Licence Pro DevOps

Cours 5 :

- Lambda expressions
- bases de données
- fichiers properties
- améliorations TP

Licence Pro DevOps

Java 8 :
Lambda expressions

Introduction

Mises à jour récentes importantes de Java :

- Java 4 en 2004 introduit la **programmation générique**
- Java 8 sort en Mars 2014 et introduit deux notions importantes :
  - **Lambda expressions**
  - **Streams**

- Java 9 est sorti en Septembre 2017, peu de changements
  - Modules
  - Mini shell

Rappel sur les classes anonymes

- Soit l'interface suivante :

```
public interface monInterface {

    public int m(int x, int y);

}
```

- Elle ne contient qu'une méthode : on dit que c'est une *interface fonctionnelle*

Rappel sur les classes anonymes

- Soit la classe suivante :

```
public class MaClasse() {

    public void foo(monInterface o) {
        System.out.println(o.m(12, 28));
    }

    public static void main(String[] args) {

        MaClasse unObjet = new MaClasse();

        unObjet.foo(****** que mettre ici ?******)
    }
}
```

- Choix 1 :
  - créer une classe implémentant l'interface
  - Créer une instance de cette nouvelle classe
  - Passer ce nouvel objet en paramètre de foo

Rappel sur les classes anonymes

- Soit la classe suivante :

```
public class MaClasse() {

    public void foo(monInterface o) {
        System.out.println(o.m(12, 28));
    }

    public static void main(String[] args) {

        MaClasse unObjet = new MaClasse();

        unObjet.foo(****** que mettre ici ?******)
    }
}
```

- Choix 2 : classe anonyme

Rappel sur les classes anonymes

Avec classe anonyme :

```
public class MaClasse() {

    public void foo(monInterface o) {
        System.out.println(o.m(12, 28));
    }

    public static void main(String[] args) {

        MaClasse unObjet = new MaClasse();

        unObjet.foo( new monInterface() {

            public int m(int x, int y) {
                return x*y;
            }

        });
    }
}
```

Lambda expressions

- Choix 3 : avec une lambda expression
- But : écrire de manière condensée une fonction

  syntaxe :

```
(paramètres) -> expression
```

  ou

```
(paramètres) -> {   plusieurs;
   instructions; }
```

- Les parenthèses sont facultatives s'il n'y a qu'un seul paramètre
- Si aucun paramètre :

```
() -> expression
```

Rappel sur les classes anonymes

Avec lambda expression :

```
public class MaClasse() {

    public void foo(monInterface o) {
        System.out.println(o.m(12, 28));
    }

    public static void main(String[] args) {

        MaClasse unObjet = new MaClasse();

        unObjet.foo( (x, y) -> x*y );
    }
}
```

Exemple d’utilisation

- Effectuer une opération sur les éléments d’une liste
  - exemple : ajouter 3 points à la note de tout le monde :

```
List<Etudiant> lpDevOps2 = new ArrayList<Etudiant>();

...

lpDevOps2.forEach( e -> e.setNote(e.getNote()+3) );
```

  - Et afficher la note de tout le monde :

```
LpDevOps2.forEach( e -> {
          System.out.println( e.getNom()+” “+e.getNote() );
      });
```

Exemple d’utilisation

- Implémentation d’un écouteur pour réagir à un événement tel que le clic d’un bouton
- Avec classe anonyme :

```
clearBtn.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
                textToSend.setText("");
            }

        });
```

Exemple d’utilisation

- Implémentation d’un écouteur pour réagir à un événement tel que le clic d’un bouton
- Avec lambda expression :

```
clearBtn.setOnAction( event → textToSend.setText("") );
            }

        });
```

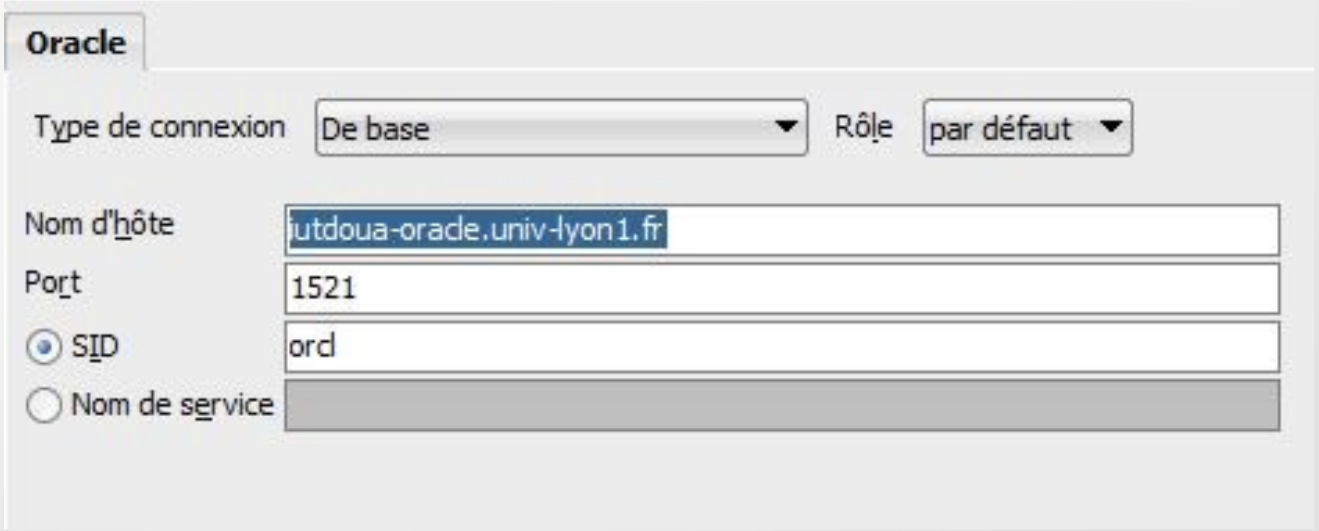
Bases de données

Bases de données en Java

- Ingrédients :
  - un serveur de base de données
    plusieurs SGBD (Systèmes de Gestion de BD) :
    - mySQL
    - PostegreSQL
    - Oracle
    - Access
    - …
  - sur ce serveur :
    - un utilisateur (et mot de passe)
    - au moins une table

Bases de données en Java

- À l'IUT :
  - serveur Oracle :
    iutdoua-ora.univ-lyon1.fr
    port : **1521**
  - pour l'administrer :
    - SQL developer installé sur les machines des salles TP :



Bases de données en Java

- Côté Java : pour chaque SGBD, un "Driver" différent pour pouvoir s'y connecter.
- Concrètement :
  - le driver est encapsulé dans un fichier .jar à ajouter à votre projet
  - il est à récupérer par exemple sur le site du SGBD
  - pour un serveur Oracle : **ojdbc8.jar**

Bases de données en Java

- Code minimal pour se connecter à la base de données (exemple sur le serveur de l'IUT)

va générer une exception si le driver n'existe pas (ex : problème avec le .jar)

```
Class.forName("oracle.jdbc.driver.OracleDriver");

System.out.println("Driver O.K.");



String url = "jdbc:oracle:thin:@iutdoua-ora.univ-lyon1.fr:1521:orcl";

String user = "le_login";
String passwd = "le_mot_de_passe";


Connection conn = DriverManager.getConnection(url, user, passwd);
System.out.println("Connexion à la base réussie");
```

Bases de données en Java

- Pour effectuer une requête :

```
Statement state = conn.createStatement();
```

- Puis, selon le type de requête, on invoque sur l'objet **`state`** :
  - **`boolean execute(String requete)`**
    - pour tout type de requête
  - **`ResultSet executeQuery(String requete)`**
    - plutôt pour les requêtes de type SELECT
  - **`int executeUpdate(String requete, int param)`**
    - pour les mises à jour : insert, delete, drop table...etc

Bases de données en Java

- Pour récupérer les données, exemple :

```
Statement state = conn.createStatement();
ResultSet result = state.executeQuery("SELECT * FROM MA_TABLE");

while (result.next()) {

    System.out.println(result.getInt("id"));
    System.out.println(result.getString("nom"));

}
```

- en général :
  `result.getUnType(columnLabel)`

  retourne directement un objet de type **`UnType`**, qui correspond à la colonne **`columnLabel`**

- On peut également bouger le curseur au début ou à la fin :
  `result.first()`
  `result.last()`

Stockage de données sensibles :

fichiers properties

Fichiers properties

- Ceci est mal :

```
Class.forName("oracle.jdbc.driver.OracleDriver");
●
System.out.println("Driver O.K.");


String url = "jdbc:oracle:thin:@iutdoua-oracle.univ-lyon1.fr:1521:orcl";

String user = "le_login";
String passwd = "le_mot_de_passe";


Connection conn = DriverManager.getConnection(url, user, passwd);
System.out.println("Connexion à la base réussie");
```

- On ne veut pas stocker ces infos dans le code source car :
  - ne peut être déployé proprement
  - problèmes de sécurité

Fichiers properties

- Comment faîtes vous généralement… en PHP ?

Fichiers properties

- Comment faîtes vous généralement… en PHP ?

  → on stocke ces infos dans un fichier séparé

- même idée ici : les fichiers properties

Fichiers properties

- Fichier properties :
  - extension .properties
  - peut être placé n'importe où
  - on peut gérer les droits de ce fichier indépendamment
- Exemple :

```
url      = jdbc:oracle:thin:@iutdoua-oracle.univ-lyon1.fr:1521:orcl
user = rwatrigant
passwd   = mon_password
```

- permet de stocker des couples (clé, valeur)
- on peut rajouter des commentaires avec #

Fichiers properties

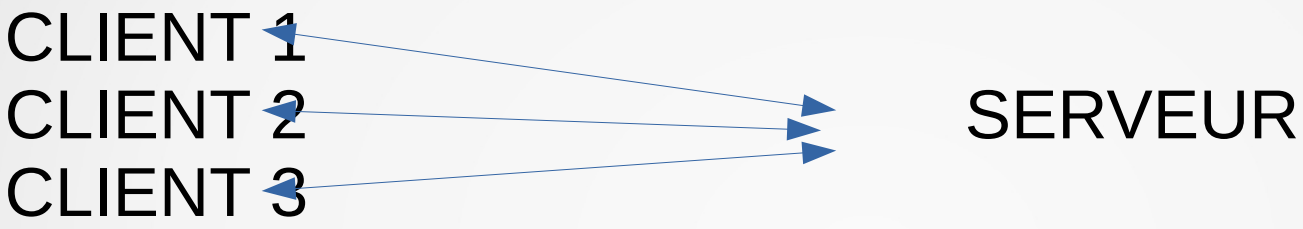
- Puis, dans votre programme Java :

```
Properties prop = new Properties();
FileInputStream input = new FileInputStream("config.properties");
prop.load(input);

String url = prop.getProperty("url");
String user = prop.getProperty("user");
String passwd = prop.getProperty("passwd");
```

Dans votre TP/projet

- Application de bureau

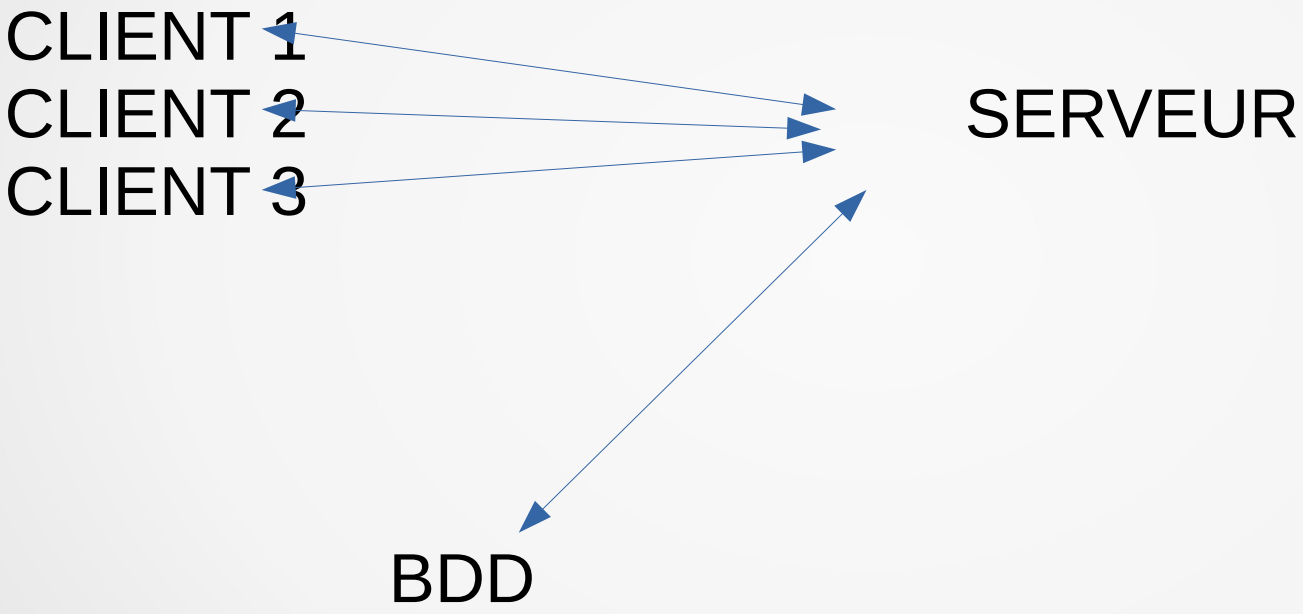


? ?

BDD

Dans votre TP/projet

- Application de bureau



C’est le serveur seulement qui se connecte à la base de données

(idem que pour un serveur web)

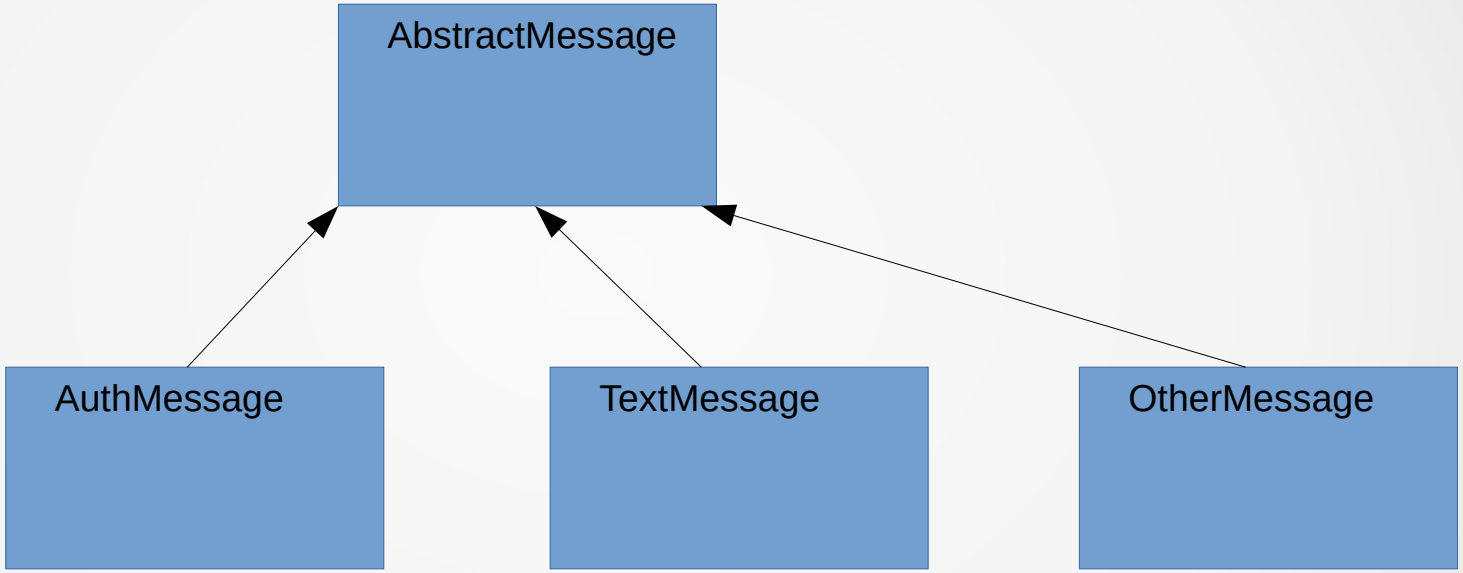
Organisation

- Généralement, une classe est responsable de :
  - Connexion à la base de données
  - Requêtes
- Bonne pratique : utiliser un pattern singleton

Extension du TP

Améliorations du TP

- Vous aurez certainement à gérer plusieurs types de messages



Améliorations du TP

- Lors de la réception d'un message, pour différencier :

```
Object o = in.readObject();

if (o instanceof TextMessage) {
   TextMessage mess = (TextMessage) o;
   //here we handle text messages
   continue;
}
if (o instanceof AuthMessage) {
   AuthMessage mess = (AuthMessage) o;
   //here we handle authentification
messages
   continue;
}
```