Travail pratique : l'application REPL de calculs

420-525-RK . POO3

Ce travail est corrigé sur 100 points, pour 20% du trimestre.

Élaborer un programme OO en Java utilisant différents patrons de conception et respectant les principes SOLID pour développer une application permettant à un utilisateur d'exécuter des programmes mathématiques.

1. (15 points) Créer une application console en Java permettant à un utilisateur d'interagir en mode REPL avec votre programme à l'aide de commandes. Au démarrage, s'assurer d'indiquer le nom du programme, sa version ainsi que l'instruction pour quitter. Votre application devra supporter les fonctionnalités décrites ci-dessous.

2. (25 points) Supporter l'évaluation des expressions mathématiques. Vous devez supporter les quatres opérateurs arithmétiques usuels (`+`, `-`, `/`, `*`). Si l'expression entrée est invalide, on l'indiquera par un message clair. Au minimum, votre solution pour cette fonctionnalité utilisera le patron de l'interpréteur et du composite.

   Exemples :

   ```
   > 42
   42

   > 7/2
   3.5

   > (2 + 3) * 4
   20

   > 2 + 3 * 4
   14
   ```

3. (10 points) Un historique des expressions, valides ou non, doit être conservé pour l'utilisateur et cet historique doit être permanent au delà de la fermeture du programme et sa capacité maximale est de 20 expressions. Lorsque cette limite est atteinte, le programme évacuera la plus vieille expression de l'historique pour ajouter la plus récente.

   L'utilisateur pourra afficher l'historique avec la commande `histoire`.

   Exemples :

```
> histoire
42
7/2
(2 + 3) * 4
2 + 3 * 4
```

4. (15 points) Vous devez supporter l'opération d'affectation à une variable et les variables ainsi créées peuvent être utilisées dans des expressions subséquentes. Le nombre de variables est illimité. Un nom de variable valide doit être d'au moins un caractère, ne pas contenir d'espaces et ne peut pas débuter par un chiffre. Le résultat affiché de l'opération d'assignation est la valeur de la variable.

   Les variables ainsi créées et leur valeur peuvent être affichées à l'aide de la commande `vars` (pluriel). Pour une variable spécifiquement, utiliser `var` (singulier).

   Exemples :

```
> a = 42
42

> aPlusDeux = a + 2
44

> aPlusDeux / 2
22

> var a
a: 42

> var aPlusDeux
aPlusDeux: 44

> vars
a: 42
aPlusDeux: 44
```

5. (10 points) Vous devez supporter le chargements de paquets de constantes (un fichier contenant des constantes). Ces constantes seront des variables, comme au numéro précédent, mais elles sont immutables. Un utilisateur pourrait les utiliser dans ses expressions suite au chargement d'un paquet de constantes. Les constantes chargées sont disponibles jusqu'à la fermeture du programme, c'est-à-dire qu'il n'y a pas moyen de décharger un paquet de constantes. Votre programme doit supporter le chargements de plusieurs paquets (fichiers) de constantes. En cas de conflit dans le nom des constantes chargées, la dernière chargée est utilisée.

   Les constantes doivent être définies dans un fichier de données représentant un paquet de constantes. Concevez un format de votre cru pour le paquet de constante et fournissez un paquet de constantes nommé "constantes" qui se chargera automatique au démarrage de votre application et qui déclarera des constantes communes telles que pi, le nombre d'or phi et la constante de gravité (g) pour la Terre. Pour le chargement de paquet de constantes supplémentaires, on utilisera la commande `constantes`.

   Les constantes apparaîtront lors de l'invocation des commandes `var` et `vars`, mais elles seront visuellement différentes des variables créées par l'utilisateur par un mécanisme judicieux de votre choix. On veut marquer l'idée que ces constantes sont différentes des variables puisqu'elles ne sont pas assignables. Par exemple, on peut les préfixer ou suffixer d'un caractère spécial ou encore les séparer en 2 colonnes.

6. (25 points) L'utilisateur doit pouvoir utiliser la commande `analyse` dont le travail est d'indiquer le décompte de chaque opérateur supporté, le nombre de nombres, le nombre de variables utilisées et ainsi que le nombre de constantes utilisées. Finalement, les nombres, les variables et les constantes utilisées seront listées.

   Exemples :

```
> analyse 42
0 +      0 -      0 /      0 *
1 nombres: 42
0 variables:
0 constantes:

> x = 3
3

> y = 4
4

> analyse ( x * x + y * y ) / pi + 10
2 +      0 -      1 /      2 *
1 nombres: 10
2 variables: x y
1 constantes: pi
```

Note générale : Tentez de découpler le plus possible votre code. Si cela semble judicieux, séparez votre code en plusieurs paquetages (*package*) Java. Utilisez les patrons de conception vus lors du trimestre lorsque vous rencontrez des problèmes dont un ou des patrons encadrent la réflexion à son sujet.

Remise : fichier zip contenant votre projet nettoyé, à remettre sur Omnivox LÉA.