

Java Avancé - Cours 1

Solution des exercices

Liste des exercices:

Exercice 1.1

Q1 On utilise les deux constructeurs suivants :

```
java.io.InputStreamReader o_1 = new InputStreamReader(socket.getInputStream());
java.io.BufferedReader o_2 = new BufferedReader(o_1);
```

Q2 La classe `java.io.BufferedReader` utilise une mémoire tampon (buffer en anglais). Si je demande à lire un caractère par `java.io.InputStreamReader` je sollicite tout le système pour ne lire qu'un caractère. Si je demande à lire un caractère par la classe `java.io.BufferedReader` si le buffer est vide il est rempli avec n caractères en sollicitant tout le système, si il n'est pas vide, le système n'est pas sollicité et simplement le caractère suivant m'est renvoyé.

Q3 Le listing suivant permet de récupérer la page `etudiants.masters.epita.net` :

```
import java.net.Socket;
import java.net.UnknownHostException;
import java.io.PrintWriter;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

class ClientWeb {
    public static void main(String[] args) throws IOException {
        Socket socket = null;
        PrintWriter out = null;
        BufferedReader in = null;
        try {
            socket = new Socket ("etudiants.masters.epita.net", 80);
            out = new PrintWriter(socket.getOutputStream(), true);
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            out.println("GET / HTTP/1.1");
            out.println("Host: "+args[0]+":80");
            out.println(""); // la ligne vide
            String userInput = null;
            while ((userInput = in.readLine()) != null) {
                System.out.println(userInput);
            }
        } catch (UnknownHostException e) {
            System.err.println("Don't know about host: "+args[0]);
            System.exit(1);
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for "
                + "the connection to: "+args[0]);
            System.exit(1);
        }
        out.close();
        in.close();
        socket.close();
    }
}
```

Q4 La section Try diffère :

```

URL url = null;
try {
    url = new URL(args[0]);
    socket = new Socket (url.getHost(), 80);
    out = new PrintWriter(socket.getOutputStream(), true);
    in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
    out.println("GET "+url.getPath()+" HTTP/1.1");
    out.println("Host: "+url.getHost()+":80");
    out.println(""); // la ligne vide
    String userInput = null;
    while ((userInput = in.readLine()) != null) {
        System.out.println(userInput);
    }
}

```

Exercice 1.2

Voici le listing qui donne à la fois la solution des questions 1 et 2. L'envoi de Bonjour ! se fait via la variable out et la lecture des données émises par le client par la variable in.

```

class Server {
    public static void main(String[] args) {
        java.net.ServerSocket serverSocket = null;
        try { // creer un objet serveur
            serverSocket = new java.net.ServerSocket(8888);
        } catch (java.io.IOException e) {
            System.err.println("Impossible de lancer le serveur ");
            System.exit(-1);
        }
        java.net.Socket clientSocket = null;
        try { // se mettre en attente d'une connexion
            clientSocket = serverSocket.accept();
        } catch (java.io.IOException e) {
            // il y a eu une erreur : relacher les ressources
            System.err.println("Accept a echoue.");
            try {
                serverSocket.close();
            } catch (java.io.IOException ee) {
            }
            System.exit(-1);
        }
        java.io.BufferedReader in = null;
        java.io.PrintWriter out = null;
        try {
            // envoyer bonjour
            out = new java.io.PrintWriter(clientSocket.getOutputStream(), true);
            out.println("Bonjour !");
            // lire ce que le client envoie
            in = new java.io.BufferedReader
                (new java.io.InputStreamReader(clientSocket.getInputStream()));
            String userInput = null;
            while ((userInput = in.readLine()) != null) {
                System.out.println(userInput);
            }
        } catch (java.io.IOException e) {
            System.err.println("Lecture impossible.");
        } finally {

```

```

        // liberere les ressources
        if (in!=null) {
            try {
                in.close();
            } catch (java.io.IOException ee) {
            }
        }
        if (out!=null) {
            try {
                out.close();
            } catch (java.io.IOException ee) {
            }
        }
    }
    try {
        serverSocket.close();
    } catch (java.io.IOException e) {
    }
}
}

```

Une fois le serveur lancé si on tape sur la même machine, dans un navigateur l'url `http://127.0.0.1:8888` on obtient (ici par exemple pour le navigateur firefox):

```

GET / HTTP/1.1
Host: 127.0.0.1:8888
User-Agent: Mozilla/5.0 (X11; U; Linux i686; rv:1.7.3) Gecko/20041001 Firefox/0.10.1
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,
image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive

```

À chaque fois que l'on va chercher une URL le navigateur donne ces informations. On remarque que la machine est configurée pour langue anglaise, que le système d'exploitation d'origine est linux et que le navigateur est firefox. Les cookies, ainsi que certaines données d'authentifications sont également transmis par ce mécanisme.

Exercice 1.3

Q1 Nous allons tout d'abord créer une classe pour gérer la connexion à la base de données :

```

package ex3;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

class DataBaseConnection {
    static public Connection conn;
    /**
     * Ouvrir statiquement la connexion a la base
     */
    static {
    try {

```

```

        Class.forName("com.mysql.jdbc.Driver");
        //Class.forName("org.gjt.mm.mysql.Driver");
    } catch (java.lang.ClassNotFoundException e) {
        System.err.println("Driver non trouve");
    }
}
try {
    Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/mysql?user=root");
} catch (SQLException ex) {
    // handle any errors
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
}
}
}
}

```

Ainsi il nous suffit maintenant d'utiliser la variable statique `DataBaseConnection.conn` pour se connecter à la base de données :

```

package ex3;

import java.sql.SQLException;

class Personne {
    int id;
    String login;
    String nom;
    String prenom;

    public Personne(int id) {
        java.sql.Statement stmt = null;
        java.sql.ResultSet rs = null;

        try {
            stmt = DataBaseConnection.conn.createStatement();
            rs = stmt.executeQuery("select login, nom, prenom from personnes where id=3");

            if (rs.next()) {
                nom = rs.getString("nom");
                prenom = rs.getString("prenom");
                login = rs.getString("login");
            }
        } catch (SQLException e) {
            System.out.println("SQLException: " + e.getMessage());
        } finally {
            if (rs != null)
                try { rs.close(); } catch (SQLException sqlEx) { }
            if (stmt != null)
                try { stmt.close(); } catch (SQLException sqlEx) { }
        }
    }
}

```

Q2

```

import org.w3c.dom.Node;
import org.w3c.dom.Attr;

```

```

class Personne {

    // ....

    public Personne(Node n) {
if (n==null) return;
if (! "personne".equals(n.getNodeName())) {
    System.err.println("Mauvaise balise.");
    return;
}
org.w3c.dom.NamedNodeMap m = n.getAttributes();
if (m.getNamedItem("id")==null ||
    m.getNamedItem("login")==null ||
    m.getNamedItem("nom")==null ||
    m.getNamedItem("prenom")==null ) {
    System.err.println("Il manque un attribut!");
    return;
}
id=Integer.parseInt(m.getNamedItem("id").getNodeValue());
nom=m.getNamedItem("nom").getNodeValue();
prenom=m.getNamedItem("prenom").getNodeValue();
login=m.getNamedItem("login").getNodeValue();
    }
}

```

Q3 Il faut tout d'abord écrire la méthode permettant de sauver une instance de la class Personne dans la base de donnée :

```

public void sauve() {
    java.sql.Statement stmt = null;
    java.sql.ResultSet rs = null;
    try {
        stmt = DataBaseConnection.conn.createStatement();
        rs = stmt.executeQuery("select login, nom, prenom from personnes where id=3");
        boolean estDejaDansLaBase=false;
        if (rs.next()) { estDejaDansLaBase=true; }
        stmt = DataBaseConnection.conn.createStatement();
        if (estDejaDansLaBase) {
            // simplement faire des requettes update
            // pour mettre a jour les donnees
            stmt.executeQuery("update personnes set nom='"+
                nom+"' where id="+id);
            stmt.executeQuery("update personnes set prenom='"+
                prenom+"' where id="+id);
            stmt.executeQuery("update personnes set login='"+
                login+"' where id="+id);
        } else {
            stmt.executeQuery("insert into personnes values(0,'"+
                login+"','"+nom+"','"+prenom+"')");
        }
    } catch (SQLException e) {
        System.out.println("SQLException: " + e.getMessage());
    } finally {
        if (rs != null)
            try { rs.close(); } catch (SQLException sqlEx) { }
        if (stmt != null)
            try { stmt.close(); } catch (SQLException sqlEx) { }
    }
}

```

La classe permettant de lire le fichier XML ainsi que le fichier ant se trouvent à l'url:

```
http://www.derepas.com/java/cours1_exercice_3.jar
```

L'archive est à décompresser à l'aide de la commande :

```
jar xvf cours1_exercice_3.jar
```

Pour tester que le code marche bien il faut:

- indiquer la bonne position du fichier `mysql-connector-java-*-bin.jar` au début du fichier `build.xml`.
- à la ligne 23 du fichier `DataBaseConnection.java` indiquer le bon login et le bon mot de passe.
- il suffit alors de taper `ant` sur la ligne de commande pour inclure le fichier `personnes.xml` dans la base de données.