ITI 1521 - Introduction à l'informatique II - Hiver 2021

Devoir 2

Échéance: **le 8 Mars, à 23:30**

En groupes de deux étudiant(e)s.

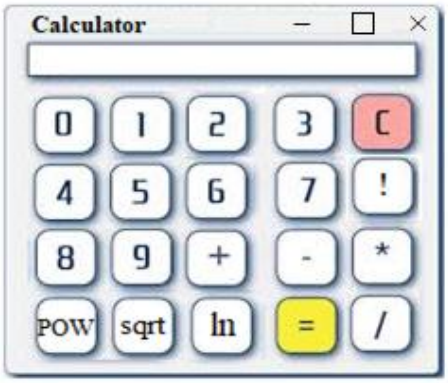
/25

Objectifs

- Réaliser une application graphique simple
- Utiliser l'héritage et l'interface
- Utiliser une fonction de rappel

Exercice 1 : (18 POINTS : 9 POINTS pour chaque classe)

Cet exercice consiste à réaliser une petite calculatrice en utilisant une interface graphique. Votre produit final aura l'allure suivante (Les couleurs ne sont pas exigées):



La calculatrice doit inviter l'utilisateur à sélectionner l'une des opérations suivantes.
Description d'opération :
+ Addition
- Soustraction
* Multiplication
/ Division
POW : Élever un nombre à une puissance
sqrt : racine carrée d'un nombre
ln : logarithme népérien d'un nombre
! factorielle d'un nombre
= égal
C efface tout

Le programme doit demander à l'utilisateur le premier opérande. Si l'opération est une opération binaire, il devrait inviter l'utilisateur pour le deuxième opérande. Ensuite, le programme doit effectuez l'opération et imprimez le résultat lors d'un appui sur la touche égale (=).

Cette calculatrice exécute séquentiellement des instructions qui peuvent être des affectations ou des évaluations. Chaque instruction doit être donnée sur une **ligne de texte** placée en haut de votre cadre

Les instructions prennent la forme suivante :
expression
L'expression est évaluée et le résultat est affiché.

Les expressions sont construites à partir des types **double** (au plus deux à la fois) avec les opérations données ci-haut (+, −, * , …). Un exemple de telle expression (expression constante) est :

12.0 / 3.0 =

L'organisation en classes est la suivante :

- **2** Classes principales :
  - **Classe Calculator :**

  Cette classe exécute les instructions, stocke les affectations et affiche les résultats des évaluations. Elle est constituée de :
- 3 variables :
  - deux de types double (*first* et *second*) pour les opérandes
  - une de type String (*oP*) pour l'opération;
- un constructeur sans arguments.
- des méthodes :
  - *void operation(String str) : (voir ci-bas)* appelée par les méthodes suivantes selon le cas. Elle doit mémoriser la valeur de la 1er opérande et faire les initialisations nécessaires pour la *seconde* et *oP* (*Opérateur binaire ou unaire d'une expression).*
    - *void add(); //addition (voir en exemple ci-dessous)*
    - *void subtract(); //soustraction*
    - *void multiply() ;//multiplication*
    - *void divide() ; // division*
    - *void factorial() ; // factorielle d'un nombre (réel)*
    - *void pow() ; //puissance*
    - *void rootSquare() ; // racine carrée d'un nombre*
    - *void nepLog() ; // logarithme népérien*
  - *void compute(); /*pour l'évaluation quand on appuie sur la touche = (effectue l'opération selon la valeur de Op)*/*
  - *void clear(); // remise à zéro quand on appuie sur la touche C*
  - *double display(); // renvoie la 2ème opérande*
  - Autres méthodes et/ou variables si nécessaire

Toutes les opérations doivent être effectuées en utilisant des doubles.
Pour le calcul de la factorielle d'un réel positif n, utiliser la formule approximative suivante :

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1+\frac{1}{12n}\right)$$

(Où e = 2.718)

Exemple de méthodes (Vous pouvez faire autrement):
*void **operation**(String str) {*
first = second; // garde la 1er operande
second = 0; // initialise et mise à jour pour la 2ème operande
oP = str;
}

*void **add**() { operation("+"); }*
*void **compute**() {*
if (oP == "+")
second = first + second;
....

○ Classe GUI

Cette classe contient la méthode `main` qui teste votre code. Elle est chargée de créer un objet de type GUI.

Créez une sous-classe de la classe **JFrame** que vous nommerez **`GUI`**. Ce sera la fenêtre principale de l'application. Son gestionnaire est un objet de type **BorderLayout.**
La classe **`GUI`** contient un attribut nommé cal de **`type Calculator:`**
•Ajoutez un constructeur à la classe **`GUI`**.
Voici ce que vous devez ajouter au constructeur.

- Donner le titre "Calculator" à votre fenêtre.
- L'appel de la méthode qui fera en sorte que l'on quitte l'application lorsque l'usager ferme la fenêtre.
- Ajout des composants au cadre : un panneau (**JPanel**) supérieur pour les instructions (Texte), et un panneau inférieur pour les boutons (chiffres et symboles). Utiliser un *GridLayout(4,4)* pour avoir vos boutons disposés en lignes et colonnes et un objet *JTextField(25)* pour le panneau supérieur.
- Ajouter (enregistrer) un gestionnaire d'évènement pour chaque bouton.

• Vous devez gérer les évènements (chaque appui sur un bouton). Il faut donc implémenter la méthode *actionPerformed (ActionEvent e)* qui doit utiliser la méthode ***getActionCommand()*** et faire appel aux méthodes la classe `Calculator`.
• Autres méthodes et/ou variables si nécessaire.

• Ajoutez une méthode principale (***main***) à la classe **`GUI`**. Cette méthode va tout simplement créer un objet de la classe **`GUI`**. Compilez votre application et assurez-vous que votre calculatrice fonctionne bien.

Exemple d'appel de méthodes (Vous pouvez faire autrement):
Dans GUI:
Calculator cal;
*public void **actionPerformed**(ActionEvent e) {*
String str = e.getActionCommand();
if (str == "+") { cal.add();}
.....

Appendix :
La classe **`String`** contient les méthodes suivantes qui peuvent vous être utiles :
***char charAt(int pos)* :** retourne le caractère se trouvant à la position spécifiée par l'index pos ;
***compareTo(char c)* :** utilisée pour comparer numériquement deux objets. Par exemple:
String str;
str.compareTo("0");

Exercice 2 : (7 POINTS : 1 POINT pour chaque méthode)

Dans cet exercice, on veut modéliser l'état et le comportement d'un Robot. On définit pour cela une classe nommée **`Robot`.**
Chaque Robot correspond à un objet qui est une instance de cette classe.
Chaque Robot:
• a un nom (attribut ***name*** : chaîne de caractères)
• a une position : donnée par l'attribut ***location*** qui est un type `Point` (classe ci-contre)
• peut **afficher** son état en détail (nom et position) à l'aide d'une méthode *display*.
• peut avancer de plusieurs pas en une seule fois grâce à une méthode ***moveTo*** qui prend deux paramètres (type int) correspondant respectivement aux nombres de pas horizontalement et verticalement.
• peut calculer sa distance (type double) par rapport à un autre Robot à l'aide d'une méthode ***distance*** qui prend comme paramètre l'autre Robot (un objet de type `Robot`).

Le nom et la position d'un Robot lui sont donnés au moment de sa création qui sont obligatoires.
- Écrire les instructions Java qui permettent de définir la classe `Robot`, **en respectant le principe de l'encapsulation des données**.
- Inclure dans votre classe `Robot,` l'accesseur `getLocation` et le modificateur `setLocation` (méthodes) qui vont permettre d'accéder et modifier respectivement l'attribut *location*.
- Inclure dans votre classe `Robot` la fonction *main* pour tester votre code : Créer par exemple un premier `Robot` et afficher son état, le déplacer et afficher son nouvel état. Créer un 2ème `Robot` et afficher son état et la distance qui le sépare du 1er `Robot`.

Exemple de sortie :
Robot : robot1
Location : (0,0)

Déplacé de 1 horizontalement et de 2 verticalement
Robot : robot1
Location : (1,2)
Robot : robot2
Location : (3,4)
Distance entre les deux robots : 2.8284271247461903

/*Classe Point.java*/

```
class Point {
      private int x;
      private int y;

 // Constructeurs
   Point(){}
   Point(int x, int y) {
     this.x = x;
     this.y = y;
   }
//Accesseurs
   int getX() { return x;    }
   int getY() {  return y;    }

 //Modificateurs
   void setX(int x) { this.x = x;    }
   void setY(int y) { this.y = y;    }

// Affichage
   public String toString() {return "(" + x + "," + y + ")";
  }
}
```

Créer et soumettre un seul fichier zip

Directives

- Créez un répertoire que vous nommerez *Devoir2_ID*, où vous remplacerez ID par votre numéro d'étudiant (celui qui soumet le devoir).
  Mettez tous les fichiers suivants dans votre répertoire compressé *Devoir2_ID.zip* pour soumission dans le campus virtuel Brightspace.

  Fichiers :
  - ✓ *README.txt*
  - ✓ *Calculator.java*
  - ✓ *GUI.java*
  - ✓ *Robot.java*

- N’oubliez pas d’ajouter des commentaires dans chaque programme pour expliquer le but du programme, la fonctionnalité de chaque méthode et le type de ses paramètres ainsi que le résultat.
- Dans le répertoire *Devoir2_ID*, créez un fichier texte nommé *README.txt*, qui devra contenir **les noms des deux étudiant(e)s**, ainsi qu’une brève description du contenu :

Nom étudiant :
Numéro d’étudiant :
Code du cours : ITI1521
Section Lab:

Fraude scolaire :
Cette partie du devoir a pour but de sensibiliser les étudiants face au problème de fraude scolaire (plagiat). Consulter les liens suivants et bien lire les deux documents:
https://www.uottawa.ca/administration-et-gouvernance/reglement-scolaire-14-autres-informations-importantes

https://www.uottawa.ca/administration-et-gouvernance/sites/www.uottawa.ca.administration-et-gouvernance/files/processus_de_traitement_des_cas_de_fraude_academique_-_nov_2019.pdf

Les règlements de l’université seront appliqués pour tout cas de plagiat.
En soumettant ce devoir :
1. vous témoignez avoir lu les documents ci-haut ;
2. vous comprenez les conséquences de la fraude scolaire.