TP Java ME : correction

Les points de correction sont indiqués en vert

Installation du "Wireless Toolkit 2.5.2" de Sun

L'environnement de Sun pour commencer à faire du développement pour Java Micro Edition s'appelle le Wireless Toolkit. Nous utilisons dans ce TP la version 2.5.2. Dans tout cet énoncé on le désignera par WTK (son nom habituel).
0°) Préparez vous un répertoire de travail. Il devra contenir :

- un sous répertoire de nom `Telecharger` qui contiendra les données téléchargées.
- un sous répertoire de nom `WTK` qui contiendra le WTK installé.

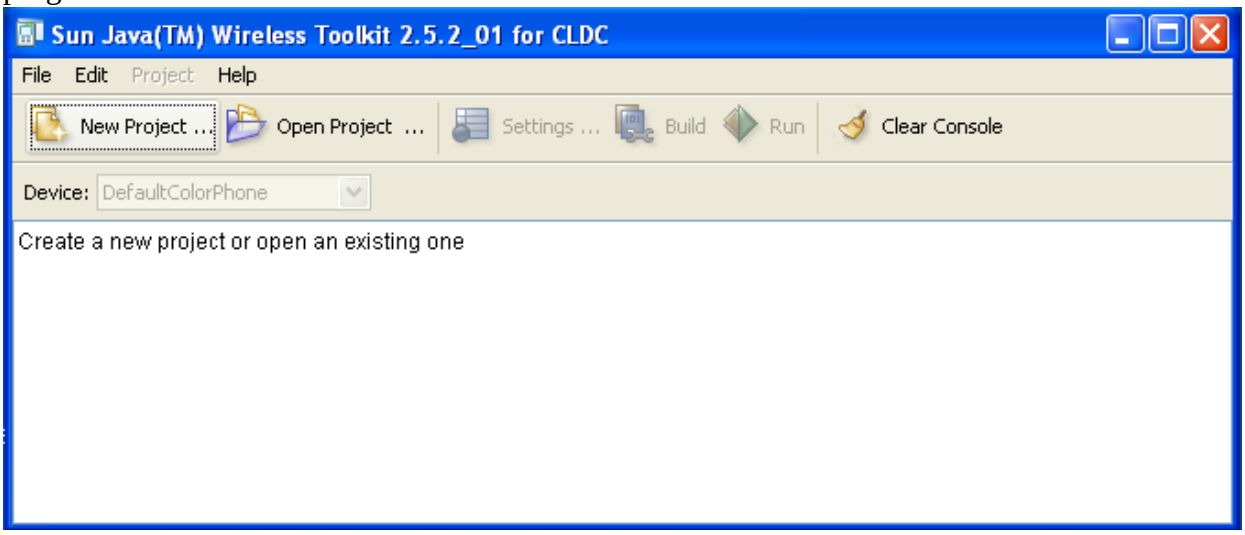
Il est bon de bien ranger les choses ! (cf. TPs précédents).

1°) Télécharger cet environnement à partir de l'URL `http://java.sun.com/products/sjwtoolkit/index.jsp`
On utilisera le lien "Sun Java Wireless Toolkit 2.5.2_01 for CLDC ". Il y a plusieurs pages web à parcourir, lisez rapidement ces pages ! Vous devez finir par télécharger le programme `sun_java_wireless_toolkit-2.5.2_01-win.exe`. Mettez-le dans votre répertoire `Telecharger`.

2°) Lancer l'installation de cet environnement. Il y a plusieurs écrans qui apparaissent lors de cette installation. On placera cet environnement dans le répertoire `WTK`.

Pas de grosses difficultés jusque là. Sinon remarquer que le programme d'installation du WTK trouve d'abord par lui-même où est Java SE sur le disque et il ne faut pas changer cet emplacement (à moins d'avoir un autre Java SE sur le disque et de connaître son emplacement). Ce n'est qu'ensuite qu'il faut indiquer où placer le WTK (sous notre répertoire de travail dans le dossier `WTK`).

3°) Lancer l'exécution de cet environnement. Pour lancer cet environnement aller dans le répertoire ou vous avez installé le WTK, puis dans son sous répertoire `bin` et lancer le programme `ktoolbar.exe`. Vous devez obtenir la fenêtre ci-dessous :



Remarque : Dans une ancienne version un raccourci était installé dans le menu des programmes. et on pouvait lancer donc cet environnement par :
démarrer | Programmes | Sun Java (TM) Wireless Toolkit 2.5.2 for CLDC | Wireless Toolkit 2.5.2 (sous Windows XP)

L'environnement "Wireless Toolkit"

4°) Quand cet environnement a été lancé, exécuter les démonstrations proposées par exemple : `Demos`, `CityGuide`, `UIDemo` par **OpenProject** puis **Run**. Votre souris ne doit servir qu'à appuyer sur les touches de l'émulateur.
5°) Changer de **Device** (de **DefaultColorPhone** à **QuertyDevice**) et tester certaines midlets.

Avec ces questions, on se familiarise avec le WTK. Remarquer qu'on a évidemment le code source de toutes les démos amenées par le WTK.

Le client Java ME `ClientCompteBancaireMIDlet`

6°) Vous allez écrire une MIDlet qui sera un client permettant d'atteindre et de faire des opérations sur un compte bancaire.
6.1) Pour cela cliquer **New Project**, Entrer un nom de projet (`EDJavaMECNAM`) et un nom de classe (`ClientCompteBancaireMIDlet`) puis cliquer **Create Project**. Accepter les configurations par défaut dans la fenêtre suivante : cliquer **OK**.

Il ne faut pas faire de faute de frappe dans la plupart des noms car l'environnement WTK prépare déjà le `.jad` qui contient le nom de la MIDlet. Utiliser plutôt les copier-coller avec l'énoncé (CTRL-C, CTRL-V). Par exemple, le nom de la MIDlet indiqué à cette étape, doit être (évidemment) le même que celui du fichier Java que vous aller écrire question suivante à la casse près.

6.2) Vérifier qu'un nouveau projet a été créé : le répertoire `EDJavaMECNAM` a été créé sous `C:\Documents and Settings\`*`nomDUtilisateurDeVotreMachine`*`\j2mewtk\2.5.2\apps`.
Dans ce répertoire `EDJavaMECNAM` ont été créés plusieurs fichiers et sous-répertoires.

6.3) Dans un éditeur de texte, éditer la MIDlet ci dessous dans le fichier `ClientCompteBancaireMIDlet.java` :
fichier `ClientCompteBancaireMIDlet.java`

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import java.io.*;

public class ClientCompteBancaireMIDlet extends MIDlet implements CommandListener,
Runnable {
  private Form mMainForm, connectForm;
  private ChoiceGroup cg;
  private Command cmdExit, cmdTransaction;
  private TextField tf;
  private String reponseDeLaServlet = "";

  // les 3 méthodes abstraites de MIDlet
  public void destroyApp(boolean condition){}
  public void pauseApp(){}
  public void startApp(){
```

```
    Display.getDisplay(this).setCurrent(mMainForm);
  }
  // La methode de l'interface CommandListener
  public void commandAction(Command c, Displayable d) {
    if (c == cmdExit) {
      destroyApp(true);
      notifyDestroyed();
    } else if (c == cmdTransaction) {
      Thread t = new Thread(this);
      t.start();
      Display.getDisplay(this).setCurrent(connectForm);
    }
  }

  public void run() {
    envoiRequete();
  }

  public void envoiRequete() {
    HttpConnection hc = null;
    InputStream in = null;
    String url = "http://localhost:8080/JavaMeSMB111/GereCompteBancaireServlet";

    try {
      int choix = cg.getSelectedIndex();
      String urlComplete = url;
      String valeur = null;
      switch (choix) {
      case 0 :
        // A FAIRE 1
        // Compléter urlComplete pour pouvoir obtenir le solde du compte bancaire
        // On pourra s'inspirer du code lignes suivantes
        break;
      case 1 :
        valeur = tf.getString();
        urlComplete += "?operation=debiter&quantite="+valeur;
        break;
      case 2 :
        valeur = tf.getString();
        // A FAIRE 2
        // Compléter urlComplete pour pouvoir crediter valeur euro sur le compte
bancaire
        // On pourra s'inspirer du code lignes précédentes !
        break;
      }
      System.out.println(urlComplete);

      hc = (HttpConnection)Connector.open(urlComplete);
      in = hc.openInputStream();

      int contentLength = (int)hc.getLength();
      byte[] raw = new byte[contentLength];
      int length = in.read(raw);

      in.close();
      hc.close();

      // traite la réponse
      reponseDeLaServlet = new String(raw, 0, length);
      mMainForm = null;
      mMainForm = new Form("MIDlet JMF pour manipuler le compte bancaire");
      remplirMainForm();
      Display.getDisplay(this).setCurrent(mMainForm);
    }  catch (IOException ioe) {
      System.out.println("Pb dans l'envoi de la requête : " + ioe);
    }
  }
```

```
  private void remplirMainForm() {
    cg = new ChoiceGroup("Sur votre compte bancaire voulez vous : ",
ChoiceGroup.EXCLUSIVE);
    mMainForm.append(cg);
    cg.append("Obtenir le solde\nou bien", null);
    cg.append("Retirer\nou", null);
    cg.append("Crediter", null);
    tf = new TextField("la somme de : ", "", 7, TextField.DECIMAL );
    mMainForm.append(tf);
    mMainForm.append(new StringItem(null, "euro"));
    mMainForm.append(new StringItem(null, reponseDeLaServlet));

    cmdExit = new Command("Exit", Command.EXIT, 0);
    mMainForm.addCommand(cmdExit);
    cmdTransaction = new Command("Lancer transaction", Command.EXIT, 0);
    mMainForm.addCommand(cmdTransaction);
    mMainForm.setCommandListener(this);
  }

  private void remplirConnectForm() {
    StringItem messageLabel = new StringItem(null, "En cours de connexion (eh
oui !)");
    connectForm.append(messageLabel);
    connectForm.setCommandListener(this);
  }

  public ClientCompteBancaireMIDlet() {
    mMainForm = new Form("MIDlet pour manipuler le compte bancaire");
    remplirMainForm();
    connectForm = new Form("Connexion à la servlet");
    remplirConnectForm();
  }
}
```

6.4) Sauvegarder cette MIDlet `ClientCompteBancaireMIDlet.java` dans ce répertoire (`C:\Documents and Settings\`*`nomDUtilisateurDeVotreMachine`*`\j2mewtk\2.5.2\apps\src`)

6.5) Construire la MIDlet par l'environnement WTK en cliquant **Build**. Lancer l'exécution en cliquant **Run**. Vous devez obtenir l'interface graphique ci dessous :

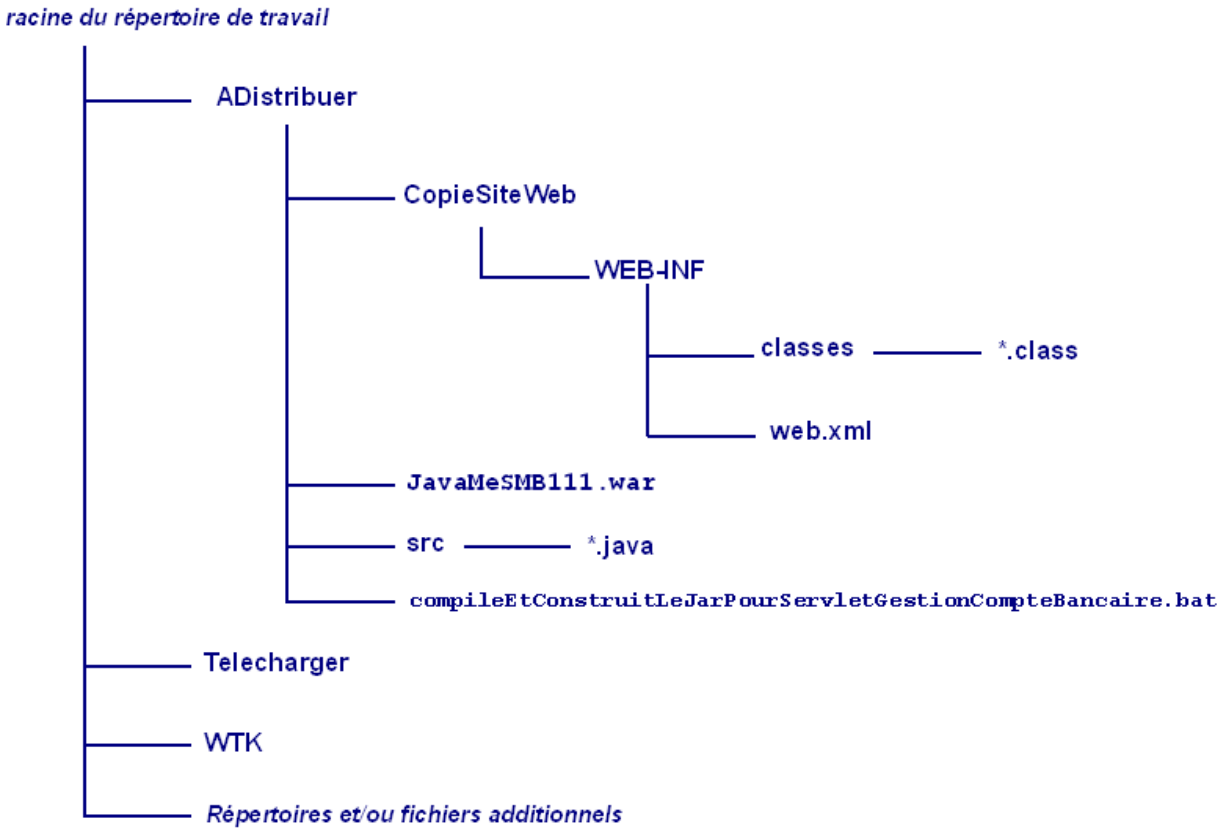
Evidemment toute la partie réseau ne fonctionne pas encore : c'est l'objet de la suite de ce TP.

Pas de gros problème jusque là. Le code de la MIDlet est intéressant à lire (et à comprendre) entre autre pour la construction de son IHM, le traitement de la partie réseau dans une thread lancée lors de l'appui sur la `Command "Lancer transaction"`, etc. Enfin rappelons que "Evidemment toute la partie réseau ne fonctionne pas encore : c'est l'objet de la suite de ce TP " !

Le serveur `GereCompteBancaireServlet`

Le programme qui va recevoir les requêtes de la MIDlet et y répondre est une servlet.

7°) Récupérer à l'URL
`http://cedric.cnam.fr/~farinone/SMB111/JavaME/`
le fichier `ADistribuer.zip`. Ouvrer ce fichier dans votre répertoire de travail. Vous devez désormais avoir une arborescence comme :



Vous avez donc dans le répertoire `ADistribuer` :
- le `.war` de cette application web
- une copie du site web qui sera finalement mise sous tomcat
- les sources `*.java` pour votre servlet
- un script qui permet de construire cette application web

En fait on ne va utiliser que le `.war` (à déployer) et le code de la servlet (à étudier).

8°) Lancer tomcat.
Déployer le fichier `JavaMeSMB111.war` dans tomcat (utiliser Tomcat Manager cf. TP sur les servlets/JSP).

9°) Lisez le code de la servlet `GereCompteBancaireServlet` (fichier `GereCompteBancaireServlet.java`) ainsi que le fichier `web.xml`.
9.1) En déduire l'URL qu'il faut utiliser pour obtenir le solde du compte bancaire.

Au vue du contenu du fichier `web.xml` et du nom de l'application web `JavaMeSMB111`, c'est l'URL :
`http://localhost:8080/JavaMeSMB111/GereCompteBancaireServlet?operation=getSolde`

9.2) De même en lisant le code de la servlet `GereCompteBancaireServlet`, déduire l'URL pour débiter 52 euro.

c'est l'URL :
```
http://localhost:8080/JavaMeSMB111/GereCompteBancaireServlet?operation=debiter&quantite=52
```

9.3) De même en lisant le code de la servlet `GereCompteBancaireServlet`, déduire l'URL pour créditer 52 euro.

c'est l'URL :
```
http://localhost:8080/JavaMeSMB111/GereCompteBancaireServlet?operation=crediter&quantite=52
```

9.4) Vérifier vos réponses aux questions 9.1, 9.2, 9.3 en accédant à ces URLs par un navigateur.

Est alors affiché dans le navigateur la réponse de la servlet lorsqu'on se connecte à ces URLs. La réponse est en format MIME `text/plain`. Ce n'est pas très "joli", mais c'est original et surtout ce sera le bon type de réponse pour le client MIDlet.

L'architecture client-serveur (MIDlet client, servlet serveur)

10°) Editer la MIDlet `ClientCompteBancaireMIDlet.java` (qui se trouve dans le répertoire (`C:\Documents and Settings\`*`nomDUtilisateurDeVotreMachine`*`\j2mewtk\2.5.2\apps\src`).
Aux vues des réponses aux questions 9.x, compléter dans cette MIDlet les parties `A FAIRE 1` et `A FAIRE 2`.

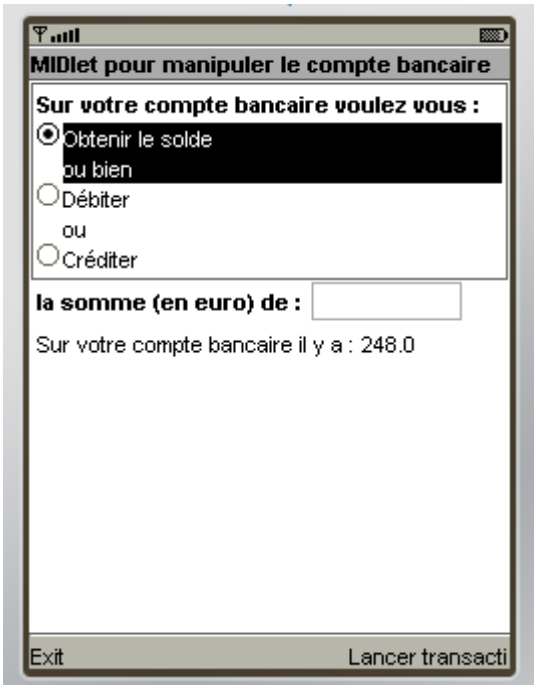
Il faut écrire des lignes comme :
```
// A FAIRE 1
// Compléter urlComplete pour pouvoir obtenir le solde du compte bancaire
urlComplete += "?operation=getSolde";
```

et

```
// A FAIRE 2
// Compléter urlComplete pour pouvoir crediter valeur euro sur le compte bancaire
urlComplete += "?operation=crediter&quantite="+valeur;
```

11°) Reconstruisez la MIDlet (par Build) et lancer l'exécution de cette MIDlet (par Run). Tester l'obtention du solde et des opérations créditer et débiter.

On obtient des réponses comme :



Vous devez désormais pouvoir manipuler votre compte bancaire par un client web ou un téléphone portable.

Et oui ! C'est une architecture distribuée où les données sont centralisées. Comme l'indique les remarques et questions 12°) et 13°) suivantes.

12°) Si votre compte bancaire est en fait géré par la servlet de l'ordinateur voisin, modifier votre MIDlet pour qu'elle puisse manipuler ce compte bancaire. Vous aurez besoin du numéro IP de l'ordinateur voisin.

13°) De même, puisque votre servlet gère un compte bancaire, en supposant que celui ci est celui de votre voisin, demander lui de gérer son compte en se connectant sur votre servlet. Vous devrez, pour cela, lui communiquer le numéro IP de votre machine et il devra modifier le code de sa MIDlet.

Conclusion

Dans les questions 12°) et 13°) on obtient un véritable architecture client-serveur où le client et le serveur sont sur deux machines différentes. On pourrait même envisager une architecture 3-tiers si la sauvegarde de la donnée pérenne était sur une troisième machine (avec une connexion de la servlet sur une base de données déposée sur cette troisième machine par exemple).

fin du TP.