Enseignement de spécialité
Informatique et sciences du numérique
Formation des IA-IPR et chargés de mission
Atelier de programmation 2

David Pichardie, Luc Bougé

Mardi 15 mars

Nous utiliserons pour cet atelier l'environnement Javascool. Dans un premier temps, il convient de vous familiariser avec cet environnement.

1. Télécharger et lancer l'application Javascool (http://javascool.gforge.inria.fr/, onglet *lancement*).
2. Choisir l'activité *Découvrir les ingrédients des algorithmes*.
3. Réaliser les exercices qui suivent, ou bien commencer par suivre le parcours d'initiation proposé par Javascool.

Ce premier atelier vous propose 3 niveaux d'activités de difficulté croissante. Choisissez un niveau en fonction de votre familiarité avec la programmation.

Activité I. Premiers pas en programmation

Nous allons programmer un système expert rudimentaire qui dialoguera avec l'utilisateur et adaptera son discours aux réponses de l'utilisateur. Voici un exemple de conversation entre le système et l'utilisateur. Dans l'exemple suivant, chaque entrée de l'utilisateur est précédée du texte `[Entrée :]`.

```
Bonjour ! Quel est votre nom ?
[Entrée :] Haddock
C'est noté. Et votre prénom ?
[Entrée :] Archibald
Êtes-vous un homme ou une femme ?
[Entrée :] homme
Désolé, je n'ai pas compris votre réponse. Êtes-vous un homme ou une femme ?
[Entrée :] un homme
Parfait. Je suis enchanté de faire votre connaissance M. Archibald Haddock.

Est-ce indiscret de vous demander votre âge ?
[Entrée :] non
```

```
Dans ce cas, quel âge avez-vous M. Haddock ?
[Entrée :] 45
Félicitations vous êtes dans la force de l'âge !

Savez-vous que je suis excellent en calcul mental ?
Proposez-moi un grand nombre s'il vous plaît.
[Entrée:] 3
Ah, c'est bien petit 3 ! Ne pouvez-vous pas me proposer
un nombre un peu plus grand ?
[Entrée:] 337
C'est mieux.
Je le note. Proposez-moi un autre grand nombre s'il vous plaît.
[Entrée:] 3456
C'est noté. Et bien... 337 fois 3456 font 1164672 !
Je vous sens impressionné...

Je suis aussi capable de compter très vite !
Jusqu'à combien voulez-vous que je compte ?
[Entrée:] 14
1
2
3
4
5
6
7
8
9
10
11
12
13
14

Merci pour ce bon moment. A bientôt Archibald.
```

Exercice I.1. Proposez un programme Javascool permettant d'obtenir ce type de conversation en tenant compte des différentes réponses de l'utilisateur (sexe, âge, etc.), et en validant le format des réponses données. Par exemple, lorsqu'on attend une réponse oui ou non, toute autre réponse doit être refusée.

Remarque. *Attention ! Pour tester l'égalité de deux chaînes, il ne faut pas utiliser l'opération ==. En effet, celle-ci teste* l'identité *des deux chaînes et non le fait qu'elles soient formées des mêmes caractères. Il faut utiliser une fonction appelée* equal.

```
String s = readString();
if (equal(s, "Haddock")) ...
```

Exercice I.2. Pour le produit de deux nombres, ne peut-on pas pousser le système à la faute en lui proposant des nombres trop grands à multiplier ? Comment filtrer les entrées de l'utilisateur pour remédier à ce problème ?

Pour éviter une saisie fastidieuse des phrases précédentes, vous pouvez copier-coller les phrases placées dans la version Javascool de cet énoncé.

Activité II. Premiers pas en programmation des tableaux

Nous nous intéressons maintenant à la manipulation des anagrammes de mots. Étant donné une collection de mots donnée sous forme d'un tableaux de chaînes de caractères, nous laissons l'utilisateur proposer un mot et nous l'informons des différents anagrammes de ce mot qui sont présents dans la collection.

Exemple. *Supposons que le tableau initial soit le suivant*[1] *:*

```
String[] choixAnagrammes =
  { "marion", "aimer", "badinage", "niche", "aspirine", "chine",
    "aspirine", "aube", "baignade", "chien", "manoir", "beau",
    "romain", "imaginer", "parisien", "migraine", "marie"}
```

L'utilisateur est invité à saisir un mot.

```
Proposez-moi un mot !
[Entrée :] beau
aube est un anagramme de beau
beau est un anagramme de beau
nombre d'anagrammes : 2
```

Afin de réaliser un tel programme, nous allons devoir tester si un mot est un anagramme d'un autre. Pour ce faire, nous vous proposons de manipuler les chaînes de caractères comme des tableaux de caractères. La fonction

```
char[] chaineVersLettres(String s)
```

vous permet d'obtenir le tableau de caractères correspondant à une chaîne de caractère. Le type **char** est un type Java primitif qui représente les caractères. Contrairement aux chaînes, on peut utiliser les opérateurs de comparaisons arithmétiques usuels pour comparer deux caractères selon leur position dans l'alphabet.

Nous vous proposons une deuxième fonction

```
String lettresVersChaine(char[] t)
```

qui permet de reformer une chaîne à partir d'un tableau de caractères pour pouvoir l'imprimer.

Pour tester si un mot est un anagramme d'un autre nous allons *trier* chacun des deux mots en utilisant l'ordre alphabétique puis vérifier si nous obtenons la même séquence de

[1] Il vous sera nécessaire de recopier cette déclaration de tableau pour faire l'exercice. Un copier-coller à partir de l'énoncé fourni dans Javascool sera certainement la façon la plus rapide de le faire.

lettres. Par exemple, si nous trions "beau" et "aube" nous obtenons dans les deux cas "abeu" ce qui nous assure que ce sont des anagrammes.

Dans un premier temps, nous vous proposons d'utiliser les fonctions de la bibliothèque standard de Java pour faire ce tri. On peut par exemple utiliser :

```
void java.utils.Arrays.sort(char[] t)
```

Exemple. *Le programme suivant affiche "abeu".*

```
char[] t = chaineVersLettres("beau");
java.util.Arrays.sort(t);
echo(lettresVersChaine(t));
```

Exercice II.1. Écrire une fonction

```
String normalise(String s)
```

qui prend en entrée une chaîne s et renvoie une nouvelle chaîne obtenue en triant par ordre alphabétique les lettres de s.

Exercice II.2. Écrire une fonction

```
boolean testAnagramme(String s1, String s2)
```

qui teste si deux chaîne s1 et s2 sont anagrammes l'une de l'autre grâce à un tri.

Exercice II.3. Proposer un programme pour interagir avec l'utilisateur selon le scénario proposé plus haut.

Nous allons maintenant tenter d'écrire notre propre fonction de tri.

```
void triSelection(char[] t)
```

Exercice II.4. Écrire tout d'abord une fonction

```
void permute(char[] t, int i, int j)
```

qui échange les caractères à la position i et j dans le tableau t. On suppose que i et j sont des indices valides dans t.

Exercice II.5. Écrire une fonction

```
int indexMin(char[] t, int i)
```

qui calcule l'indice de la plus petite lettre (pour l'ordre alphabétique) de t, entre les indices i et t.length−1.

Exercice II.6. Écrire une fonction

```
void triSelection(char[] t)
```

qui échange les lettres du tableau t pour les mettre dans l'ordre alphabétique.

Activité III. Réapprendre à compter comme à l'école

Dans cette activité, nous vous proposons de réapprendre à compter en expliquant à l'ordinateur comment poser et résoudre les opérations arithmétiques élémentaires : addition, multiplication, soustraction, division. Nous allons pour cela manipuler les nombres comme des tableaux de chiffres décimaux. Dans un deuxième temps nous vous proposons d'adapter ces programmes à l'arithmétique binaire en manipulant cette fois des tableaux de bits.

Nous manipulerons pour cette activité le type Chiffre suivant.

```
enum Chiffre { ZERO, UN, DEUX, TROIS, QUATRE, CINQ, SIX, SEPT, HUIT, NEUF }
```

Un nombre sera manipulé sous la forme d'un tableau de valeurs de type Chiffre.

Pour résoudre les opérations arithmétiques élémentaires, il faut connaître ses tables d'addition et de multiplication. Ces tables vous sont fournis sous la forme du type NombreA2Chiffres qui encapsule un chiffre des dizaines et un chiffre des unités. Ce type abstrait est munis des opérations suivantes :

- Chiffre unite(NombreA2Chiffres n) renvoie le chiffe des unités d'un nombre n.
- Chiffre dizaine(NombreA2Chiffres n) renvoie le chiffe des dizaines d'un nombre n.
- NombreA2Chiffres nb(Chiffre dizaine, Chiffre unite) fabrique un nombre à partir d'un chiffre dizaine des dizaines et unite des unités.
- NombreA2Chiffres plus(NombreA2Chiffres x, NombreA2Chiffres y) renvoie le nombre obtenu en faisant la somme des deux nombres x et y.
- NombreA2Chiffres moins(NombreA2Chiffres x, NombreA2Chiffres y) renvoie le nombre obtenu en faisant la différence des deux nombres x et y si x est plus grand que y et renvoie une erreur sinon.
- NombreA2Chiffres plus(NombreA2Chiffres x, NombreA2Chiffres y) renvoie le nombre obtenu en faisant le produit des deux nombres x et y.

Ces trois dernières opérations ignorent le chiffre des centaines.

Les valeurs de types Chiffre et NombreA2Chiffres peuvent être affichées avec les fonctions println et echo habituelles sous Javascool.

Nous fournissons de plus une fonction

```
String nombreVersChaine(Chiffre[] nombre)
```

qui permet d'afficher un nombre représenté par un tableau de chiffre.

Exemple. *echo(plus(nb(UN,TROIS),nb(DEUX,QUATRE))); provoque l'affichage 37*

Exemple. *Le tableau nombre suivant peut être affiché avec les instructions :*

```
Chiffre[] nombre = { ZERO, TROIS, HUIT };
System.out.println(nombreVersChaine(nombre));
```

pour produire l'affichage 038.

Exercice III.1. Écrire une fonction

```
Chiffre[] addition(int taille, Chiffre[] n1, Chiffre[] n2)
```

qui construit le tableau de chiffre résultant de l'addition des nombres n1 et n2, ces nombres étant représentés par des tableaux de taille taille. Le résultat de l'addition devra être exprimé dans un tableau de même taille et une erreur devra être lancée si cela n'est pas possible.

Pour réaliser un test et lancer une erreur en cas d'échec, javascool propose une fonction particulière

```
void assertion(boolean test, String message)
```

Si l'argument test est vrai, l'exécution continue sans afficher de message particulier, sinon toute l'exécution courante se termine avec le message d'erreur proposé en argument.

Exercice III.2. Écrire une fonction

```
Chiffre[] multiplication(int taille, Chiffre[] n1, Chiffre[] n2)
```

qui calcule le tableau de chiffres résultant de la multiplication des nombres n1 et n2. Ces nombres sont représentés par des tableaux de taille taille. Le résultat de la multiplication devra être exprimé dans un tableau de même taille et une erreur devra être lancée si cela n'est pas possible.

Exercice III.3. Écrire une fonction

```
Chiffre[] soustraction(int taille, Chiffre[] n1, Chiffre[] n2)
```

qui calcule le tableau de chiffre résultant de la soustraction des nombres n1 et n2. Ces nombres sont représentés par des tableaux de taille taille. Le résultat de la soustraction devra être exprimé dans un tableau de même taille et une erreur devra être lancée si n1 n'est pas supérieur ou égal à n2.

Pour cette fonction, il sera utile d'utiliser la fonction

```
boolean estPlusPetit(NombreA2Chiffres x, NombreA2Chiffres y)
```

pour tester si un nombre à deux chiffres x est inférieur ou égal à un autre nombre à deux chiffres y.

Exercice III.4. [Pour les programmeurs Java] Reprendre les questions précédentes en manipulant cette fois des tableaux de bits. Un bit pourra être représenté par un simple booléen.