RAPPORT TP1 : FLUX RSS HERBERT SIMONIN

But :

Créer une application permettant de gérer des flux rss. Permettant à utilisateur de se connecter (s’inscrire) et de s’abonner à un (des) flux.

Permettant également à un (des) administrateur(s) de gérer les flux et les utilisateurs.

Mode d’emploi :

La class Application est la class principale contenant le main.

Au lancé de l’application, l’utilisateur sera demander de choisir le type connexion (Admin/User) ou s’inscrire si non-inscrit.

Les données sont stocker dans une base de donnée mySQL ( flux.sql , présent dans le dossier du projet)

Une fois connecté :

\_Admin : -Peut :

-ajouter un flux,

-supprimer un flux,

-supprimer un utilisateur

-Voir les informations sur un Utilisateur ou u flux

-User : Peut :

-S’abonner à un flux/catégorie

-se désabonner d’un flux/catégorie

-consulter les informations lui concernant

En cas de soucis avec l’accès à la base de donné (ex : problème au niveau du fuseau horaire :

Modifier la variable  « time zone » du server(si Apache-MySql) à « +10 :00 »

Les class :

Class User :

Class définissant un utilisateur :

Paramètre : **protected** String nom;

**protected** String prenom;

**protected** String mail;

**protected** String identifiant;

**protected** String mdp;

Methodes:

**public** String getNom() {

**return** nom;

}

**public** **void** setNom(String nom) {

**this**.nom = nom;

}

**public** String getPrenom() {

**return** prenom;

}

**public** **void** setPrenom(String prenom) {

**this**.prenom = prenom;

}

**public** String getMail() {

**return** mail;

}

**public** **void** setMail(String mail) {

**this**.mail = mail;

}

**public** String getIdentifiant() {

**return** identifiant;

}

**public** **void** setIdentifiant(String identifiant) {

**this**.identifiant = identifiant;

}

**public** String getMdp() {

**return** mdp;

}

**public** **void** setMdp(String mdp) {

**this**.mdp = mdp;

}

**public** String toString () {

**return** **new** String ("nom : " + getNom() + "\nprenom : " + getPrenom() + "\nmail : " + getMail() + "\nidentifiant : " + getIdentifiant() + "\nmot de passe : " + getMdp());

}

Inscription

//Ajout un nouvel utilisateur dans la table users

**public** **void** inscription () **throws** SQLException {…}

connexion

//Vérifier si le mot de passe correspodant à l'identifiant

**public** **boolean** connexion () **throws** SQLException {…}

Abonement à un flux

**public** **void** abonnement (User user, Flux flux, **int** nbJours) **throws** SQLException {…}

Abonnement à une catégorie

**public** **void** abonCat (String nomCat) **throws** SQLException {…}

désabonnement d’un utilisateur

**public** **void** desabonner(Flux flux) **throws** SQLException {…}

Ajouter une catégorie

**public** **void** ajoutCat (String nomCat) **throws** SQLException {…}

compte nombre d’abonné

**public** **int** nbAbonCat (String nomCat) **throws** SQLException {..}

ajouter +”nb” abonné à une catéforie

**public** **void** ajoutAbonCat (String nomCat, **int** nbAbonCat) **throws** SQLException {…}

prolonger un abonnement

**public** Date ajouterJour(Date date, **int** nbJours) {…}

Verifier si nombre maximum d’abonnement d’un user est atteint

**public** **boolean** verifContrainte() **throws** SQLException {…}

Class Admin: **public** **class** Admin **extends** User

Class définissant les fonctions pour un administrateur

Connexion Admin

**public** **boolean** connexion () **throws** SQLException {…}

Supprimier un Utilisateur

**public** **void** suppUser(User user) **throws** SQLException {…}

Supprimer un abonnement d’un utilisateur

**public** **void** suppAbonnements (User user) **throws** SQLException {…}

Supprimer une contrainte d’un utilisateur

**public** **void** suppContraintes (User user) **throws** SQLException {…}

Ajouter une contrainte(limite nb d’abonnement)

**public** **void** contraintes(User user, **int** nb) **throws** SQLException {…}

Supprimer un flux

**public** **void** suppFlux(Flux flux) **throws** SQLException {…}

Supprimer un abonnement à un flux d’un utilisateur

**public** **void** suppAbonnementsF (Flux flux) **throws** SQLException {…}

supprimer un abonnement à une catégorie flux d’un utilisateur

**public** **void** suppAbonnementsC (User user) **throws** SQLException {…}

Class Flux: Class définissant les flux

Paramètre : **private** URL url;

**private** String nom;

**private** String langue;

**private** String localisation;

**private** Date dateAjout;

**private** **int** nbAbos;

Methodes :

**public** URL getUrl() {

**return** url;

}

**public** **void** setUrl(URL url) {

**this**.url = url;

}

**public** String getNom() {

**return** nom;

}

**public** **void** setNom(String nom) {

**this**.nom = nom;

}

**public** String getLangue() {

**return** langue;

}

**public** **void** setLangue(String langue) {

**this**.langue = langue;

}

**public** String getLocalisation() {

**return** localisation;

}

**public** **void** setLocalisation(String localisation) {

**this**.localisation = localisation;

}

**public** Date getDataAjout() {

**return** dateAjout;

}

**public** **void** setDateAjout(Date dateAjout) {

**this**.dateAjout = dateAjout;

}

**public** **int** getNbAbos() {

**return** nbAbos;

}

**public** **void** setNbAbos(**int** nbAbos) {

**this**.nbAbos = nbAbos;

}

**public** String toString () {

**return** **new** String ("url : " + getUrl() + "\nnom : " + getNom() + "\nlangue : " + getLangue() + "\nlocalisation : " + getLocalisation() + "\ndateAjout : " + dateAjout + "\nnbAbos : " + getNbAbos());

}

Ajouter un flux

**public** **void** ajoutFlux (Flux flux) **throws** SQLException {…}

Récupérer le nombre d’abonné à un flux

**public** **int** nbAbos () **throws** SQLException {…}

Ajouter/Modifier le nombre d’abonné

**public** **void** ajoutAbos (**int** abon) **throws** SQLException {…}

Interface: Interface :

**public** **static** **int** ***jour*** = 28; Nombre de jours d’abonnement par défaut

Méthodes :

Menu principale : Demande à l’utilisateur de s’inscrire ou se connecter (user ou admin). Options : A-> Administrateur ; C-> Se connecter ; I-> S’inscrire

**public** **default** **char** bienvenu () {…}

Afficher le menu utilisateur après connexion :

Options : A-> s'abonner à un nouveau flux ; B-> s'abonner à une catégorie ; C-> consulter les flux disponibles ; D-> se désabonner d'un flux ; F-> consulter vos catégories ; I-> consulter vos informations(flux) Q-> Quitter

**public** **default** **char** choixUser() {…}

Afficher le menu administrateur :

A->Ajouter un flux ; S-> supprimer un compte utilisateur ou un flux ; I-> afficher les infos\n Q: Quitter

**public** **default** **char** choixAdmin() {…}

Afficher le sous-menu Supprimer :

Options : U-> Utilisateur ; F-> Flux

**public** **default** **char** choixSupp() {…}

Affichier le sous menu Informations

Options : U-> Utilisateur ; F-> Flux

**public** **default** **char** choixInfos() {…}

Inscription d’un nouvel utilisateur (Option I de la méthode bienvenue).

Demande à l’utilisateur d’entrer les informations nécessaire (nom, prénom, mail, identifiant, mot de passe) et vérifie s’il identifiant et mail existe déjà dans la base de données

**public** **default** **void** inscription () **throws** SQLException {…}

Connexion d’un utilisateur : vérifie si l’indentification est correcte (existe dans la base de données).

Affiche un message de succès, erreur sinon

**public** **default** User connexion () **throws** SQLException {…}

Idem mais pour administrateur

**public** **default** Admin connexionA () **throws** SQLException {…}

Administrateur : Affiche la liste des flux dans la base de données

**public** **default** **void** afficherFluxA () **throws** SQLException {…}

Administrateur : Affiche la liste des utilisateurs dans la base de données

**public** **default** **void** afficherUsersA () **throws** SQLException {

Permet à l’utilisateur de s’abonner à un flux ;

**public** **default** Flux abonnement () {…}

Permet à l’utilisateur de s’abonner à une catégorie de flux ;

**public** **default** **void** abonnementC (User user) **throws** SQLException {

Permet voir les informations sur utilisateur

**public** **default** **void** afficherInfos (User user) **throws** SQLException {

Permet à l’utilisateur de se désabonner/Admin de désabonner un utilisateur

**public** **default** Flux desabonner (User user) **throws** SQLException {

Admin : Permet de supprimer un utilisateur

**public** **default** **void** supprimerUsersA () **throws** SQLException {

Admin : Permet d’ajouter un flux

**public** **default** **void** AjouterfluxA() **throws** SQLException{

Admin : Permet de supprimer un flux

**public** **default** **void** supprimerFluxA () **throws** SQLException {…}

Class abonnement : Les fonctions de la class ont été déplacés dans la class User

Class Application : Class principale contenant le programme(le main) qui définit dans le mode d’emploi un peu plus haut.

Elle implémente l’interface Interface. Utilise ainsi les méthodes pour lancer l’application.

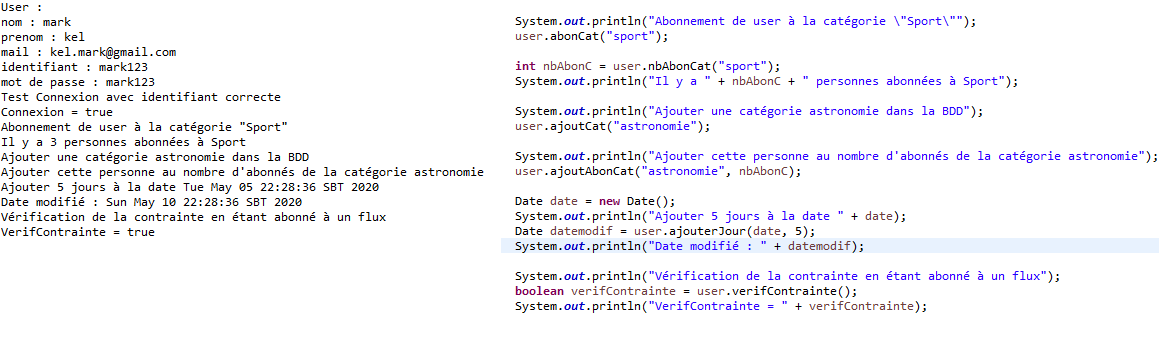
**public** **class** Application **implements** Interface {…}

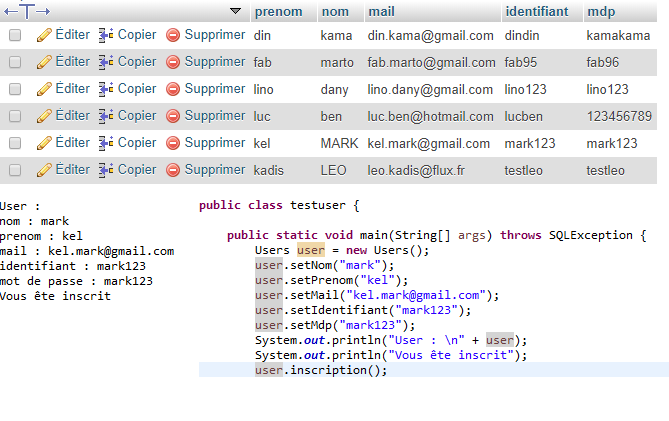
Qui demande le type de connexion ou inscription si non inscrit et et affiche le menu selon le type d’utilisateur et les actions choisit par l’utilisateur(admin/user).

Class de test :

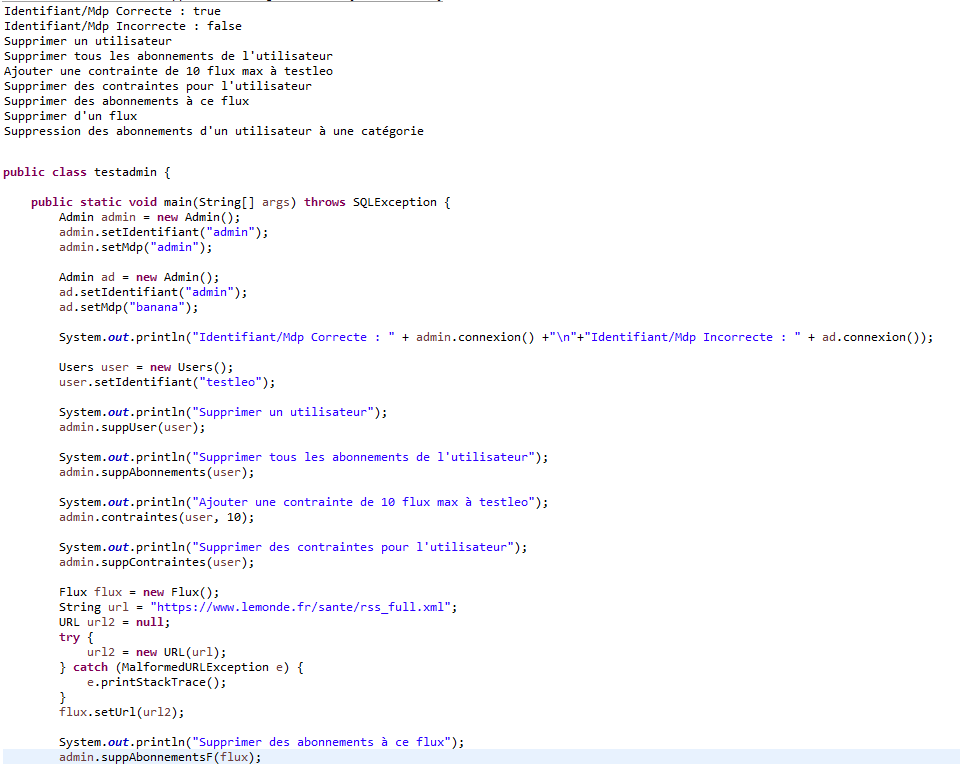
testUser : Test les fonctions de la class



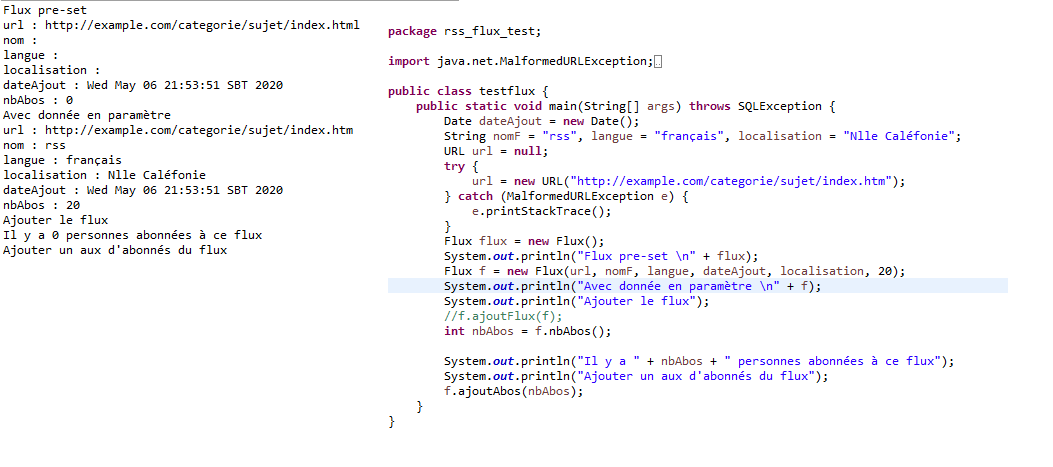




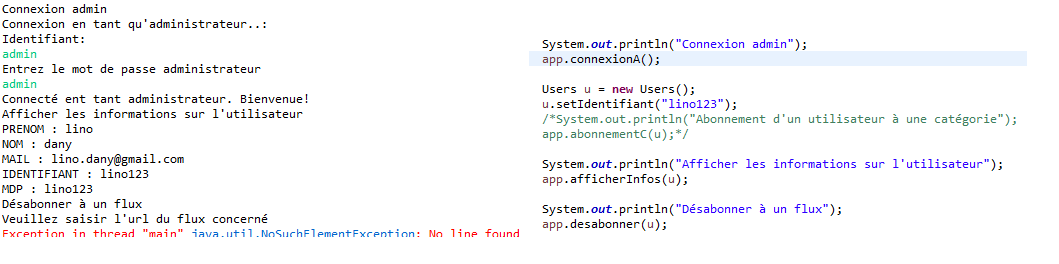
testAdmin :



testFlux :



testApp :



Test application n’est pas parfaitement fonctionnel, certain methodes (les méthodes des sous menu) posent des soucis au niveau de la saisi clavier.

Les autres méthodes de l’interfaces fonctionnent.

Conclusion :

Application est fonctionnelle avec un petit souci au niveau de sous menu admin/user.

Peut-être améliorer en ajoutant par exemple d’autres types de contraintes.