Algorithmique et programmation
TP 10 : Le langage Java, entrées/sorties

1 Quelques rappels

En Java, lorsqu'un programme fait des lectures avec System.in, on dit qu'il lit ses données sur son *entrée standard*. De la même manière, on parle de la *sortie standard* pour les sorties (écritures) effectuées à l'aide de System.out. On peut également faire des sorties sur la *sortie d'erreur* avec System.err.

Par défaut, sous Linux, l'entrée et les sorties standards sont connectées au terminal. En interactif, il faut taper `Ctrl-d` pour signaler la fin de l'entrée. Il est également possible de rediriger le contenu d'un fichier en utilisant les redirections du shell. Si on veut envoyer le contenu du fichier *MonBeauFichier* dans le programme *MonJoliProgramme* on tapera alors la commande

```
java MonJoliProgramme < MonBeauFichier
```

ou

```
cat MonBeauFichier | java MonJoliProgramme
```

Enfin, en Java, les fins de lignes sont repérées par un caractère particulier qui est noté `\n` dans les constantes de type caractère (`'\n'`) ou de type chaîne de caractères (`"\n"`).

2 Entrées/sorties de caractères

Normalement, lors de lectures avec la construction input.nextXXX(), les caractères d'espacement (espaces, tabulations, sauts de ligne, etc.) sont ignorés. Si on désire lire tous les caractères, il faut utiliser la fonction **int** System.in.read(). Cette fonction retourne le caractère lu, ou bien la constante -1 si la fin de l'entrée est atteinte. Pour écrire des caractères, on peut utiliser la fonction System.out.write(**int** b).

Exemple Le programme écrit sur sa sortie standard tous les caractères lus sur son entrée standard, comme la commande Unix `cat`.

```
class Cat {

    public static void main(String[] args) throws java.io.IOException {
        int c;                      // doit être de type int
        c = System.in.read();       // lecture du premier caractère
        while (c != -1) {           // test de la fin de l'entrée
            System.out.write(c);    // écriture du caractère
            c = System.in.read();   // lecture du caractère suivant
        }
        System.out.flush();         // force le vidage du tampon de sortie
    }
}
```

On trouve parfois la construction suivante où la lecture et le test sont combinés dans la même instruction :

```
int c;
while ((c = System.in.read()) != -1) {
    System.out.write(c);
}
```

NB (1) : Lors de l'utilisation de System.in.read(), il faut ajouter aux fonctions qui l'utilisent (main() dans l'exemple) l'annotation **throws** java.io.IOException.

NB (2) : Les caractères écrits avec System.out.print() peuvent être mis en attente dans un tampon avant d'être effectivement écrits sur la sortie. La dernière instruction System.out.flush() est nécessaire pour forcer le vidage de ce tampon (et donc l'affichage de tous les caractères) avant de quitter le programme.

3 Fin de l'entrée avec input.nextXXX()

Pour des entrées formatées, il est possible de tester si la prochaine lecture va bien se passer en utilisant les fonctions input.hasNextXXX (remplacer XXX par le nom du type souhaité).

Par exemple, le programme suivant lit des valeurs entières sur son entrée standard et affiche ensuite la somme de ces valeurs. La lecture s'arrête à la première erreur de lecture qui correspond normalement à la fin des entrées.

```
import java.util.*;

class Somme {

    static final Scanner input = new Scanner(System.in);

    public static void main(String[] args) {
        int val;
        int somme = 0;
        while (input.hasNextInt()) {
            val = input.nextInt();
            somme += val;
        }
        System.out.println(somme);
    }
}
```

4 Exercices

À l'exception de l'exercice 4.3, il est attendu que les programmes fonctionnent en lisant sur leur entrée caractère par caractère (et non ligne par ligne par exemple).

4.1 Numéroter les lignes

Écrire un programme qui, comme la commande Unix `nl`, recopie son entrée standard sur sa sortie standard en faisant précéder chaque ligne de texte par son numéro de ligne.

4.2 Compter les lignes, les mots et les caractères

Écrire un programme qui, comme la commande Unix `wc`, affiche le nombre de fins de lignes, le nombre de mots et le nombre de caractères du texte lu sur son entrée standard.

4.3 Moyenne

Écrire un programme qui lit un nombre inconnu de valeurs réelles sur son entrée standard. La lecture des valeurs s'arrêtera lorsque la lecture n'est plus possible. Le programme devra ensuite afficher différente statistiques sur les données, en particulier le nombre de valeurs lues, la somme, la moyenne, la variance, l'écart-type, le minimum et le maximum.

4.4 Commentaires Java

Écrire un programme qui lit un programme Java sur son entrée standard, et qui le réécrit sur sa sortie standard en ayant supprimé tous les commentaires.
Remarques :

- les commentaires Java commencent par `//` et s'étendent jusqu'à la fin de la ligne ;
- si le délimiteur `//` apparaît dans une constante de type (chaîne de) caractères, délimitée par `'` ou `"`, il ne marque évidemment pas le début d'un commentaire ;
- le programme pourra ensuite être complété pour supprimer également les commentaires multi-lignes qui s'étendent entre les délimiteurs `/*` et `*/` ;
- les commentaires ne peuvent pas être imbriqués.