Consignes. Avant toute chose, prenez conscience que les corrections sont effectuées à l'aide d'un automate, par conséquent toute violation des règles conduira inévitablement à la délivrance d'une note nulle.

- Comment livrer mon code ?
  Si mon nom de famille est **yunès** et que je souhaite rendre le travail pour l'exercice 12 du tp 3 alors je dois construire une archive au format ZIP de nom `yunes-tp3-ex12.zip`, le nom ayant été converti en minuscules, sans accent et sans espace (*ie* : **de Gaulle** doit être traduit en `degaulle`). Si vous avez un homonyme rajoutez des initiales, par exemple `yunesjb-tp1-ex1`. Respectez votre nommage pour tout le semestre.
- Que mettre dans l'archive ?
  Le code source (et uniquement le code source) correspondant à l'exercice.

De plus, suivez bien les instructions complémentaires des livrables.

Exercice 1

1. Écrire un programme permettant de compter les mots d'un fichier contenant du texte et dont le nom est passé en argument, *i.e. :* `java CompteMots` *fichier*. Pour cela, il faut :
   - découper les lignes en mots (un mot c'est n'importe quoi sauf des espaces, tabulations et retours à la ligne). (Aide : `String.split`)
   - utiliser les `Stream` de Java.

Pour tester vous pouvez utiliser les deux fichiers textes proposés sur la page du cours.
Livrable : l'archive contenant le programme `CompteMots.java` (pas de package!).

Exercice 2

1. Écrire un programme permettant d'obtenir la liste des mots d'un fichier contenant du texte et dont le nom est passé en argument, *i.e. :* `java ListeMots` *fichier*. Pour cela, il faut :
   - découper les lignes en mots (un mot c'est n'importe quoi sauf des espaces, tabulations et retour à la ligne). (Aide : `String.split`)
   - utiliser les `Stream` de Java,
   - que chaque mot ne soit présent qu'une seule fois dans la liste,
   - que les majuscules/minuscules ne fassent pas de différence (*ie. :* `Bonjour` et `bonjour` soient le même mot.

Livrable : l'archive contenant le programme `ListeMots.java` (pas de package!).

Exercice 3

À l'aide les classes fournies (`AccessEntry`, `UserAgent` et du fichier `access.log` (qui sont à télécharger depuis la page du cours):

1. Écrire un programme permettant de parser un fichier de traces d'accès à un site web, *i.e. :* `java Parse` *fichier* et générer un `Stream<AccessEntry>` des entrées parsées puis d'afficher en sortie celles qui ont été correctement parsées.
   - utiliser les `Stream` de Java en injectant chaque ligne lue dans un le constructeur de `AccessEntry`.

   L'affichage se fera via la classe `AccessEntry`:

   ```
   IP=89.144.201.133, Date=2016-02-27T22:30:13+01:00, Request=GET, URL=/, Proto=HTTP/1.1, Code=200, Le
   ```

   Livrable : l'archive contenant le programme `Parse.java` (pas de package!).

2. Modifier le programme de sorte à obtenir, la liste ordonnée de tous les codes réponse HTTP. Le nouveau programme s'exécutera *via* `java Code` *fichier*. Livrable : l'archive contenant le programme `Code.java` (pas de package!).

3. Modifier le programme de sorte à obtenir, pour chaque code réponse HTTP, le nombre d'entrées parsées correspondant. Le nouveau programme s'exécutera *via* `java Code` *fichier*. La sortie doit être pour chaque code, le code, un espace et le nombre d'entrée pour ce code donné, comme :

```
200 4507
403 45
404 12045
```

   Livrable : l'archive contenant le programme `Code.java` (pas de package!).

4. Modifier le programme de sorte à obtenir, pour chaque mois et année, le nombre d'entrées parsées correspondant. Le nouveau programme s'exécutera *via* `java Stat` *fichier*. La sortie doit être pour chaque ligne : le mois en anglais sur 3 caractères, l'année et le nombre d'entrées correspondantes comme :

```
Dec 2015 14100
Jan 2016 28177
Feb 2016 117140
```

   Livrable : l'archive contenant le programme `Stat.java` (pas de package!).