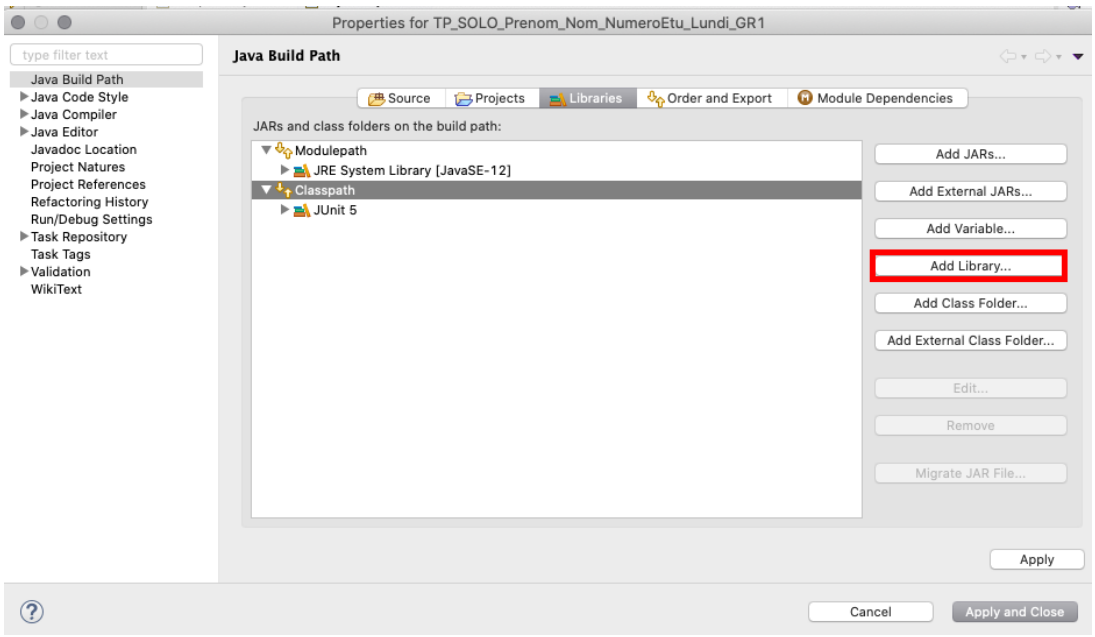

Quelques instructions avant de commencer avec JavaFX :

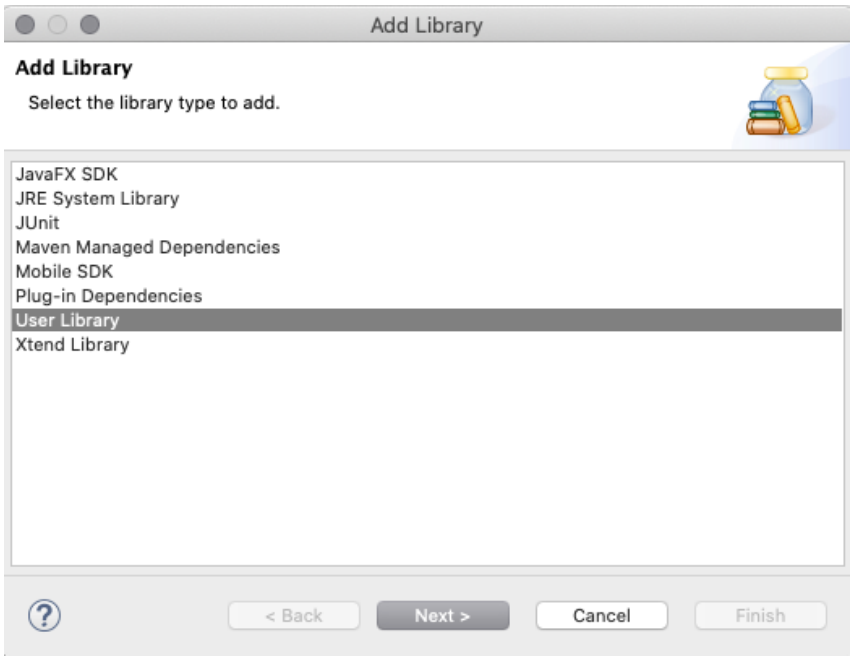
1) Téléchargez le SDK pour votre système d'exploitation en faisant bien attention de choisir JavaFX 11 : `https://gluonhq.com/products/javafx/`

2) Extrayez le contenu de l'archive `zip` à l'emplacement de votre choix

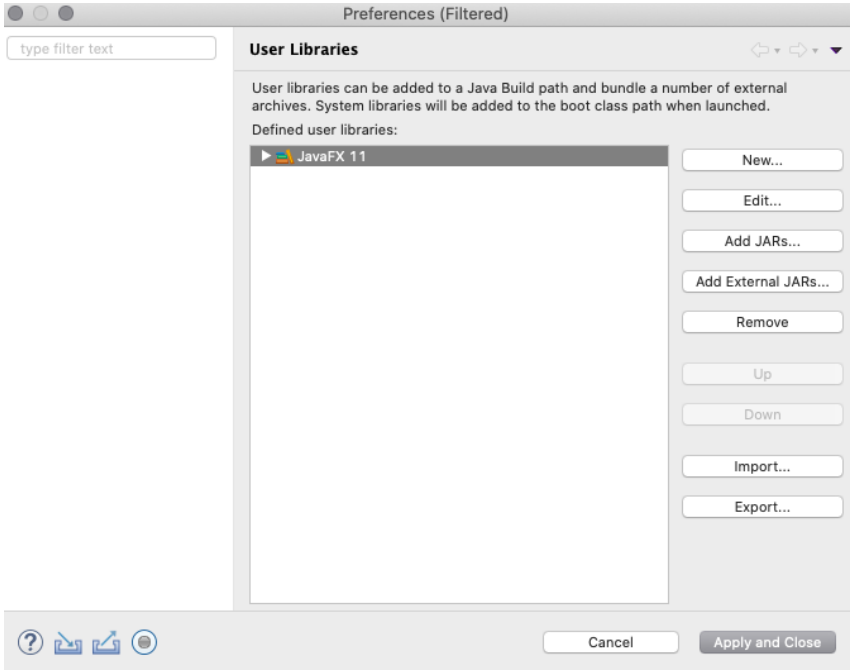
3) Dans Eclipse, ajoutez une bibliothèque au classpath du projet



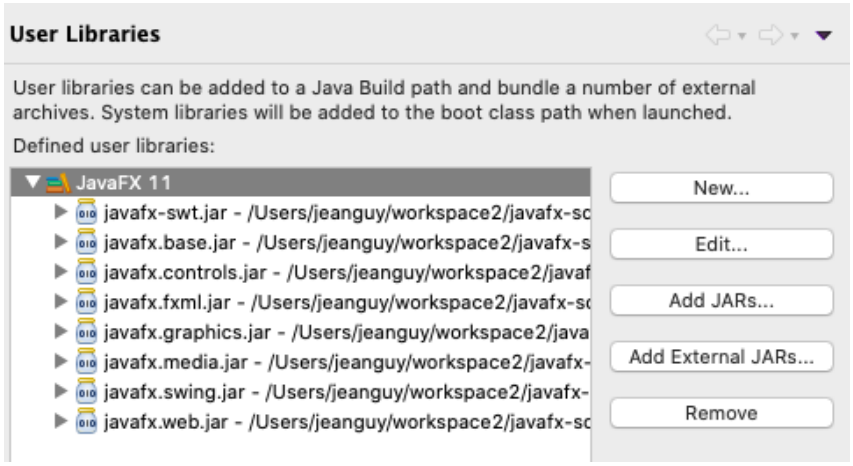
4) Selectionnez `User Library`



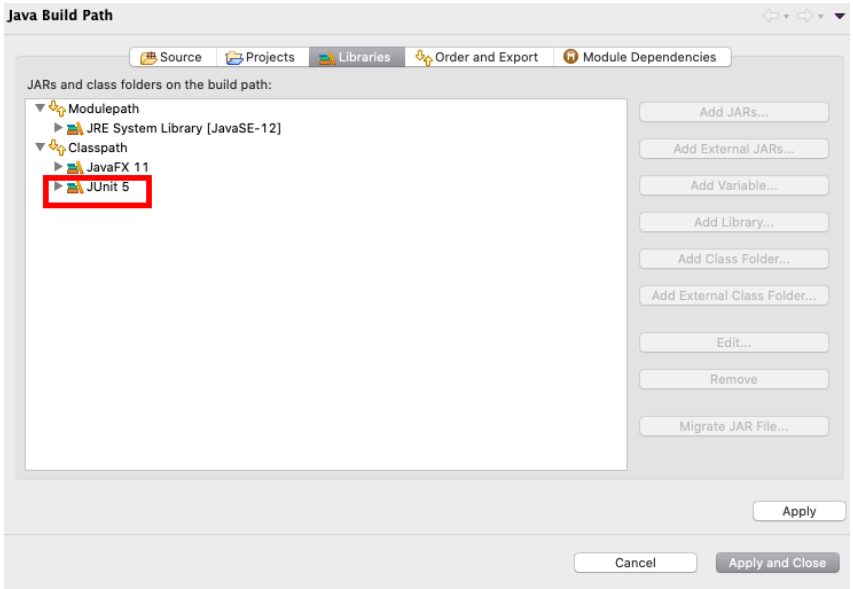
5) Créez une nouvelle bibliothèque que vous appelez JavaFX 11



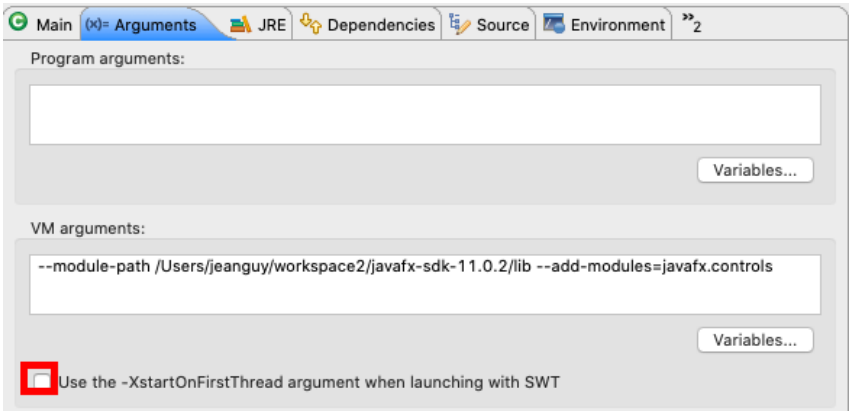
6) Ajoutez les archives `jars` présentes dans le répertoire `lib` du SDK



7) Validez. La nouvelle bibliothèque doit être présente dans le classpath.



8) Pour exécutez le programme, allez dans `Run -> Run Configurations -> Arguments` et ajoutez les arguments suivants à la machine virtuelle : `--module-path /chemin/du/sdk/lib --add-modules=javafx.controls`. Décochez la case `Use the -XstartOnFirstThread ...`



Exercice I
Tester le compteur de clics vu en cours.

Exercice II

1. Définir une classe PanelDate dérivée de BorderPane permettant, grâce à un bouton “Mettre à jour la date”, d’actualiser une date affichée dans un label. La date affichée dans ce label est donc la date au moment du dernier clic sur le bouton.

2. Définir une classe PanelTempsEntreDeuxClics dérivée de BorderPane. Le panel est composé d’un label et d’un bouton. Lors du clic sur le bouton, on note grâce au label le temps en secondes écoulé depuis le clic précédent.

3. Écrire le programme de la question 1. avec :

   1) une version PanelDateI avec une classe de handler définie comme une classe interne non anonyme (pour cette version, déclarer en attribut du panel les composants graphiques utilisés dans le handler).

   2) une version PanelDateA avec une classe de handler définie comme une classe interne anonyme.

   3) une version PanelDateL en utilisant une lambda expression.

Exercice III
Définir une classe PanelCouleurBouton avec deux boutons :

- un bouton de texte “Couleur rouge” et de couleur rouge ;
- un bouton de texte “Couleur verte’ et de couleur verte ;
- un label de texte “Je change de couleur.”.

Faire le nécessaire pour que les deux boutons modifient la couleur du texte du label (le foreground : setTextFill (Color.RED) ou Color.GREEN) : rouge ou vert.

Exercice IV
Définir une classe LabelLien dérivée de Label telle que lors du passage de la souris sur le composant, le foreground du label soit modifié.

Exercice V

L'objectif de cet exercice est d'implémenter une interface graphique pour le jeu du nombre secret. L'ordinateur doit trouver un nombre compris entre 1 et 100, et l'utilisateur du programme lui donne des indications. Ci-dessous se trouve le code source de la classe Jeu que vous devez utiliser sans modification.

Pour voir le fonctionnement du jeu, récupérer l'archive `nombreSecret.jar` sur Moodle. C'est un jar exécutable qui permet de tester le jeu.

```
public class Jeu {
  /** le nombre minimum a trouver */
  private int minInitial ;
  /** le nombre maximal a trouver */
  private int maxInitial ;
  /**
    * au cours d'un jeu, le minimum deduit des reponses precedentes
    */
  private int min ;
  /**
    * au cours d'un jeu, le maximum deduit des reponses precedentes
    */
  private int max ;
  private int nbPropositions ;
  private boolean gagne = false ;

  public Jeu(int minInitial, int maxInitiam) {
    this.minInitial = minInitial ;
    this.maxInitial = maxInitial ;
    resert() ;
  }

  /**
    * Methode utilisee pour redemarrer le jeu
    */
  public void resert() {
    this.min = minInitial ;
    this.max = maxInitial ;
    this.gagne = false ;
    this.nbPropositions = 0 ;
  }

  /**
    * @return true si le jeu est sans solution
    */
  public boolean isJeuSansSolution() {
    return this.min > this.max ;
  }

  /**
    * @return true si le joueur a gagne
    */
  public boolean isGagne() {
```

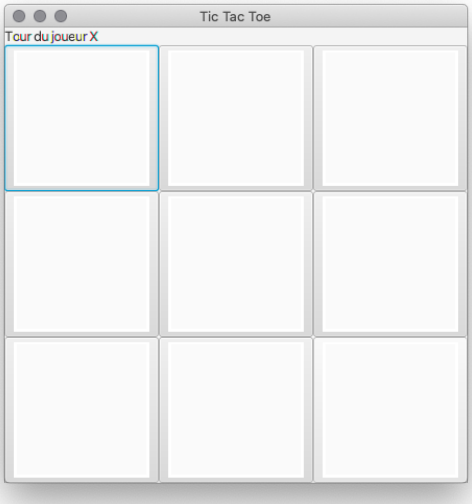
```
    return this.gagne ;
  }

  /**
    * @return la proposition de l'ordinateur
    */
  public int getProposition() { return (this.min + this.max)/2 ; }

  public int getNBPropositions() { return this.nbPropositions ;}

  /**
    * Rajoute 1 au minimum
    */
  public void noterPropositionTropPetite() {
    this.nbPropositions++ ;
    this.min = this.getProposition() + 1 ;
  }

  /**
    * Enleve 1 au maximum
    */
  public void noterPropositionsTropGrande() {
    this.nbPropositions++ ;
    this.max = this.getProposition() - 1 ;
  }

  /**
    * Le maximum et le minimum prennent la valeur du nombre
    * qui vient d'etre trouve
    */
  public void noterPropositionOK() {
    this.nbPropositions++ ;
    this.min = this.getProposition() ;
    this.max = this.min ;
    this.gagne
  }
}
```

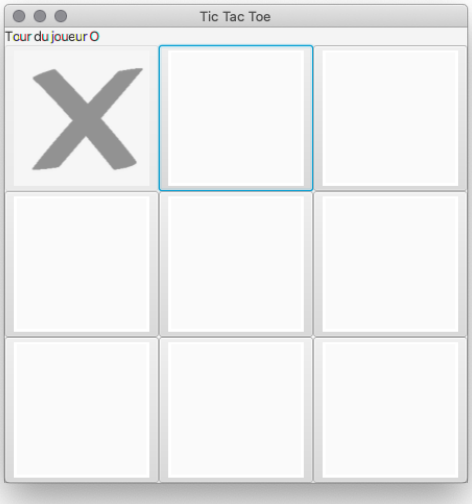
Exercice VI (Bonus : répertoires et interfaces graphiques)
En utilisant ce qui a été fait dans les TD précédents, écrivez un programme qui permet à l'utilisateur de :

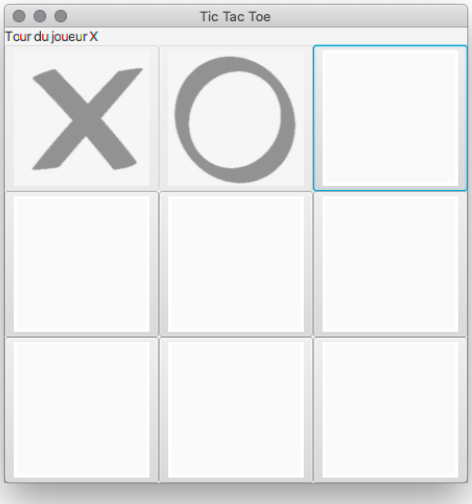
- créer un répertoire lui appartenant ;
- ajouter des contacts dans son répertoire ;
- faire des recherches dans son répertoire (en fonction du prénom et du nom, ou en fonction du numéro de téléphone) ;
- d'afficher tous les contacts ;
- de sauvegarder le répertoire dans un fichier texte ;

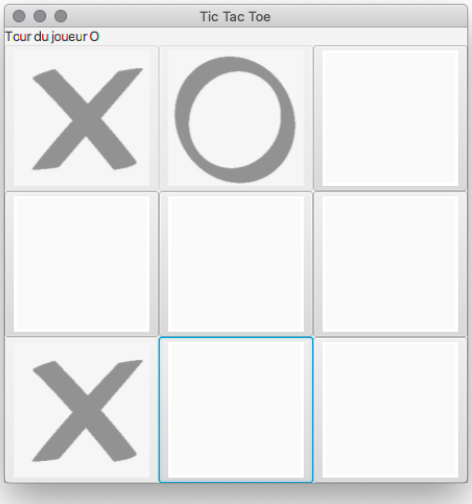
- de charger un répertoire depuis un fichier texte.

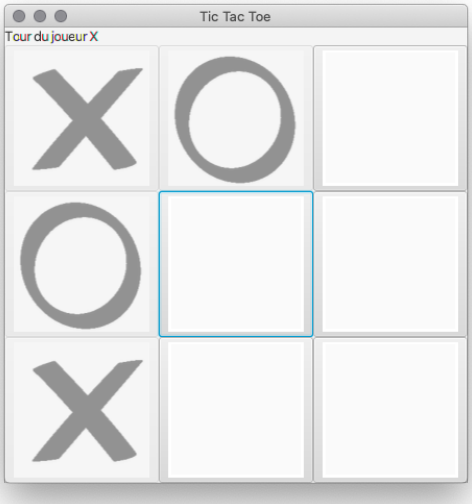
Exercice VII (Bonus : Tic-Tac-Toe)
Reprenez les interfaces et les classes vues en cours. Créez une version du Tic-Tac-Toe avec interface graphique, en utilisant JavaFX.

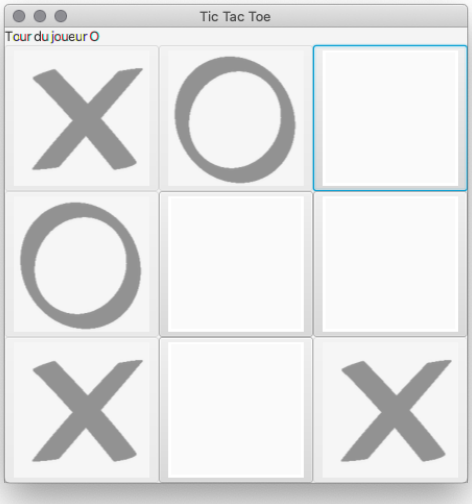


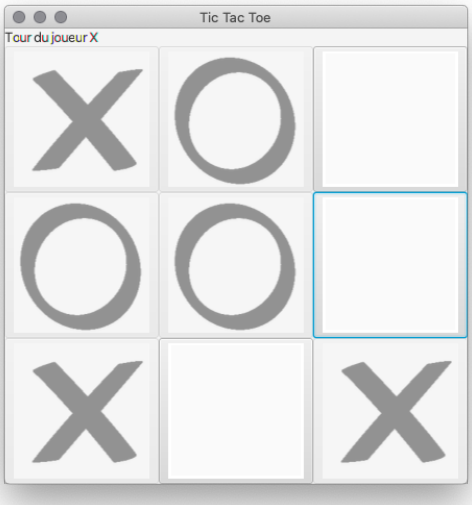


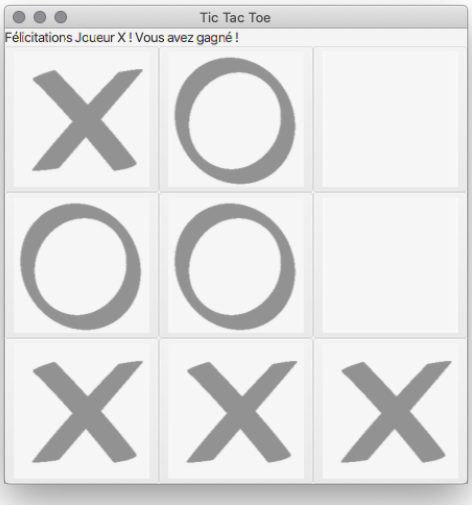





Tic Tac Toe
Tour du joueur X


Tic Tac Toe
Tour du joueur O


Tic Tac Toe
Tour du joueur X


Tic Tac Toe
Félicitations Jcueur X ! Vous avez gagné !