TP2

Exercice – Javadoc

À l'instar des outils `javac` et `java`, Eclipse contient un assistant Javadoc servant d'interface à l'outil `javadoc` en ligne de commande.

1. Ajoutez des commentaires structurés à la classe que vous avez écrite au TP1. Par exemple, donnez une description de la classe, un nom d'auteur et un numéro de version.
2. Dans le menu `Project`, sélectionnez *Generate Javadoc...*
3. Dans la fenêtre qui s'ouvre, si la commande Javadoc n'est pas renseignée, fournissez le chemin vers l'outil `javadoc` (exécutable) qui se trouve dans le répertoire `bin` du JDK (*e.g.*, en utilisant le résultat de la commande `which javadoc` entrée dans un terminal), cochez la visibilité *Public* afin de générer seulement la documentation de l'interface publique des classes de votre projet, puis cliquez sur *Finish*. **Corrigé** ►*Pour Windows : C :\Program Files\Java\jdk1.7\bin\javadoc.exe*◄
4. Par défaut, la documentation (fichiers HTML) du code Java est enregistrée dans le répertoire `doc` de votre projet. Ouvrez-le et consultez la page `index.html`.
5. Rajoutez la génération de la documentation au script Ant que vous avez écrit au TP1.

Exercice – opérateurs et conditions

Soient x, y et z des variables entières. Utilisez les opérateurs de comparaison (`==`, `!=`, `<`, `>`, `<=`, `>=`) et les opérateurs de logique booléenne (`||`, `&&`, `!`) pour traduire les situation suivantes :

1. x et y sont toutes deux strictement supérieures à 3 **Corrigé** ►*((x>3) && (y>3))*◄
2. x, y et z ont la même valeur **Corrigé** ►*((x==y) && (y==z))*◄
3. La valeur de x est comprise (non strictement) entre celle de y et celle de z et z est strictement plus grand que y **Corrigé** ►*(((y<=x) && (x<=z)) && (y<z))*◄
4. Parmi les valeurs de x, y et z, deux au moins sont identiques **Corrigé** ►*(((x==y) || (y==z)) || (x==z))*◄
5. Parmi les valeurs de x, y et z, deux au plus sont identiques **Corrigé** ►*(((x !=y) || (y !=z)) || (x !=z)) OU ((x !=y) || (y !=z)) OU ((y !=x) || (x !=z)) OU ((x !=z) || (z !=y))*◄

Exercice – instructions conditionnelles et itératives

1. Écrivez une méthode statique qui retourne le jour de la semaine correspondant à l'entier passé en paramètre (0 pour dimanche, 1 pour lundi, ..., 6 pour samedi).
2. Dans la méthode principale, utilisez une boucle pour afficher dans la console tous les jours de la semaine en commençant par lundi.

3. Soit la formule de Zeller permettant d'obtenir, à partir d'une date, le jour de la semaine à laquelle elle correspond :

$$\left(\frac{13 \times mm - 1}{5} + j + aa + \frac{aa}{4} + \frac{ss}{4} - 2 \times ss\right) mod\ 7$$

où :
   - `j` est le numéro du jour dans le mois
   - `mm` est le numéro du mois dans l'année, diminué de 2 pour tous les mois sauf janvier et février, numérotés respectivement 11 et 12, et qui sont considérés comme appartenant à l'année précédente
   - `aa` est le nombre composé des 2 derniers chiffres de l'année
   - `ss` est le nombre composé des chiffres de l'année sauf les 2 derniers (`aa` et `ss` peuvent être modifiés par la remarque précédente sur janvier et février)

   Cette formule retourne un nombre compris entre 0 et 6 (0 pour dimanche, ..., 6 pour samedi), mais elle n'est valable qu'à partir du 15 octobre 1582 en raison du changement de calendrier le jeudi 4 octobre 1582 : le lendemain de ce jour est le vendredi 15 octobre 1582.

   Écrivez une méthode statique qui prend en paramètre un jour, un mois, une année (en deux parties) et retourne le résultat de la formule de Zeller.
4. Dans la méthode principale, utilisez ces méthodes pour afficher dans la console le jour d'aujourd'hui, ainsi que celui de votre anniversaire.
5. Si votre code est mal formaté, faîtes un clic-droit dans la fenêtre de code et sélectionnez *Source*, puis *Format* (ou utilisez le raccourci clavier [Ctrl] + [Maj] + [F]).
6. Soit la formule de Machin permettant d'obtenir en moyenne 1,4 décimales de pi par itération :

$$\begin{aligned}\pi &= \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}\left(16 \times \frac{1}{5^{2k+1}} - 4 \times \frac{1}{239^{2k+1}}\right) \\ &= \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}\left(S5_k - S239_k\right)\end{aligned}$$

   Écrivez une méthode statique qui prend en paramètre un entier (car on ne peut pas faire une somme à l'infini) et retourne une approximation de pi en utilisant la formule de Machin. Attention au calcul des 5^{2k+1} et 239^{2k+1} : un calcul direct par élévation à la puissance serait trop lourd, il est préférable de ramener ces calcul à des suites récurrentes, où les opérations multiplicatives se réduisent à des opérations additives. Soit sous leur forme de suite récurrente :

$$\begin{aligned}S5_k &= S5_{k-1}/5^2 \\ S239_k &= S239_{k-1}/239^2\end{aligned}$$

   De plus, notez que la valeur $(-1)^k$ correspond à un choix entre addition et soustraction, ce peut se faire avec une conditionnelle suivant la parité de k.
7. Quel est le type de retour de cette méthode ? Quel problème pose-t-il ? **Corrigé** ▶*Le calcul est effectué en réel, donc avec une précision limitée à 16 décimales. Pour calculer pi avec plus d'une dizaine de décimales, il est nécessaire d'utiliser à la place d'un `double` un objet "grand nombre" comportant le nombre avec ses milliers de décimales et des méthodes de calcul simulant les opérations (arithmétiques et entrées/sorties) sur les réels.*◀
8. Récrivez cette méthode, mais sans paramètre, le calcul s'arrêtant lorsque $S5 - S239 = 0$.
9. Dans la méthode principale, utilisez ces méthodes pour afficher dans la console une approximation de pi.

Exercice – représentation mémoire

Définissez l'état (*i.e.*, déclarez les variables d'instance) de classes modélisant :

1. Un point du plan **Corrigé** ▶*On peut représenter un point du plan par ses coordonnées cartésiennes*◀

```
public class Point { // pas de redondance
  private double abscisse;
  private double ordonnee;
}
```

Corrigé ▶*ou polaires*◀

```
public class Point { // pas de redondance
  private double rayon;
  private double angle;
}
```

2. Un segment du plan **Corrigé** ▶*On peut représenter un segment par ses deux extrémités*◀

```
public class Segment{ // pas de redondance
  private Point extremite1;
  private Point extremite2;
}
```

Corrigé ▶*ou l'une de ses extrémités puis les coordonnées polaire de l'autre extrémité dans le repère centré sur la première extrémité*◀

```
public class Segment { // pas de redondance
  private Point extremite1;
  private double rayon;
  private double angle;
}
```

Corrigé ▶*ou...*◀

3. Un rectangle du plan dont les côtés sont parallèles aux axes **Corrigé** ▶*On peut représenter un rectangle du plan dont les côtés sont parallèles aux axes par deux de ses coins opposés*◀

```
public class Rectangle {
  private Point pointBasGauche;
  private Point pointHautDroit;
}
// pas de redondance mais deux conditions a verifier : d'apres l'intitule
// des champs, on doit avoir l'abscisse de pointBasGauche inferieure ou
// egale a l'abscisse de pointHautDroit (idem pour ordonnee)
```

Corrigé ▶*ou un point, la largeur et la hauteur*◀

```
public class Rectangle { // pas redondant
  private Point pointBasGauche;
  private double largeur;
  private double hauteur;
}
```

Corrigé ▶*ou 4 points*◀

```
public class Rectangle {
  private Point ptBasG, ptBasD, ptHautG, ptHautD;
}
// redondance : on peut deduire ptBasD et pgHautG si on a juste ptBasG et
// ptHautD (et vice versa)
// Des conditions a verifier : ptBasG.ordonnee==ptBasD.ordonnee,
// ptHautG.ordonnee==ptHautD.ordonnee, ptBasG.abscisse==ptHautG.abscisse,
// ptBasD.abscisse==ptHautD.abscisse car les cotes sont paralleles aux axes
```

Pour chacune de ces classes, proposez plusieurs solutions en précisant si elles présentent de la redondance (une partie des données peut être déduite des autres et pourrait donc être omise) et si des conditions doivent être vérifiées pour que les données soient cohérentes.