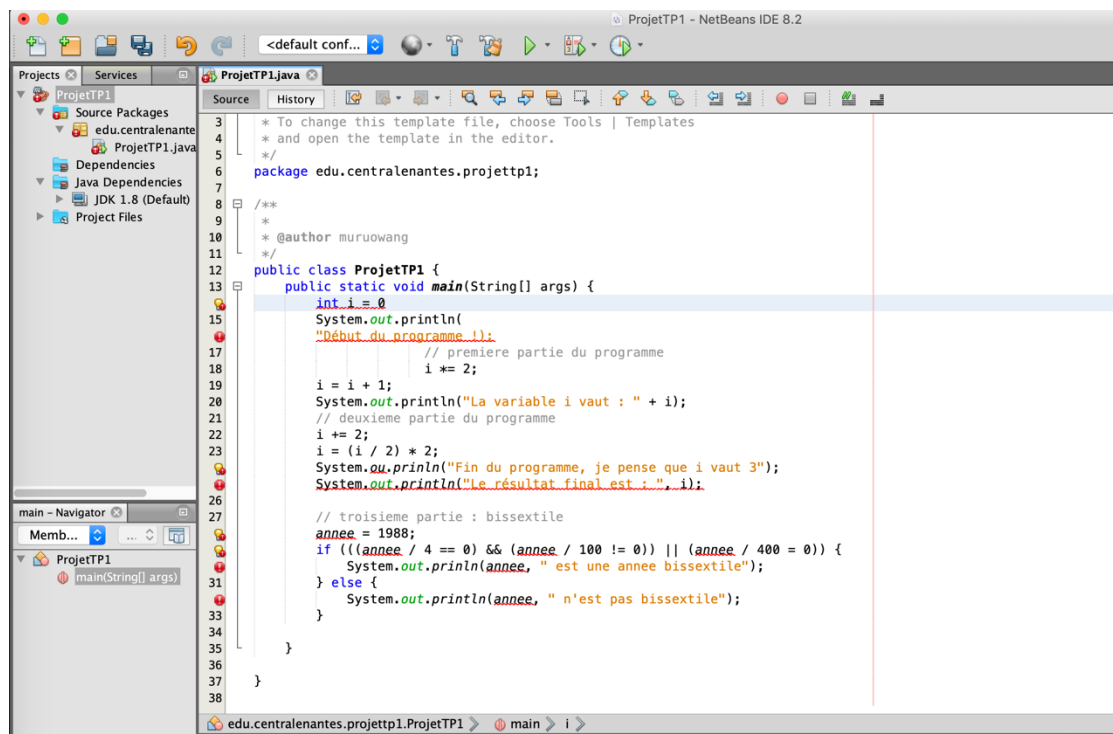


## Rapport TP1 OBJET

L'objectif de ce TP est de prendre en main l'IDE NetBeans et de s'initier à la programmation orientée objet en corrigeant des codes et en créant une classe.

### Exercice 1 : Correction d'erreurs

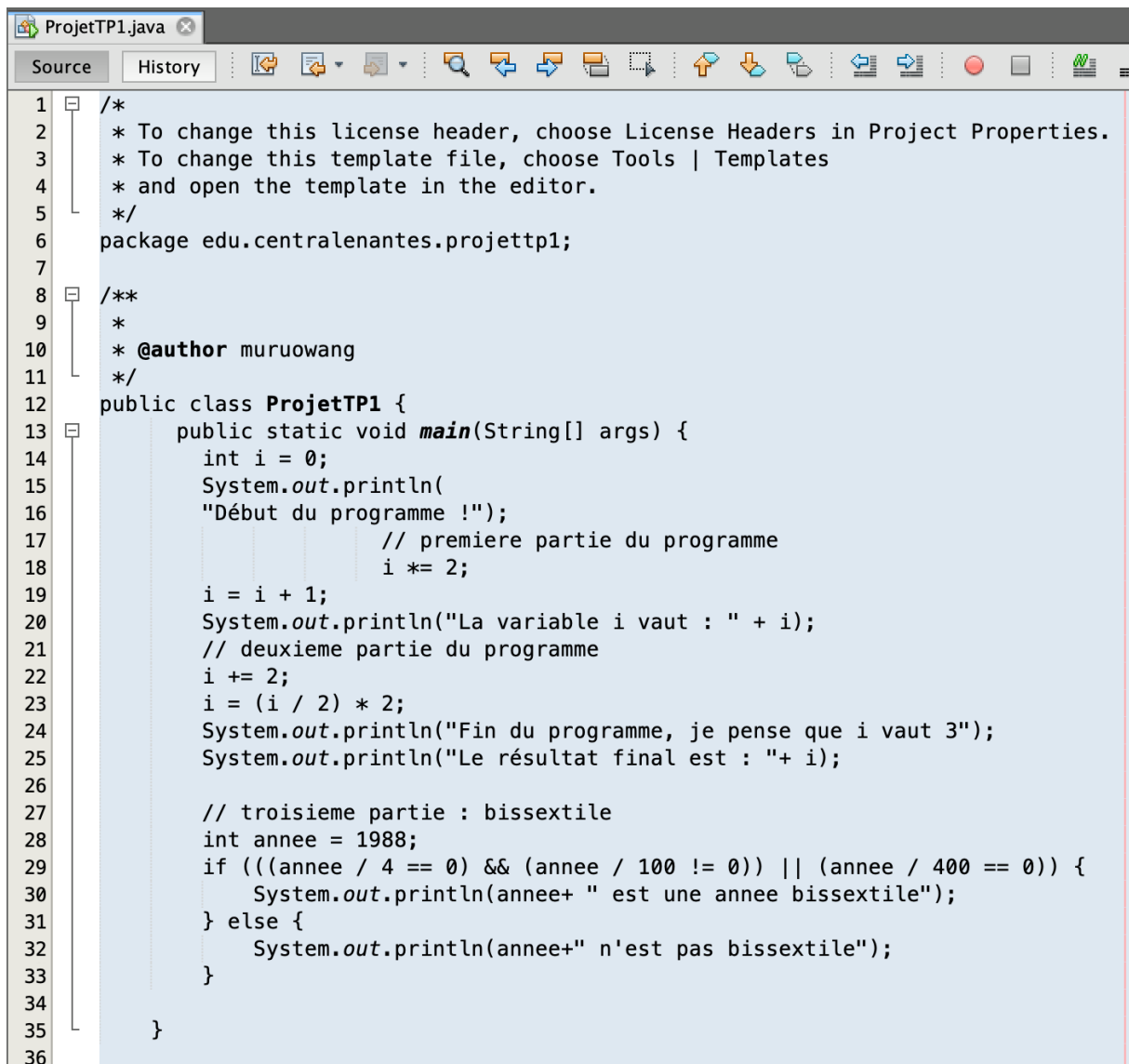
NetBeans nous indique que le code du fichier ProjetTP1-Ex1.java comporte plusieurs erreurs (en rouge) :



Nous avons effectué les corrections suivantes :

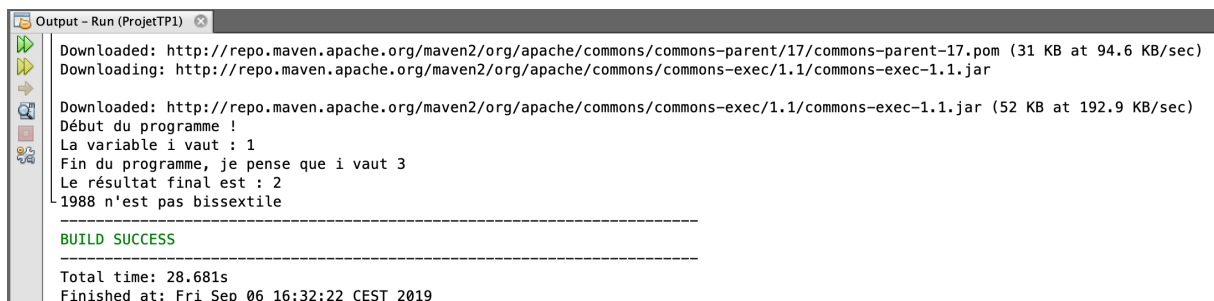
- Ligne 14 : ajout d'un point-virgule en fin de ligne pour terminer l'instruction
- Ligne 16 : ajout de guillemets après le point d'exclamation pour finir la chaine de caractères.
- Ligne 24 : correction de « ou » en « out » et de « println » en « println ».
- Ligne 25 : on remplace la virgule par un + pour faire de la concaténation de chaine de caractères. En effet, la fonction System.out.println ne prend qu'un seul argument.
- Ligne 28 : ajout du type de la variable année (int) pour sa déclaration.
- Ligne 29 : ajout d'un « = » pour annee / 400 == 0 car on souhaite comparer annee / 400 et 0, et non pas affecter une valeur.
- Ligne 30 : ajout d'un « t » à « println » et remplacement de la virgule par un + dans l'argument de la fonction.
- Ligne 32 : remplacement de la virgule en un +, même erreur que précédemment.

Au final, le code est le suivant :



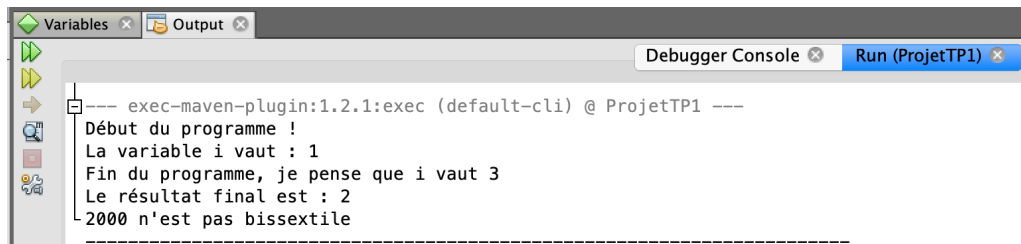
```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package edu.centralenantes.projettp1;
7
8   /**
9    *
10   * @author muruowang
11   */
12   public class ProjetTP1 {
13       public static void main(String[] args) {
14           int i = 0;
15           System.out.println(
16               "Début du programme !");
17           // premiere partie du programme
18           i *= 2;
19
20           i = i + 1;
21           System.out.println("La variable i vaut : " + i);
22           // deuxieme partie du programme
23           i += 2;
24           i = (i / 2) * 2;
25           System.out.println("Fin du programme, je pense que i vaut 3");
26           System.out.println("Le résultat final est : "+ i);
27
28           // troisieme partie : bissextile
29           int annee = 1988;
30           if ((annee / 4 == 0) && (annee / 100 != 0)) || (annee / 400 == 0)) {
31               System.out.println(annee+ " est une annee bissextile");
32           } else {
33               System.out.println(annee+" n'est pas bissextile");
34           }
35       }
36   }
```

Il compile bien et affiche le résultat suivant :



```
Output - Run (ProjetTP1)
Downloaded: http://repo.maven.apache.org/maven2/org/apache/commons/commons-parent/17/commons-parent-17.pom (31 KB at 94.6 KB/sec)
Downloading: http://repo.maven.apache.org/maven2/org/apache/commons/commons-exec/1.1/commons-exec-1.1.jar
Downloaded: http://repo.maven.apache.org/maven2/org/apache/commons/commons-exec/1.1/commons-exec-1.1.jar (52 KB at 192.9 KB/sec)
Début du programme !
La variable i vaut : 1
Fin du programme, je pense que i vaut 3
Le résultat final est : 2
1988 n'est pas bissextile
-----
BUILD SUCCESS
-----
Total time: 28.681s
Finished at: Fri Sep 06 16:32:22 CEST 2019
```

Même si le code compile, il est faux car l'année 2000 est bissextile :



Dans le code, il faut remplacer l'opérateur « / » par « % » pour que le code soit correct. En effet, il faut tester le reste de la division euclidienne et non pas le quotient. Le bloc de code modifié est le suivant :

```
int annee = 2000;
if (((annee % 4 == 0) && (annee % 100 != 0)) || (annee % 400 == 0)) {
    System.out.println(annee+ " est une annee bissextile");
} else {
    System.out.println(annee+" n'est pas bissextile");
}
```

Et les résultats sont maintenant bons :

```
Début du programme !
La variable i vaut : 1
Fin du programme, je pense que i vaut 3
Le résultat final est : 2
2000 est une annee bissextile
```

```
--- exec-maven-plugin:1.2.1:exec (default-cli) @ ProjetTP1 ---
Début du programme !
La variable i vaut : 1
Fin du programme, je pense que i vaut 3
Le résultat final est : 2
1996 est une annee bissextile
```

```
--- exec-maven-plugin:1.2.1:exec (default-cli) @ ProjetTP1 ---
Début du programme !
La variable i vaut : 1
Fin du programme, je pense que i vaut 3
Le résultat final est : 2
1997 n'est pas bissextile
```

Un programme qui compile n'est pas forcément un programme correct puisqu'il peut toujours y avoir des erreurs algorithmiques.

### Exercice 2 (1) :

Erreurs affichées par NetBeans :

- ligne 2 : On ajoute « import java.util.Scanner » qui est utilisé en ligne 18
  - ligne 30 : On ajoute « } » pour fermer instruction de « if »
  - ligne 37 : On ajoute « ; » à la fin de la ligne
  - ligne 55 : On ajoute « ) » pour fermer l'appel à la fonction de « println »
  - ligne 69 : On ajoute des guillemets avant « Decembre » pour fermer une chaîne de caractères
  - ligne 92 : On ajoute « ; » pour respecter la règle(séparer les instructions)
- Maintenant il n'y a plus de rouge, en suite on va tester.

### Debug : Fiche PriseEnMain.java

- Inversion entre les mois pairs et impairs. L'erreur est la même que pour les années bissextiles, il faut utiliser « % » au lieu de « / » pour comparer le reste de la division euclidienne à 0.

Entrez un mois sous forme de nombre (p. ex. : 1 pour Janvier, etc.)

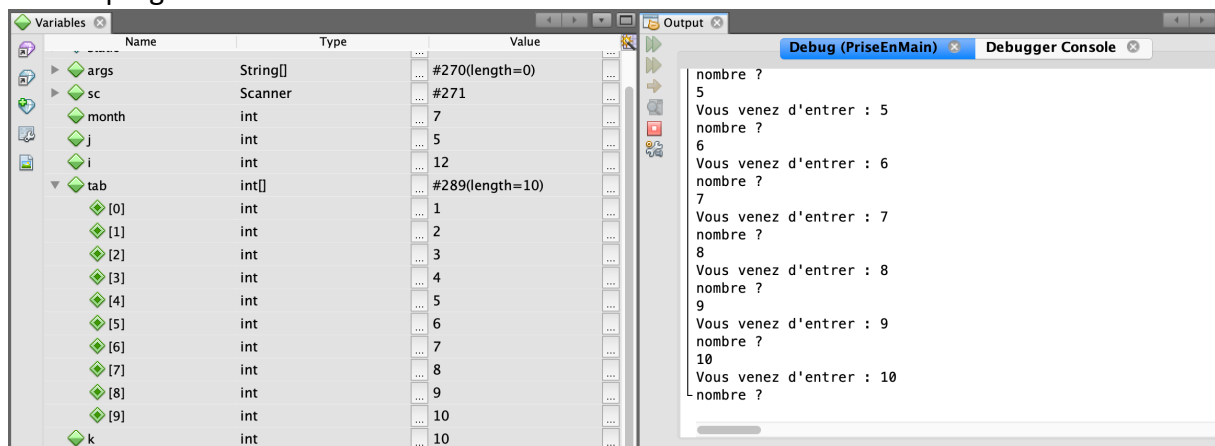
7

Le mois est Impair

Juillet

- Erreur de définition de la taille de tableau

Dans le code, on demande à l'utilisateur d'entrer 11 nombres (à cause du  $k \leq 10$ ) alors que le tableau est de taille 10, ce qui cause une erreur. Dans l'image suivante, le tableau est plein mais le programme demande encore à l'utilisateur d'entrer un nombre.



### Ce qui provoque l'erreur :

Failed to execute goal org.codehaus.mojo:exec-maven-plugin:1.2.1:exec (default-cli) on project PriseEnMain: Command execution failed. Pr

To see the full stack trace of the errors, re-run Maven with the `-e` switch.  
Re-run Maven using the `-X` switch to enable full debug logging.

For more information about the errors and possible solutions, please read the following articles:  
[Help 1] <http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException>

On corrige en changeant «  $\leq$  » par «  $<$  ».

### Debug : Compteur.java

En exécutant le main de PriseEnMain, on a l'affichage suivant qui montre que le code est faux :

```
Je suis un compteur !
Mon but est de compter jusqu'a la valeur de mon compteur qui vaut : 5
Je debute le comptage
1
3
7
Fin du comptage
```

1 erreur dans la méthode compte :

- Utilisation de l'opérateur « += » ET incrémentation de i avec « i + 1 » ce qui cause i à s'incrémenter de 2 à chaque passage dans la boucle. On remplace « += » par « = ».

Après correction, on a la sortie suivante (correcte) :

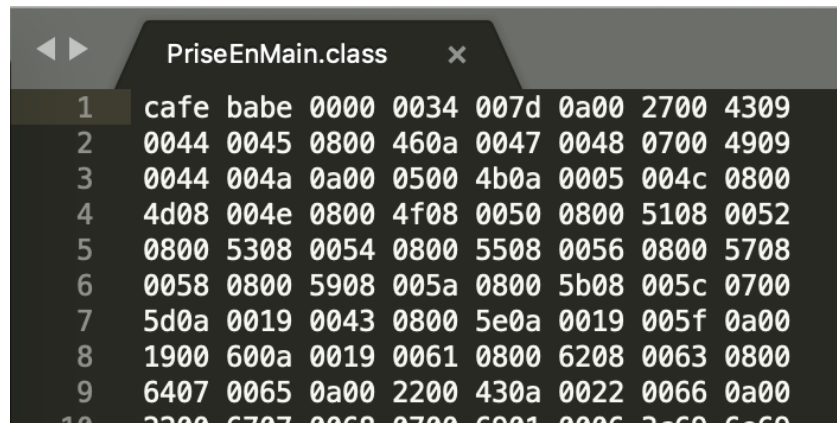
```

Je suis un compteur !
Mon but est de compter jusqu'a la valeur de mon compteur qui vaut : 5
Je debute le comptage
1
2
3
4
5
Fin du comptage

```

## Exercice 2 (2) :

PriseEnMain.class avec un éditeur de texte :



Le fichier contient du code hexadécimal, incompréhensible pour nous. Il vaut mieux rendre le fichier .java pour qu'on puisse le comprendre.

Un débogueur permet de voir plus en détail le programme et de mieux comprendre quelle ligne est fautive. Grâce à l'inspection de variables, on peut suivre le déroulement du programme et ainsi détecter à quel moment un résultat anormal apparaît. Il sert donc à détecter des erreurs au niveau de l'algorithme.

Lorsque le code ne compile pas, on peut utiliser le débogueur jusqu'à la ligne qui cause l'erreur de compilation et ainsi la résoudre. Nous avons testé de supprimer un « t » à « println » et le débogueur fonctionne encore (en sautant la ligne) même si en mode « normal » on a une erreur d'exécution.

Cependant, si le problème est plus gros et que la compilation n'est pas seulement bloquée par une ligne, le débogueur ne doit pas être très utile (on ne peut pas suivre l'exécution du programme).

## Exercice 3 :

Capture d'écran du résultat du test :

```

[1 ; 8]
[11 ; 9]
[2 ; 9]
[3 ; 10]
[0 ; 0]

```

Dans ce TP, nous avons vu comment utiliser NetBeans pour créer un projet et avons utilisé ses fonctionnalités pour corriger des erreurs de compilation ainsi que des erreurs algorithmiques grâce au débogueur. Nous avons aussi créé notre première classe, qui contenait un constructeur surchargé, ainsi que sa classe de test associée.