retourne une *Scene*. Ce n'est pas une coïncidence si la signature de cette méthode est la même que celles des méthodes qui crée les scènes (*setGridScene*, *setBorderScene*, *setVHScene*)…
- On va programmer la méthode *associeDialogClosure*
    - Associez au bouton reçu la scène que l'objet implémentant *CreationScene* vous retourne.
- Maintenant, il faut appeler la méthode *associeDialogClosure* en lui envoyant un objet qui implémente l'interface *CreationScene.* C'est encore un cas pour pratiquer les callbacks. Dans la méthode start, on va le faire de 3 façons différentes;
    - affectez à la variable *vhScene* un objet anonyme dont la méthode *setScene* retourne le résultat de *setVHScene*. (elle appelle donc *setVHScene*)
    - affectez à la variable borderScene, une closure qui retourne le résultat de la méthode setBorderScene.
    - Affectez directement à la variable *gridScene* la méthode *setGridScene* de l'objet *this* à l'aide de l'oprateur *::* .
- Finalement, appelez *associeDialogClosure* 3 fois. Chacun des boutons 3,4 et 5 sera associé respectivement aux variables *gridSene*, *vhScene* et *borderScene*.

FIN