CONSERVATOIRE
NATIONAL
DES ARTS
ET METIERS

CENTRE REGIONAL
MIDI - PYRENEES

NFA 031 – Travaux pratiques

IPST-
CNAM
Licence

PROJET

Années 2015-2016
TOULOUSE
J. LAFORGUE

1. LE BESOIN

On se propose de réaliser un programme qui permet de gérer son compte bancaire dont son budget.

Un compte bancaire est une classe **Compte** qui gère des opérations (du relevé bancaire ou budgétées).

Une opération est un objet de la classe **Operation** (les attributs minimal sont dans le code fourni)**.**

A terme le programme devra être capable de :

- [FUNC 1] lire un fichier texte (fourni) contenant les opérations et les mémoriser dans le programme
- [FUNC 2] afficher toutes les opérations depuis l'IHM
- [FUNC 3] afficher les opérations non budgétées d'un mois donné (MM/AAAA) depuis l'IHM
- [FUNC 4] naviguer dans les opérations non budgétées (mois suivant du mois courant, mois précédent du mois courant, mois courant initial) depuis l'IHM
- [FUNC 5] afficher toutes les opérations non budgétées de l'année du mois courant et les opérations budgétées de l'année du mois courant à partir du mois suivant du mois courant. Et afficher le solde final.
- [FUNC 6] ajouter une nouvelle opération
- [FUNC 7] supprimer une opération

Bien sur, Il n'est pas envisagé de tout développer en une seule fois. Nous allons réaliser ce programme en 3 étapes.

2. ETAPE 1

Pour démarrer votre programme vous partez du code fourni (projet.zip) dans lequel vous trouvez :

une classe **Projet** avec une méthode main qui crée le **Compte** et l' **IHMCompte**.
L'IHMCompte prend en entrée le Compte.

Dans cette étape, on se propose de réaliser les fonctionnalités [FUNC 1] [FUNC 2] et [FUNC 3]

Le constructeur de la classe **Compte** doit lire le fichier texte dont chaque ligne est une opération puis ajoute les opérations dans une collection.
(Vous utilisez la méthode String[] Terminal.lireFichierTexte("data/Compte.txt"); pour lire le fichier)
(La méthode String[] split(String ligne,7) permet de décoder une ligne du fichier contenant 7 champs).

Chaque ligne du fichier est de la forme :

date;montant;type;budgeté;num_cb;num_cheque;libellé

avec

date	au format JJ/MM/AAAA ou JJ ou JJ/MM (*)
montant	une valeur double (ex : 123.45)
type :	"CB","CHEQ", "VIRT", "PRVT", "DEPT" ou ""
budgeté :	"B" ou ""
num_cb	si type="CB" alors numéro du chèque (string) sinon ""
num_cheque	si type= CHEQUE" alors numéro du chèque (int) sinon ""
libellé	texte libre (tiers, motif, bénéficiaire) (peut être "")

(*) JJ ou JJ/MM ne sont utilisés que pour une opération budgétée.
Si JJ alors cela signifie une opération mensuelle qui est réalisée le JJ de chaque mois.
Si JJ/MM alors cela signifie opération annuelle qui est réalisé le JJ/MM de chaque année.

Exemple de fichier :

```
01/10/2015;-10.50;CB;;F123A67;;boulangerie
03/10/2015;-110.50;CB;;F123A67;;boulangerie
03/11/2015;2000.50;VIRT;;;;salaire
03/11/2015;-500.00;PRVT;;;6789;edf
23/11/2015;1110.50;CHEQUE;;;6789;voiture
03/12/2015;2200.50;VIRT;;;;salaire
03/10/2015;-75.50;CB;;F123A67;;restaurant
13/11/2015;-45.00;CB;;F123A67;;
13/10/2015;-35.00;CB;;F123A67;;
05/10/2015;50.00;;;;;
05;-150.00;;B;;;eau (le 5 de chaque mois)
05;-300.00;;B;;;impot (le 5 de chaque mois)
05/09;-300.00;;B;;;assurances (chaque annee)
05/11/2015;-200.00;;B;;;cadeau (1 jour donné)
02/10/2015;-40.45;CB;;F123A67;;essence
```

Pour chaque ligne lue, une opération est créée (new Operation(...)). On passe en paramètre du constructeur de Operation les champs lus. Puis cette opération est ajoutée dans une collection (ArrayList) du Compte.

La classe **IHMCompte** crée un formulaire dans lequel :

- un bouton permet d'afficher tous les rendez-vous dans la zone de résultat (chaque opération est affichée sous la forme d'une chaîne de caractère)

- un texte de saisi et un bouton permettant d'afficher les opérations d'un mois donné (MM/AAAA). Vous devez tester si le mois saisi est mal écrit.

Pour réaliser ces deux actions, codez dans la classe Compte les deux méthodes suivantes :

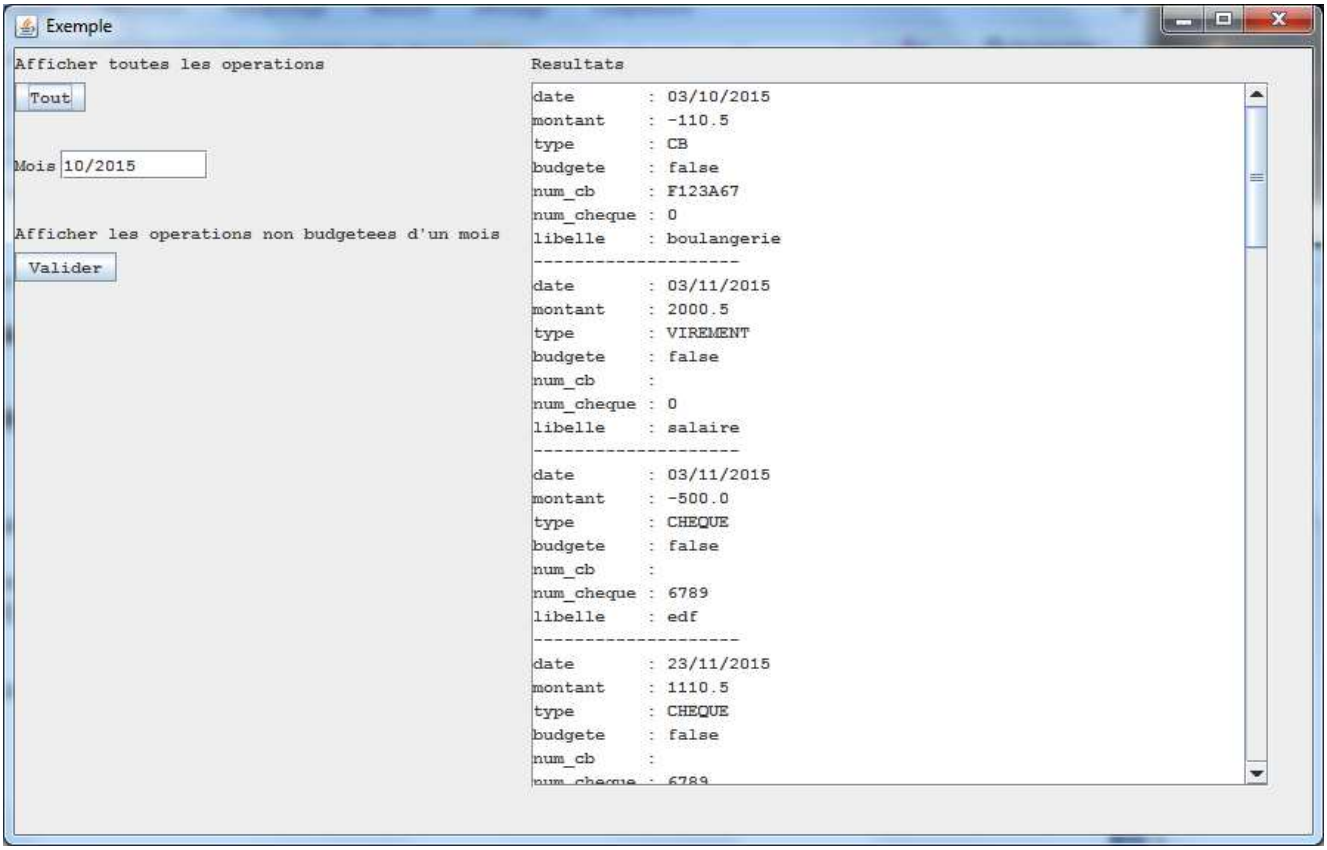
public String listerToutesOperations()
Cette méthode retourne sous la forme d'une chaine de caractère, toutes les opérations (une boucle qui appelle la méthode toString de chaque opération)

public String listerMoisOperations(String mois)
Cette méthode retourne sous la forme d'une chaine de caractère, toutes les opérations non budgétées du mois passée en paramètre (MM/AAAA).

Ces deux méthodes sont appelées dans la méthode **submit** de la classe IHMCompte.

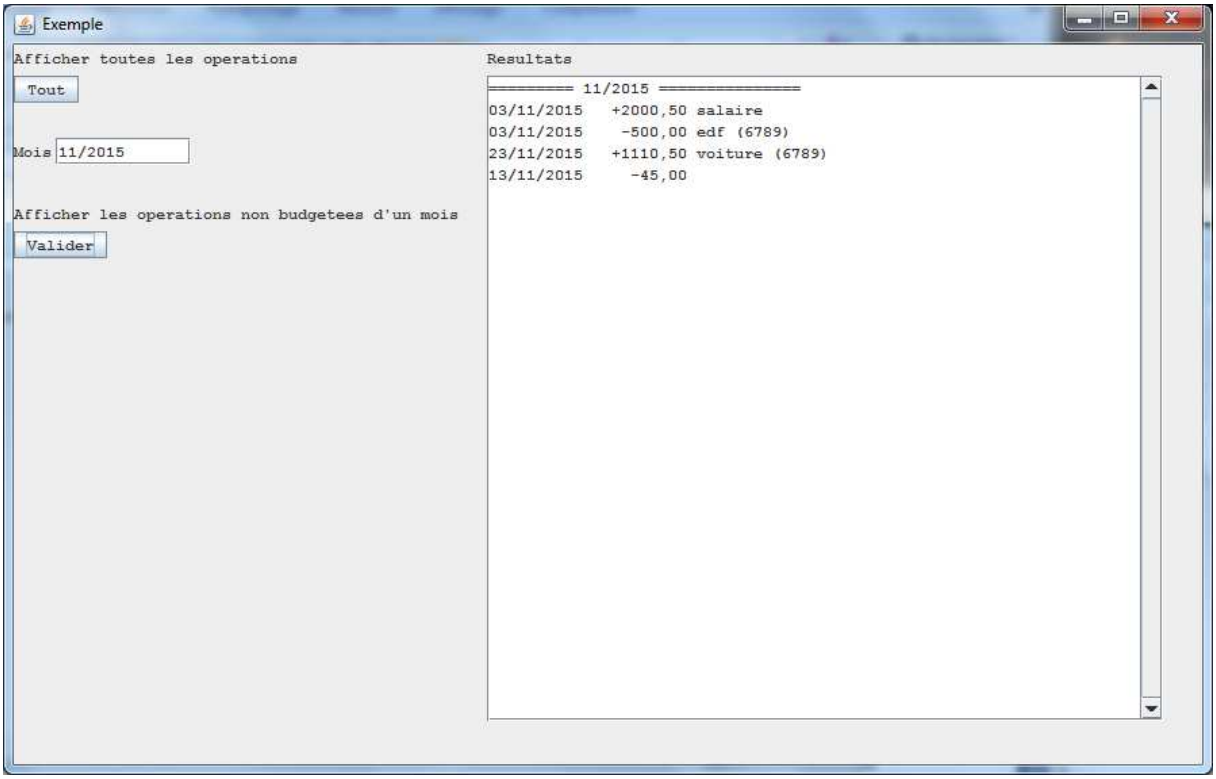
Nous supposons que le fichier texte ne contient pas d'erreur de syntaxe: pas de "point-virgule" en moins, toutes les informations sont correctement écrites.

La méthode listerToutesOperations affiche le résultat ainsi :



On affiche tous les attributs de la classe Operation. 1 ligne par attribut.

La méthode listerMoisOperations affiche le résultat ainsi :



Chaque opération est affichée sur 1 seule ligne sur le format :

date montant libellé (cheque)

3. ETAPE 2

Dans cette étape on se propose de réaliser les fonctionnalités [FUNC 4] [FUNC 5].

1/ [FUNC 4]
L'IHM contient 3 boutons : Suivant, Courant, Precedent :

Le bouton Courant affiche toutes les opérations non budgétées du mois courant initial (celui de la 1ère ligne du fichier).
Le bouton Suivant incrémente le mois courant et affiche les opérations non budgétées de ce nouveau mois courant.
Le bouton Precedent décrémente le mois courant et affiche les opérations non budgétées de ce nouveau mois courant.

On fait évoluer le format du fichier :
La première ligne du fichier texte est au format MM/AAAA et correspond à la valeur du mois courant initial.
Viennent ensuite les autres lignes contenant les opérations (comme dans l'étape 1).

2/ [FUNC 5]
Pour afficher le budget de l'année du mois courant, on ajoute un bouton qui appelle un traitement de Compte qui réalise le calcul et retourne le résultat sous la forme d'une chaine de caractère.

Le budget de l'année du mois courant MM/AAAA est la liste de toutes les opérations non budgétées de l'année AAAA et de toutes les opérations budgétées qui sont supérieures ou égales à 01/MM+1/AAAA.
Toutes les opérations sont dans l'ordre des dates.

Réfléchissez à l'algorithme de ce traitement. Ne pas utiliser de méthode de tri (pas vu en cours).

Pour coder ce traitement, vous pouvez vous aider des méthodes statiques de la classe **DateString** fournie.

4. ETAPE 3

Dans cette étape on se propose de réaliser les fonctionnalités [FUNC 6] et [FUNC 7].

1/ [FUNC 6]
On se propose d'ajouter une opération dans le compte avec une saisie de tous les champs d'une opération dans une IHM.

Pour cela vous devez créer une nouvelle classe **IHMAjouterOperation** dont le constructeur prend en entrée le compte et l'ihm du compte, et qui crée une nouvelle IHM permettant de saisir tous les champs d'une opération.
Un bouton Valider permet de valider la saisie : appel d'un traitement de Compte qui fait la création d'une opération et ajoute cette opération dans le compte.
Un bouton Fermer permet de fermer l'IHM.
Par défaut le champ de saisi de la date de l'opération est le 1er du mois courant du compte.

Ensuite, vous devez ajouter dans l'IHM du compte un bouton **Créer une operation** qui affiche l'ihm **IHMAjouterOperation.**

Une fois l'ajout réalisé, on affiche dans la zone résultat l'affichage de toutes les opérations.

Remarque : aucune vérification des valeurs saisies n'est demandée. Il faut saisir des valeurs correctes.

Vous pouvez utiliser la méthode addListeChoix de Formulaire qui affiche une liste de choix possible, pour les saisies des champs type et budgeté de l'opération.

2/ [FUNC 7]
On se propose de supprimer une opération dans le compte en fonction d'un numéro saisi. Pour cela, il faut que toutes les opérations soient numérotées de manière unique (vous définissez un compteur dans la classe Compte).

Dans l'IHM du compte on ajoute le texte de saisi du numéro et le bouton **Supprimer**.

S'il n'existe pas d'opération du numéro saisi, vous affichez un texte d'erreur.
Sinon on affiche dans la zone résultat l'affichage de toutes les opérations.

Remarque : aucune vérification du type de la valeur saisie du numéro n'est demandée.

5. QUELQUES PRECISIONS

Les règles de codage que vous devez suivre sont :

- sauf pour les constantes, les attributs de toutes les classes doivent être privés
- toutes les classes commencent par une majuscule
- tous les attributs, paramètres et variables commencent par une minuscule
- les noms de tous les répertoires de package sont en minuscule
- les packages créés commencent tous par "fr.cnam"
- une indentation stricte est respectée dans tout le code
- un minimum de commentaire est nécessaire afin de comprendre ce que fait le code. Surtout quand ce que fait le code n'est pas décrit précisément dans le sujet.

Conseils :

- Ecrivez peu de code à chaque fois afin d'avoir toujours un code qui se compile correctement
- Après chaque étape ou sous-étape, le programme s'exécutant correctement, faites une sauvegarde des sources. Cela permet de pouvoir revenir en arrière si besoin et permet aussi de pouvoir donner à l'enseignant une version du programme qui s'exécute correctement et qui remplit correctement une partie des fonctionnalités demandées dans le sujet.
- Dans un premier temps ne faites pas plus que le sujet ne le demande. Une fois que le programme respecte le sujet, vous faites une sauvegarde et vous pouvez apporter des améliorations qui sont toujours appréciées par le correcteur, à condition qu'elles soient bien commentées.

Dans le fichier Projet.java, vous renseigner les noms et prénoms du groupe.

6. ARCHITECTURE DES REPERTOIRES

L'architecture suivante est une proposition de création des répertoires et des fichiers pour réaliser votre projet. Cette architecture doit être respectée.

projetNFA031
- **data**
  - Compte.txt
- **src**
  - **fr**
    - **cnam**
      - **projet**
        - Projet.java (*main*)
        - Compte.java
        - IHMCompte.java
        - Operation.java
        - IHMAjouteroperation.java
      - **ihm**
        - Formulaire.java
        - FormulaireException.java
        - FormulaireInt.java
        - ....
      - **util**
        - DateString.java
        - Terminal.java
        - TerminalException.java
        - ...
- **bin**
  - <les .class>

Les classes en rouge sont des classes qui vous sont données.

Le répertoire **data** contient le fichier contenant un compte initial (cela permet de ne pas saisir les opérations à chaque exécution). Le nombre d'opération ne doit pas être trop grand mais doit être représentatif des traitements demandés dans le projet.

7. PLANNING

Vous avez 3 séances de TP pour réaliser ce projet

Le projet sera ramassé 1/4 heure avant la fin de la dernière séance de TP (le temps de les ramasser tous).
L'enseignant vous demandera de déposer vos projets dans le répertoire "commun" spécifique (**lecteur réseau Q:**) qu'il récupèrera sur une clef USB.

Il est donc **impératif** que, au moins un membre du groupe de TP, soit présent lors de la dernière séance afin de donner son projet.