Voici mon projet de mini calculatrice avec une interface javafx et java réalisé pour Simplon.Co.

Ce code a été fait sous Eclipse IDE.
Utilisation de la calculatrice : il suffit de lancer le programme, puis de selectionner les chiffres avec les boutons associés afin de faire des calculs simple de forme (a + b), ou bien (a-b).
Si vous faites une erreur de saisie vous pouvez appuyer sur le bouton corriger afin d'effacer la saisie précédente, ou bien vous pouvez appuyer sur le bouton effacer pour supprimer toute votre saisie précédente.

Explication de mon code. (Remarques en fin de document)

Premièrement j'ai initialisé la classe "**Main**" qui étend la classe "**Application**".
J'ai créé une nouvelle instance de la classe "**gridPane"**, que j'ai appelée "grid".
Puis j'ai créé une nouvelle scène de fenêtre graphique que j'ai appelée "**vuFenetre**".
Ensuite j'ai défini la taille de la scène à 500 px de largeur et 500px de hauteur ainsi que la couleur de l'arrière plan en **blanc**.
Et après j'ai créé une instance de la classe "**Image**", en utilisant un lien url de l'image que je voulais afficher, il s'agit du logo Simplon.co.
Ensuite j'ai créé une nouvelle instance de la classe "**ImageView**", que j'ai appelée "**monImage**".
Grâce aux méthodes "**setFitWidth"** et "**setFitHeight**". On a défini la largeur et la hauteur de l'image à 225 px.
Mon objet imageView est positionné dans la grille "grid" en utilisant la méthode **GridPane.setConstraints** pour fixer mes contraintes.

On s'est servi de la méthode "**GridPane**.**setHalignment**" pour définir l'alignement horizontal de mon objet "ImageView" au centre de la cellule de la grille.
Pour finir l'objet "ImageView" est ajouté à la liste d'enfants du GridPane "grid" en utilisant la méthode "**getChildren**().**add**" ce qui permet d'afficher l'image dans la fenêtre.
J'ai fait en sorte que la fenêtre principale ne puisse pas être redimensionnée.
Et enfin la scène sera affichée dans la fenêtre principale.

J'ai créé objet de type **TextField** et l'ai initié avec une chaîne de caractère vide.
J'ai appelé cette variable **text_emplacement**.

J'ai créé des boutons pour les chiffres de **0 à 9**, ainsi que les opérateurs **"+", "-", "=",** "**Corriger**" et "**Effacer**" grâce à la classe "**Button**".
Dans lesquels j'ai respectivement affiché la chaîne de caractère correspondant.

Pour le positionnement de mes boutons j'utilise les méthodes "**setHgap**()" et "**setVgap**()", pour faire des espacements de 10 pixels sur la grille, et "**setAlignment**()" pour les centrés.
J'ai ajouté tous mes objets créés précédemment à la grille avec la méthode "**add**".
J'ai précisé les coordonnées de chacun des mes objets afin de contrôler exactement où ils sont affichés dans la grille, selon un axe (colonne, ligne).

J’ai personnalisé mes boutons, de 2 tailles différentes, avec respectivement 2 tableaux.

Le 1er tableau comprend 12 boutons de calculs, avec 2 variables de type int, "**boutonLargeur**" et "**boutonHauteur**" et une boucle "**for**" qui est utilisée pour parcourir chaque bouton dans le tableau des boutons.
Pour chaque bouton, les méthodes "**setPrefWidth**" et "**setPrefHeight**" sont appelées pour définir respectivement leur largeur et hauteur en utilisant mes variable “**boutonLargeur”** et “**boutonHauteur”**.

J'ai effectué à nouveau la même procédure pour avoir les boutons que je souhaitais dans une taille différente avec cette fois-ci 2 nouvelles variables “**boutonLargeur1” et “boutonHauteur1”.**

Dans cette partie du code je définit les intéractions qu'ont les boutons dès lors que l'on clique dessus.

Lorsque l'on clique sur un des boutons de 0 à 9, il récupère le texte affiché dans le champ de text ("**text_emplacement**")
puis ajoute un "**Insérer chiffre respectif**" à la fin, et affiche le nouveau texte.

Les boutons des opérateurs agissent différemments :

J'ai défini le comportement du bouton "**boutonPlus**" lorsque l'on clique dessus.
Après un clic, le code récupère le texte actuel du composant "**text_emplacement**" en appelant la méthode "**getText**()".
Puis il vérifie si le texte récupéré ne se termine pas par soi "+" ou "-", en utilisant la méthode "**endsWith**()".
Dans ce cas, il ajoute le caractère "+" à la fin du texte.
Ensuite il actualise le composant "**text_emplacement**" avec le nouveau texte en appelant la méthode "setText()".
Pour le bouton “boutonMoins” j’ai fait strictement le même code mais avec l’opérateur -.

Je définis une action pour le bouton "**bouttonCorriger**".
Lorsque l'on clique dessus, le texte contenu dans "**text_emplacement**" est corrigé.
Cette correction consiste à vérifier dans un premier temps si le texte n'est pas vide, en utilisant la méthode "**isEmpty()**".
Si le texte n'est pas vide, le dernier caractère saisi sera supprimé par la méthode "**substring**()".
La méthode "**setText**()" met à jour "**text_emplacement**" avec le texte corrigé.

Je définis une action pour le bouton "**bouttonEffacer**".
Lorsque l'on clique dessus, le texte contenu dans "**text_emplacement**" est effacé.
→ L'action est effectuée par la méthode "**handle**" qui est appelée lorsque l'on clique sur le bouton, puis l'interface "**EventHandler**" est implémentée.

J’ai bien évidemment importé tous les packages nécessaires au bon fonctionnement de mon code.

Dans cette partie du code je définit les intéractions qu'a le bouton Égale

Lorsque l'on clique sur le bouton, le code :
-Récupère le texte saisie dans la zone de texte (**text_emplacement**) avec "**getText**()"
-Il vérifie si le texte saisi n'est pas vide.
-Il sépare le texte actuel en 2 parties (soit avec un + ou un -), sous forme de tableau "nombre"
-Il tente de convertir les 2 composantes du tableau "nombre" en entiers en utilisant la méthode "**Integer.parseInt(String)**" pour convertir la chaîne en un entier (int).
-Si la conversion réussit, le code détermine le signe de l'opération (+ ou -) dans la chaîne, calcule le résultat et le convertit en chaîne en utilisant la méthode "**Integer.toString(Int)**".
-Le texte du résultat sera alors affiché dans la zone de saisie "**text_emplacement**".
-Si la conversion échoue, le message d'erreur "Entrez deux nombres valides" est affiché dans la zone de saisie "**text_emplacement**".

Remarques

On m'a demandé de faire une calculatrice qui pourra faire l'addition et la soustraction de 2 nombres, donc j'ai volontairement ignoré les opérations multiplier, diviser ainsi que les nombres décimales.

J'ai privilégié le format calculatrice standard avec des boutons au lieu d'une calculatrice avec des saisies de texte à faire.

J'ai décidé de faire un seul fichier "Main" qui contient tout mon code puisqu'il est relativement simple, je n'ai pas eu la nécessité de faire un fichier fxml.

La partie la plus compliquée de code a été le bouton Égale, j'utilisais initialement la méthode Scanner, que je n'ai pas réussi à implémenter, car aucune réaction après compilation.
J'ai coder à la main sans utiliser d'interface graphique, j'aurai pu utiliser JavaFX et Scene builder pour me faciliter la tâche sur la fenêtre, mais j'ai préféré rester dans la simplicité.

J'ai choisi de mettre des int dans mon bouton égale cependant j'aurai pu décider de mettre un long pour faire des calculs de nombre plus grand, ou bien des doubles pour des nombre bien plus grand.

CONCEPTS DE POO

On m'a demandé de donner / expliquer 3 concepts de la programmation orienté objet. :

Le principe fondamental de la POO est l'encapsulation, qui consiste à regrouper les données et les fonctionnalités associées dans des objets pour mieux les gérer et les organiser.

Un autre exemple est le principe d'encapsulation, il permet de cacher les détails d'implémentation d'une classe derrière une interface publique.
Exemple : dans une application de gestion de compte bancaire, une classe "CompteBancaire" peut-être définie avec des attributs tels que le titulaire du compte, le numéro de compte, la date d'ouverture du compte ou bien le solde de ce dernier.
Les méthodes de la classe peuvent inclure des opérations telles que des retraits, des dépôts, des vérifications du solde.
Les détails de la mise en œuvre de ces opérations sont cachés à l'extérieur de la classe et ne sont accessibles qu'à travers les méthodes publiques. Cela garantit la sécurité des données en empêchant un accès direct et non autorisé aux attributs du compte.

Pour finir le concept d'héritage, il permet de définir une nouvelle classe en se basant sur une classe existante. La classe dérivée peut utiliser toutes les propriétés et méthodes de la classe de base, et éventuellement les surcharger pour les adapter à ses besoins spécifiques.

Par exemple, dans mon code avec "public class Main extends Application", on a la classe "Main" qui hérite de la classe "Application", ce qui veut dire qu'elle peut accéder à toutes les propriétés et méthodes publiques de cette dernière.
Si la classe "Application" possède des propriétés et méthodes qui peuvent être utiles pour la classe "Main", elles pourront être utilisées sans avoir à les réécrire.