Haute École de Bruxelles
École Supérieure d'Informatique



# Challenge : les boucles



**Résumé**

Voici quelques conseils pour vous guider dans la résolution de tels problèmes :

— il convient d'abord de bien comprendre le problème posé ; assurez-vous qu'il est parfaitement spécifié ;
— résolvez le problème via quelques exemples précis ;
— mettez en évidence les variables **«données »**, les variables **«résultats »** et les variables de travail ;
— n'hésitez pas à faire une ébauche de résolution en français avant d'élaborer l'algorithme définitif pseudo-codé ;
— déclarez ensuite les variables (et leur type) qui interviennent dans chaque algorithme ; les noms des variables risquant de ne pas être suffisamment explicites.
— Écrivez la partie algorithmique **AVANT** de vous lancer dans la programmation en Java.
— Pour la partie Java, dessinez l'arborescence des fichier.

Avez-vous compris les boucles ? Voyons ça en relevant le défi du calendrier :

réalisez un programme permettant d'afficher sur le terminal le calendrier d'un mois et d'une année données.

Voici un exemple d'affichage pour le mois de avril 2016 :

```
Mars 2015

Lun Mar Mer Jeu Ven Sam Dim
                01  02  03
04  05  06  07  08  09  10
11  12  13  14  15  16  17
18  19  20  21  22  23  24
25  26  27  28  29  30
```

Votre programme devra demander à l'utilisateur d'entrer au clavier un mois et une année au format MM (ou M) AAAA (par exemple 4 2016) et afficher le calendrier de ce mois.

Pour déterminer le jour de la semaine où commence le mois, vous utiliserez la congruence de Zeller (en.wikipedia.org/wiki/Zeller%27s_congruence).

Pour le calendrier grégorien, actuellement utilisé dans la majeure partie du monde, la congruence de Zeller est la suivante :

où

$$h = \left( q + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + K + \left\lfloor \frac{K}{4} \right\rfloor + \left\lfloor \frac{J}{4} \right\rfloor + 5J \right) \mod 7,$$

FIGURE 1 – zeller.png

— `h` est un entier représentant le jour de la semaine (0 = samedi, 1 = dimanche, 2 = lundi, . . .),
— `q` est un entier représentant le jour du mois (de 1 à 31),
— `m` est un entier représentant le numéro du mois (3 = mars, 4 = avril, . . . Janvier et février étant considérés comme les mois 13 et 14 de l'année précédente. Donc janvier 2016 sera considéré comme 13 2015),
— `J` est un entier représentant year/100 (par exemple 20 pour l'année 2016),
— `K` est un entier représentant l'année dans le siècle, c'est à dire year mod 100 (par exemple 16 pour l'année 2016)
— `x/y` représente le résultat de **division entière** de x par y.

Ainsi, pour le 3 avril 2016, `q` = 3, `m` = 3, `J` = 20 et `K` = 16.

Le résultat `h` de la congruence de Zeller est ((3+13+16+4+5+100) mod 7) = (141 mod 7) = 1 et donc **dimanche**.

Une autre règle est nécessaire : celle des années bissextiles.

En effet, le mois de février de ces années contient 29 jours au lieu de 28. Depuis l'instauration du calendrier grégorien, sont bissextiles les années divisibles par 4 mais non divisibles par 100 ou les années divisibles par 400. Ainsi, 2016 est une année bissextile mais 2015 pas.

### Calcul de la catégorie

Écrivez les algorithmes suivants, dont voici les prototypes :

— `algorithme isLeapYear(year : entier)` → `booléen`
qui reçoit une année au format AAAA et qui renvoie vrai si cette année est bissextile et faux si elle ne l'est pas.
— `algorithme daysInMonth(month, year : entiers)` → `entier`
qui retourne le nombre de jours du mois donné en paramètre.
— `algorithme isAvailableDate(day, month, year : entiers)` → `booléen`
qui reçoit le jour, le mois et l'année d'une date et qui renvoie vrai si cette date est valide et faux si elle ne l'est pas. Une date est considérée comme valide si le mois est compris entre 1 et 12 et le jour entre 1 et (le nombre de jours du mois).
— `algorithme dayOfWeek(day, month, year : entiers)` → `entier`
qui retourne un entier correspondant au jour de la semaine (0 = lundi, 1 = mardi, . . .) pour une date donnée en paramètre.
Le paramètre year est au format AAAA, par exemple 2016. Si la date fournie est invalide, l'algorithme lancera une erreur.
— `algorithme printDay(day : entier)`
qui affiche à l'écran au format JJ le numéro du jour donné en paramètre.
Par exemple, si on donne 1 en paramètre à cette fonction, celle-ci affichera 01.

— `algorithme printCalendar(month, year : entiers)`
qui affiche à l'écran le calendrier du mois de l'année passés en paramètre, les semaines commençant par le lundi.
— `algorithme principal`
qui demande à l'utilisateur d'entrer un mois et une année au clavier.
Si le mois entré n'est pas valide, le programme redemandera la lecture jusqu'à obtenir une valeur correcte pour le mois.
Ensuite, le programme affichera à l'écran le calendrier du mois et de l'année lus.

Écrivez le code java correspondant ainsi que la javadoc.

Toutes les méthodes sauf la méthode `main` seront écrites dans une classe `g12345.boucles.UtilCalendrier`.

La méthode `main` sera écrite dans une classe `g12345.boucles.Calendrier`. Elle attrapera les éventuelles exceptions lancées et affichera alors un message d'erreur explicite.