IUT de Lens – BUT Info S1
SAÉ 1.01 – Implémentation d'un besoin client – étape 2

Objectifs et modalités de cette SAÉ En partant d'un besoin exprimé par un client, l'objectif de cette SAÉ est de réaliser une application qui réponde à ce besoin.

Vous devez travailler en groupe de 3 étudiants (les mêmes que pour la première partie de cette SAÉ en novembre). Vous devrez rendre le code source de vos programmes sur Moodle, dans le dépôt prévu à cet effet, avant la fin de la journée (minuit). Votre code doit être commenté et documenté pour qu'on comprenne directement tout ce qui y est fait. Vos noms de variables et de fonctions doivent aussi être choisis pour qu'on comprenne immédiatement à quoi elles servent. Vous devez impérativement tester vos programmes et garantir qu'ils fonctionnent dans tous les cas.

Évaluation La partie *codage* (liée à R1.01 Introduction au développement) comptera pour 70% de la note, la partie *web* (liée à R1.02 Développement d'interfaces web) comptera pour 20% et la partie *Anglais* (liée à R1.10 Anglais) comptera pour 10%.

1 Introduction

Dans la première partie de la SAÉ, vous avez écrit des programmes qui généraient des pages HTML qu'il fallait ensuite consulter avec un navigateur. Dans cette partie, nous allons connecter directement le navigateur et votre programme qui génère les pages. Nous allons aussi exploiter les formulaires HTML (et éventuellement les liens) pour saisir des informations. Autrement dit, alors qu'un programme Java "classique" lit des données au clavier et affiche des informations dans le terminal, les programmes que vous allez écrire vont utiliser le navigateur web pour interagir avec l'utilisateur. Un intérêt immédiat est de permettre d'améliorer la qualité de l'interface utilisateur.

Normalement, utiliser un navigateur pour l'interface utilisateur nécessite d'utiliser de la programmation client-serveur, de savoir gérer plusieurs clients en même temps et donc de gérer une forme de programmation parallèle. Vous verrez tout cela ultérieurement dans la formation. Pour l'instant, nous allons simplifier au maximum votre tâche en ne gérant qu'un seul client et en vous cachant toute la partie client/serveur et protocole HTTP.

Essais préliminaires Commencez par lire la section 3 pour comprendre comment votre programme va échanger avec le navigateur. Écrivez ensuite un petit programme de test qui va proposer à l'utilisateur un formulaire où il pourra saisir deux entiers et des boutons ou une liste déroulante pour choisir entre une addition, une soustraction, une multiplication ou une division. Lorsque l'utilisateur valide son formulaire, votre programme doit donner le résultat du calcul.

2 Travail à réaliser

Vous avez toutes et tous déjà utilisé un tableur. Il présente à l'utilisateur une grille (matrice à deux dimensions) de cellules. Une cellule peut contenir un nombre, une chaîne de caractères ou une formule. Les colonnes sont identifiées par une lettre et les lignes par un nombre. On vous demande d'utiliser vos connaissances en Java, HTML et CSS pour implanter un mini-tableur qui sera utilisable dans un navigateur. Évidemment, vous ne connaissez pas encore toutes les techniques qu'il faut utiliser pour programmer correctement un tableur. Nous allons donc devoir simplifier un peu le problème et adopter des solutions qui sont abordables à votre niveau.

Dans votre programme, vous fixerez sous forme de constantes le nombre de lignes et de colonnes de la grille. La grille sera représentée en mémoire par un tableau à 2 dimensions de chaînes de caractères (String). Si la chaîne commence par un +, un - ou un chiffre, la cellule contient un nombre. Si la chaîne commence par le signe =, elle contient une formule, qui sera représentée en notation polonaise inversée (NPI, cf. TD sur le sujet). Dans les autres cas, la cellule contient une chaîne de caractères.

Un autre tableau à 2 dimensions permettra de stocker le style CSS à appliquer à chacune des cellules, là aussi sous la forme d'une chaîne de caractères.

On vous demande de réaliser en groupe une première version du tableur, avec des fonctionnalités minimales. Ensuite, vous devrez ajouter des fonctionnalités à votre programme, en les sélectionnant dans la liste des fonctionnalités additionnelles proposées. Chaque fonctionnalité additionnelle devra être développée par un seul membre de l'équipe. Comme vous êtes normalement 3 par groupe, on s'attend à ce qu'au moins trois fonctionnalités additionnelles soient disponibles.

Vous devez utiliser au mieux les techniques que vous avez vues en cours. Vous ne devez pas utiliser d'autres techniques si vous ne les comprenez pas. Vous serez éventuellement interrogé pour vérifier que vous comprenez le code que vous aurez rendu. Il ne sert donc à rien de copier un code récupéré sur Internet et que vous ne comprenez pas.

2.1 Fonctionnalités de base

Votre programme doit impérativement permettre :

— de visualiser en HTML la grille du tableur, en appliquant à chaque cellules le style CSS qui peut lui être associé. Vous devez être en mesure d'afficher n'importe quel identifiant de colonne, en particulier s'il y a plus de 26 colonnes. Concrètement, les numéros de colonnes sont affichés en base 26 : après la colonne Z, on a la colonne AA, puis AB, ... puis ZZ, puis AAA, etc.

— de proposer à l'utilisateur un formulaire qui permet de saisir les coordonnées d'une cellule et la chaîne de caractères qu'on veut ranger dans cette cellule.

— lors de l'affichage, quand une cellule contient une formule en NPI, vous devez calculer le résultat de cette formule et l'afficher. Une formule pourra contenir évidemment des références à d'autres cellules, comme dans la formule "= A1 2 * A2 +" qui s'écrirait "=A1*2+A2" dans un tableur classique. Vous supposerez dans un premier temps que les formules ne font pas référence à une cellule qui contient une formule. A minima, vous devez gérer dans les formules des additions, soustractions, multiplications et divisions de nombres réels. Si le calcul d'une formule génère une erreur, vous devez afficher dans la cellule un code d'erreur tel que "#DIV/0 !" en cas de division par zéro ou "#VALUE ?" quand une opération n'a pas de sens (comme additionner un nombre et une chaîne de caractères).

— de proposer à l'utilisateur des boutons (dans le formulaire) qui permettent de modifier le style CSS d'affichage d'une cellule. A minima, il y aura un bouton pour mettre en gras, italique ou style normal. Il y aura aussi des boutons pour choisir la couleur de fond et la couleur du texte (au moins noir, blanc, rouge, marron, vert, jaune, bleu et violet). Il doit être possible de combiner ces différents éléments de style (par exemple gras, italique, fond jaune et texte noir en même temps).

2.2 Fonctionnalités additionnelles

On vous propose ci-dessous un certain nombre de fonctionnalités additionnelles. Certaines sont faciles, d'autres plus difficiles, voire infaisables dans le temps dont vous disposez. Vous devez en implémenter autant que possible dans le temps qui vous est imparti. Attention cependant, il faut privilégier la qualité à la quantité. Ne commencez pas à implémenter une fonctionnalité si une autre n'est pas implémentée de manière satisfaisante.

Chaque fonctionnalité additionnelle doit être implémentée par un des membre du groupe, qui assume la responsabilité pleine et entière de son développement. Vous devez indiquer au début de votre code quelles fonctionnalités ont été développées, et par qui.

Les fonctionnalités sont listées dans le désordre, à vous d'évaluer leur difficulté et lesquelles sont abordables pour vous.

1. étendre les opérations possibles dans les formules : sqrt, cos, sin, tan, exp, log, pow, mod (modulo), div (divison entière). Par exemple, Si A1, A2, A3 contiennnent les coefficents d'une équation du second degré à une inconnu, on doit pouvoir écrire la formule "= A2 2 pow 4 A1 A3 * * - sqrt" pour calculer le discriminant.
2. étendre les opérations possibles dans les formules aux opérations booléennes : and, or, not, xor, <, <=, >, >=, ==, !=
3. étendre les opérations possibles dans les formules à des calculs sur des zones rectangulaires : somme, moyenne, min, max. Par exemple, la formule "= A1:Z26 moyenne" devra calculer la moyenne des cellules comprises entre A1 et Z26.
4. permettre à des formules de faire référence à des cellules qui contiennent elles-même des formules. Dans un premier temps, vous supposerez qu'on peut calculer les formules dans l'ordre d'affichage des cellules de la grille.
5. trouver le bon ordre pour évaluer les formules. Pour cela vous devez effectuer un tri toppologique sur un graphe orienté dont les sommets sont les cellules contenant une formule, et qui contient un arc depuis une cellule X vers une cellule Y lorsque la formule de la cellule Y fait référence à la cellule X (auquel cas, il faut calculer la valeur de X avant d'évaluer Y). Voir https://en.wikipedia.org/wiki/Topological_sorting pour plus d'informations et l'algorithme à utiliser.
6. permettre à l'utilisateur d'annuler ses dernières modifications (bouton undo). Il suffit de gérer une pile où l'on stocke la valeur qu'avait une cellule avant de la modifier.
7. permettre à l'utilisateur d'annuler ses annulations (bouton redo)
8. permettre d'encadrer des cellules
9. permettre de désigner une zone rectangulaire (par exemple S2:Z7) pour l'application d'un style CSS
10. gérer plusieurs onglets, chaque onglet contenant une grille du tableau (c'est une troisième dimension du tableau). Chaque onglet doit avoir son propre nom, qu'on doit pouvoir modifier. On n'affiche qu'un onglet à la fois mais on doit pouvoir basculer d'un onglet à un autre.
11. ajouter des styles supplémentaires tels que alignement à gauche, à droite ou au centre, changement de police, ...
12. permettre le tri d'une zone de cellules comme dans un tableur, potentiellement selon plusieurs critères (par exemple, d'abord selon la colonne Z par ordre croissant, puis en cas d'égalité, par ordre décroissant selon la colonne X).
13. rechercher un mot dans la grille et "surligner" chaque cellule qui contient ce mot
14. rechercher un mot dans la grille et le remplacer par un autre
15. permettre un copier/coller ou un couper/coller. Un champ de formulaire permettra de désigner la zone à copier ou couper (par exemple A1:B10), une autre champ indiquera la première cellule de la destination (par exemple C1) et un bouton permettra de choisir entre copier et coller.
16. toute autre fonctionnalité d'un tableur qui vous semble abordable (nécessite la validation des enseignants).

3 Dialogue avec le navigateur

Concrètement, vous allez écrire un programme en Java qui devra respecter des règles simples. Ce programme sera en fait un nano-serveur web. Vous exécuterez ensuite votre programme qui attendra une requête de votre navigateur. Il faudra donc ouvrir votre navigateur et taper l'URL http://localhost:8080. À ce moment, votre navigateur va contacter votre programme Java, qui génèrera une page HTML qui sera envoyée au navigateur qui l'affichera. Si cette page contient un formulaire ou des liens, la validation du formulaire ou le clic sur le lien provoquera l'envoi d'une nouvelle requête par le navigateur et votre programme devra générer une nouvelle page HTML à transmettre au navigateur.

Votre code Java devra suivre le squelette suivant :

```
import java.io.*;
import java.util.*;
import navigator.*; // <<<<<

class NomDeVotreProgramme {
  // la variable nav représente le dialogue avec le navigateur
  Navigator nav = new Navigator(); // <<<<<

  void run() {
    // début d'une page web et attente d'une connexion du navigateur
    nav.beginPage();

    // envoi du code HTML (très simplifié sur cet exemple)
    nav.println("<html>");
    nav.println("<header><title>Test</title></header>");
    nav.println("<body>Ceci est un test</body>");
    nav.println("</html>");

    // fin de la page web
    nav.endPage();
  }

  public static void main(String[] args) {
    new NomDuProgramme().run();
  }
}
```

Comme vous le voyez sur l'exemple, avant de générer une page HTML, vous devez appeler la procédure `nav.beginPage()`. Concrètement, cette procédure attend que le navigateur envoie sa requête (et il ne se passe rien tant qu'il ne le fait pas). Elle analyse ensuite cette requête, puis met à votre disposition certaines informations telles que les données d'un formulaire. Après chaque appel à `nav.beginPage()`, votre programme doit envoyer le code d'une page HTML au navigateur en utilisant la variable `nav` qui se comporte comme un **fichier** en écriture seule. Les procédures `nav.println(String s)` et `nav.print(String s)` permettent d'écrire dans ce fichier, et donc de transmettre le code HTML au navigateur.

Une fois que vous avez transmis tout votre code HTML, vous devez appeler la fonction `nav.endPage()` qui signale au navigateur que la page est complètement transmise.

Voici un exemple, qui transmet au navigateur un formulaire. Pour simplifier, le code HTML n'est pas complet.

```
void run() {
  nav.beginPage();
  // ici, on utilise les triples guillemets doubles pour écrire une chaîne de
  // caractères sur plusieurs lignes et éviter d'échapper les guillemets doubles
  nav.println("""
<html>
  <body>
    <form method="POST">
      Votre nom : <input type="text" name="nom">
      Votre prénom : <input type="text" name="prenom">
      <input type="submit" name="Valider">
    </form>
  </body>
</html>
""");
```

```
  nav.endPage();

  // ...
}
```

Quand l'utilisateur soumet le formulaire, votre programme doit générer une nouvelle page web en utilisant les données du formulaire. Vous pouvez tester si un champ de formulaire a été transmis en appelant `nav.containsKey("nomDuChamp")` qui retourne vrai ssi le champ existe. Pour obtenir la valeur du champ de formulaire, vous utiliserez `nav.get("nomDuChamp")` qui retourne cette valeur sous forme d'une chaîne de caractères (String) ou qui retourne la valeur `null` si ce champ n'existe pas. Voici un exemple, qui est la suite du code précédent.

```
  // ...
  nav.beginPage();
  if (nav.containsKey("nom") && nav.containsKey("prenom")) {
    nav.print("""
<html>
  <body>
    <p>
    Vous vous appelez
""");
  nav.print(nav.get("prenom")+" "+nav.get("nom"));
  nav.print("""
  </p>
  <a href="/">suite</a>
  </body>
</html>
""");
  }
  else {
    nav.print("<html><body><p>Erreur : formulaire non rempli</p>" +
              "<a href=\"/\">suite</a></body></html>");
  }

  nav.endPage();
}
```

Vous comprenez qu'on peut enchaîner de la sorte autant de générations de pages que l'on souhaite, en utilisant aussi des boucles si nécessaire.

Enfin, vous pouvez appeler `nav.getPath()` qui fournit le chemin utilisé dans l'URL transmise par le navigateur. Attention, pour tester l'égalité des chaînes en Java, il faut écrire `s1.equals(s2)` à la place de `s1 == s2`. Cela permet de gérer des liens hypertextes comme dans l'exemple ci-dessous.

```
while(true) {
    nav.beginPage();
    if (nav.getPath().equals("/")) {
        nav.print("""
<html>
  <body>
    <a href="/page1">page1</a> <a href="/page2">page2</a>
  </body>
</html>
""");
    }
    else if (nav.getPath().equals("/page1")) {
        nav.println("<html><body>Page 1 <a href=\"/\">home</a></body></html>");
    }
    else if (nav.getPath().equals("/page2")) {
        nav.println("<html><body>Page 2 <a href=\"/\">home</a></body></html>");
    }
    else {
        nav.println("<html><body>La page demandée n'existe pas.</body></html>");
    }
    nav.endPage();
}
```

Pour terminer, votre code Java ne pourra générer que du code HTML. Mais votre code HTML pourra faire référence à d'autres fichiers (fichiers CSS, fichiers images, ...). Le transfert de ces fichiers est géré pour vous par la procédure `nav.beginPage()`. Évidemment, il n'est pas question que votre programme permette à une personne mal intentionnée de télécharger tous les fichiers de votre machine. Le programme n'acceptera de fournir que les fichiers présents dans le répertoire courant, et uniquement si ces fichiers sont dans la liste des fichiers autorisés au téléchargement. Pour autoriser un fichier "style.css" à être téléchargé, vous devez écrire `nav.allowDownload("style.css")` au début de votre programme.

Par exemple, si vous avez besoin d'un fichier `style.css` et `image.png`, vous devez mettre ces deux fichiers dans le répertoire de votre programme et appeler au début de votre code

```
nav.allowDownload("style.css");
nav.allowDownload("image.png");
```

Pour des raisons techniques, vous ajouterez à la fin de votre code la génération d'une page web qui ne contient aucun lien. Voici le code qui génère cette page finale. Bien sûr, votre programme ne se terminera que quand cette page s'affichera dans le navigateur.

```
  nav.beginPage();
  nav.println("<html><body>Fin du programme</body></html>");
  nav.endPage();
```

Pour compiler votre code, vous devez télécharger sur Moodle le fichier `nav.jar` et le placer dans le même répertoire que vos fichiers *.java. Pour compiler, vous écrirez :

```
javac -cp .:nav.jar NomDeVotreProgramme.java
```

Pour exécuter votre programme, vous écrirez :

```
java -cp .:nav.jar NomDeVotreProgramme
```

Sous Windows, il faut écrire `-cp ".;nav.jar"` à la place de `-cp .:nav.jar`.

Dernière remarque essentielle : le nano-serveur web que vous utiliserez n'est pas prévu pour gérer toute la sécurité requise par un véritable serveur web. Vous ne devez donc en aucun cas l'utiliser comme un véritable serveur web. Il ne doit être utilisé que depuis le poste local.

4 Boîte à outils Java et autres indications

1. Pour découper une chaîne de caractères *formule* en mots qui seront stockés dans un tableau *mots*, vous pouvez écrire :

   ```
   String[] mots = formule.split(" ");
   ```

2. Pour convertir une chaîne de caractères en nombre (entier ou réel), vous pouvez reprendre le code utilisé dans le TD sur la notation polonaise inversée et le généraliser.
3. Pour tester si une chaîne *s* contient un mot *mot*, vous écrirez `s.contains(mot)`.
4. Pour remplacer dans une chaîne *s* un mot *mot* par le mot *remplacement*, vous écrirez `s.replace(mot,remplacement)`.
5. Pour obtenir le i-ème caractère d'une chaîne *s*, vous écrirez `s.charAt(i)`. On obtient le nombre de caractères dans une chaîne *s*, on écrit `s.length()`.
6. `String s=Integer.toString(x)` et `String s=Double.toString(x)` permettent de convertir un nombre x (entier ou réel selon le cas) en une chaîne de caractères s.
7. `int x=Integer.parseInt(s)` et `double x=Double.parseDouble(s)` permettent de convertir une chaîne de caractères s en un nombre x (entier ou réel selon le cas).