Projet Grapheur

Création d'une interface graphique et d'un évaluateur de fonctions permettant l'affichage des courbes d'une ou plusieurs fonctions.

Wenger Arthur

36000244

06/09/2017

I. PRESENTATION DU PROJET

Le grapheur présenté dans ce projet propose les fonctionnalités suivantes :

- Affichage dynamique d'une ou deux fonctions avec possibilité de zoomer/dézoomer et déplacer le graphique.

- Affichage d'informations concernant les bornes de la fenêtre de dessin, le tracé des courbes et les coordonnées de la souris sur le graphique.

- Options pour calculer automatiquement le pas en fonction de l'échelle, afficher un quadrillage, modifier la couleur des courbes, recentrer les axes, et rafraichir l'affichage.

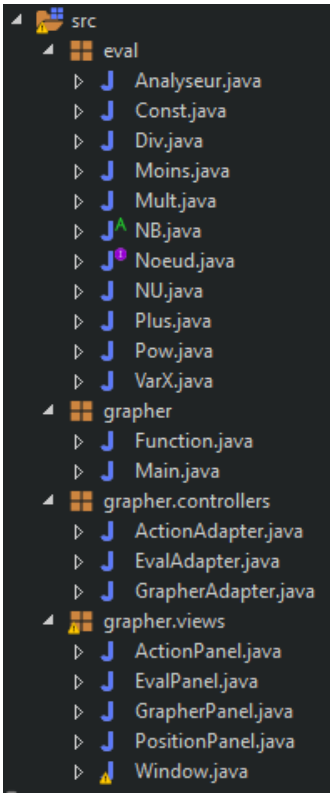
A. Lancement de l'application

Le projet est constitué d'une archive contenant les classes compilées « projet_compile.zip », d'une autre archive contenant le code source de l'application « projet_sources.zip » ainsi que de la présente documentation.

Afin de lancer l'application, renommer l'archive « projet_compile.zip » en « projet_compile.jar » puis cliquer sur ce fichier jar.

Afin d'obtenir les classes compilées et le code source du projet il suffit de dézipper l'archive appropriée.

B. Structure des fichiers



Le projet est scindé en plusieurs packages séparant les fonctionnalités de l'application.

Le package « eval » contient l'ensemble des classes permettant de construire un évaluateur de fonction. La classe « Analyseur » permet de transformer une chaîne de caractère en arbre d'expression tandis que l'ensemble des autres classes permettent de décrire les différents types de nœuds composant cet arbre.

Le package « grapher » contient l'ensemble des classes permettant de construire l'interface graphique de l'application. La classe « Main » permet de lancer l'application tandis que la classe « Function » contient un modèle de fonction manipulable par les vues et les controlleurs.

Le package « grapher.views » contient la fenêtre principale « Window » ainsi que les quatre vues qui la compose. Ces quatre vues sont des panels disposés au nord au sud et à l'ouest de la fenêtre principale.

Le package « grapher.controllers » contient les trois contrôleurs permettant de gérer les évènements et les interactions entre les vues.

II. L'EVALUATEUR

A. Principe

L'évaluateur permet de transformer une chaîne de caractère en une fonction mathématique. Cette fonction est modélisée sous la forme d'un arbre d'expression dans lequel chaque nœud représente un opérateur et chaque feuille représente les constantes ou les variables manipulées.

Un arbre d'expression est défini comme l'imbrication d'un ensemble de nœuds disposant chacun d'une méthode d'évaluation. Ainsi, calculer la valeur d'une fonction f(x) revient à calculer récursivement le résultat de l'évaluation de chaque nœud composant l'arbre en fonction d'une valeur de x.

L'ordre de création des nœuds composant un arbre dépend donc essentiellement de l'ordre de priorité des opérateurs détectés dans une chaîne de caractère.

L'évaluation des expressions parenthésées est effectuée en utilisant un algorithme de masquage. Celui-ci consiste à remplacer temporairement le contenu d'un couple de parenthèses afin d'évaluer en premier les opérateurs non parenthésés. L'analyse du contenu des parenthèses sera donc retardée, modifiant ainsi sa priorité dans l'évaluation d'une chaîne de caractères.

B. Implémentation

Tous les nœuds de l'arbre d'expression implémentent l'interface « Nœud » qui dispose d'une méthode d'évaluation « eval » ainsi que d'une méthode d'affichage « toString ». La méthode eval renvoie le résultat de l'évaluation de chaque nœud composant l'arbre en fonction de la valeur d'une variable x. La méthode toString affiche la fonction correspondant à l'arbre en notation préfixé.

On distingue deux types de nœuds : les nœuds binaires et les nœuds unaires.

Les nœuds binaires sont définis par la classe abstraite « NB ». Tous les nœuds binaires sont des opérateurs qui ont deux fils de type Nœud. Chaque opérateur binaire hérite de la classe « NB » et définit sa propre méthode d'évaluation.

Ainsi, l'opérateur de soustraction est défini par la classe « Moins », la multiplication par la classe « Mult », l'addition par la classe « Plus », la division par la classe « Div » et la puissance par la classe « Pow ».

Les nœuds unaires sont définis par la classe « NU ». Plutôt que de créer une classe héritant de NU pour chaque type d'opérateur unaire, cette classe dispose d'un attribut « op » permettant de modifier le comportement de la méthode eval en fonction du type d'opérateur unaire. Les opérateurs unaires susceptibles d'être crées avec cette classe sont : les fonctions ln, log, exp, sin, cos, tan, abs (valeur absolue) et sqrt (racine carré) ainsi que l'opérateur négatif – et l'opérateur carré ^{2}.

Enfin, la classe « Analyseur » permet de construire un arbre d'expression à partir d'une chaîne de caractères. La méthode « parse » détecte les opérateurs présents dans la phrase en fonction de leur priorité et construit chaque nœud récursivement sur des morceaux de phrase de plus en plus petits. Lorsque des parenthèses sont détectées, leur contenu n'est pas évalué tant qu'il est encore possible de créer d'autres nœuds. Lorsque la phrase ne contient plus qu'une expression parenthésée alors on supprime ces parenthèses et on évalue son contenu.

C. Exemple

Exemple d'évaluation pour la chaîne de caractères « 2*(x+1) » :

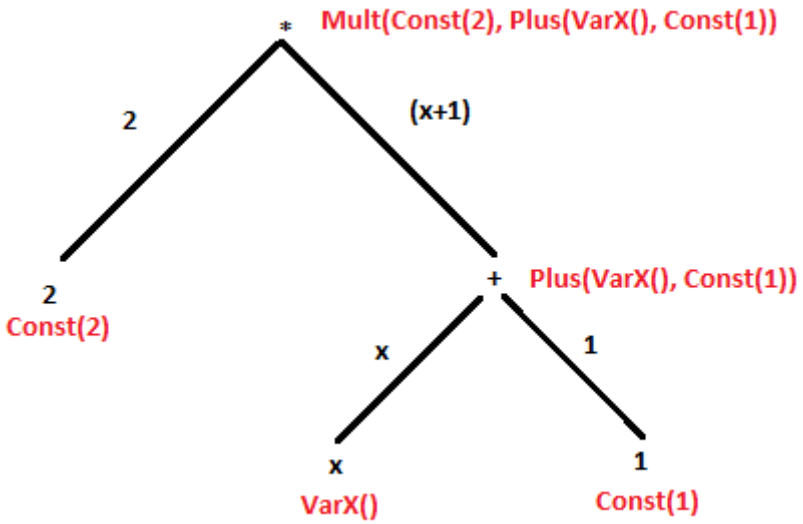
L'algorithme commence par chercher le prochain opérateur qu'il doit évaluer avec la méthode « nextOp ». Il détecte la présence de parenthèses dans la phrase et masque temporairement son contenu. La phrase devient : « 2*###### ».

Le caractère « * » étant détecté, la méthode parse va créer un nœud binaire « Mult » ayant pour fils gauche le résultat de la transformation de la chaîne « 2 » par la méthode parse et pour fils droit le résultat de la transformation de la chaîne « (x+1) » par la méthode parse.

La méthode parse évalue la phrase « 2 » et détecte qu'il s'agit d'un nombre. Un nœud unaire « Const » contenant la valeur 2 va donc être crée.

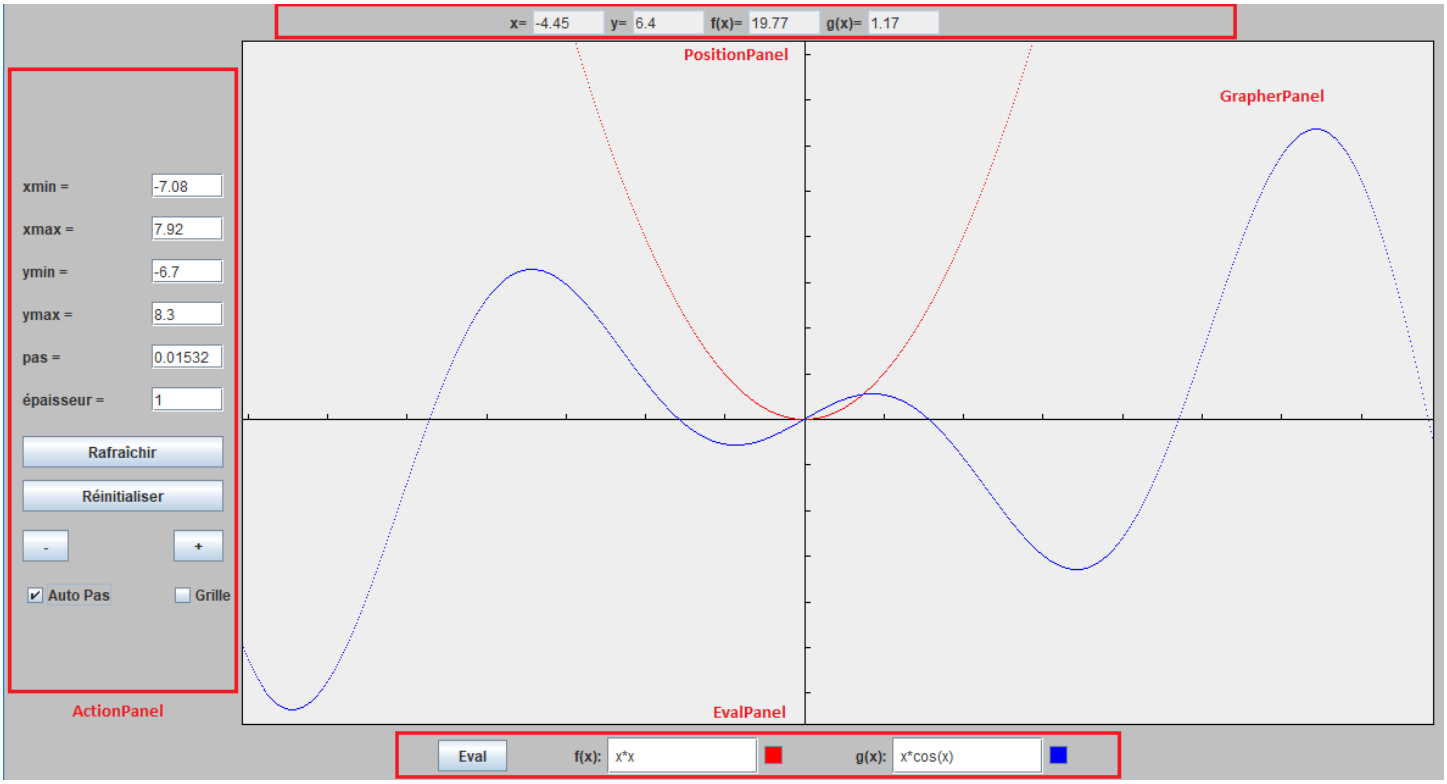
La chaîne de caractères « (x+1) » est à son tour évaluée. Les parenthèses inutiles sont supprimées et l'opérateur + est détecté. Un nœud Plus est donc crée ayant pour fils gauche la chaîne « x » et pour fils droit la chaîne « 1 ». De la même façon, un nœud « VarX » sera créé lors de l'évaluation de « x » et un nœud « Const » sera créé lors de l'évaluation de « 1 ».

La notation préfixée correspondant à l'arbre d'évaluation sera donc : *(2, +(x, 1)).



III. LE GRAPHEUR

A. Les vues



La fenêtre principale de l'application est construite avec la classe « Window ». Celle-ci regroupe quatre vues disposées au nord, au sud, à l'ouest et au centre de la fenêtre.

Le panel Sud « EvalPanel » permet de saisir deux fonctions F et G, de modifier leur couleur et de les afficher simultanément à l'écran avec le bouton Eval. Si le champs d'une fonction est vide, celle-ci n'est plus affichée sur le graphique.

Le Panel Nord « PostionPanel » permet d'afficher les coordonnées de la souris en abscisse et en ordonnée ainsi que les valeurs des fonctions F et G correspondantes.

Le Panel Ouest « ActionPanel » permet de modifier les bornes d'affichage des fonctions, leur épaisseur ainsi que l'intervalle entre deux points successifs de la courbe. Le bouton « Rafraîchir » permet de mettre jour l'affichage avec les options définies plus haut. Le bouton « Réinitialiser » permet de rétablir les valeurs par défaut pour l'ensemble de ces options. Les boutons « + » et « - » permettent respectivement de zoomer et de dézoomer sur le graphique. L'option « Autopas » permet de calculer automatiquement la valeur du pas en fonction de l'échelle. Enfin, l'option « Grille » permet d'afficher un quadrillage sur le graphique.

Le Panel Central « GrapherPanel » définit le cœur de l'application. Il permet d'afficher les courbes des fonctions à l'écran, de zoomer et dézoomer avec la molette de la souris et de déplacer le graphique à l'aide d'un glisser déposer.

B. Les contrôleurs

Les événements du JPanel situé au sud « EvalPanel » sont gérés par le contrôleur « EvalAdapter ». Celui-ci définit le comportement de l'application lors des clics sur les boutons « Eval » et les boutons permettant de changer la couleur des fonctions. Par ailleurs ce contrôleur interagit avec le panel central et le panel nord pour mettre à jour l'affichage des fonctions. Le panel nord est utilisé par ce contrôleur uniquement pour ne plus mettre à jour les coordonnées d'une fonction lorsque le champ correspondant à cette fonction est vide.

Les événements du JPanel situé à l'ouest « ActionPanel » sont gérés par le contrôleur « ActionAdapter ». Il définit les clics sur les boutons « Rafraîchir », « Réinitialiser », « + » et « - » ainsi que sur les cases à cocher « Auto Pas » et « Grille ». Ce contrôleur interagit avec le panel central afin de mettre à jour les informations concernant l'affichage du graphique.

Les événements du JPanel central « GrapherPanel » sont gérés par le contrôleur « GrapherAdapter ». Celui-ci décrit le comportement du graphique lors du mouvement de la souris, le glisser déposer, le scroll et le redimensionnement de la fenêtre. Ce contrôleur interagit avec le panel ouest afin de mettre à jour les informations du graphique dans les champs appropriés. Il utilise également le panel nord afin de mettre à jour la position de la souris sur le graphique.

C. Les classes Main et Function

La classe « Function » définit un modèle pour le traitement des données liées à une fonction. Elle contient notamment un arbre d'évaluation de la fonction, une couleur et un booléen permettant d'activer ou de désactiver son affichage. Etant donnée la simplicité des données manipulées par l'application, celles-ci sont essentiellement regroupées dans la classe « GrapherPanel ». Si le nombre de fonctions affichées à l'écran venait à augmenter il serait probablement nécessaire de séparer complètement le contenu des vues et les données du modèle.

La classe Main définie quant à elle le point d'entrée de l'application et lance une nouvelle fenêtre « Window ».