Algorithmique et structures de données : Mission 1 (produit)

Ivan Ahad - Jérôme Bertaux - Rodolphe Cambier - Guillaume Coutisse
Baptiste Degryse - Wojciech Grynczel - Joachim De Droogh - Thomas Grimée - Benoît Ickx

03 octobre 2014

Commentaire sur la solution apportée

Il nous a été demandé de concevoir et d'implémenter un interpréteur du mini langage PostScript. L'interpréteur prendra en entrée un fichier contenant les opérations à effectuer et produira en sortie un nouveau fichier résultant des opérations demandées. Pour réaliser cela, il nous a fallu lire le fichier, savoir lire et décrypter les informations s'y trouvant, générer un fichier de sortie et stocker les valeurs à écrire dans celui-ci.

Lire le fichier

Pour lire le fichier, nous avons utilisé les classes FileReader et BufferedReader. Grâce à elles, nous avons accès aux lignes d'opérations que nous stockons dans un arraylist. Cet arraylist contiendra toutes les lignes à exécuter via notre interpréteur.

Lire et décrypter les informations

Chaque ligne va être interprétée. Les mots (token dans le programme) seront comparés aux expressions connues telles "add", "sub", etc.

- Si l'expression n'est pas connue, c'est que le token correspond à un nombre ou à une variable prédéfinie. Ce nombre/variable sera stocké dans la *Stack*.

```
expressionStack.push(new Number(token));
expressionStack.push(new Variable(token));
```

- Si l'expression est connue, on l'effectue telle que voulu. Par exemple pour les opérateurs, on utilise les deux éléments au sommet de la *Stack* afin d'effectuer l'opération. L'élément de la *Stack* qui est passé est interprété afin que si il représente une variable, il soit remplacé par sa valeur enregistrée dans une *Map*. Ces deux éléments sont alors enlevés da la *Stack* et remplacés par le résultat de l'opération.

```
if (token.equals("add")) {
  Expression first = expressionStack.pop();
  Expression second = expressionStack.pop();
  Expression subExpression = new Add(first, second);
  expressionStack.push(new Number(subExpression.interpret(variables)));
}
else if (token.equals("sub")) {
        [...]
}
else if (token.equals("mul")) {
        [...]
}
```

Générer le fichier de sortie

A chaque fois que le token *pstack* est rencontré, le contenu de la *Stack* est imprimé grâce à un *PrintWriter* dans un fichier texte.