Algorithmie et Java : TD2

Votre compte-rendu (qui sera votre base de révision pour les interrogations et les DS) comprendra les **réponses aux questions** et les **programmes sources** (ou parties de programmes sources) eventuellement demandés, le tout rassemblé dans un document OpenOffice. Il est fortement conseillé de faire un **glossaire** des instructions rencontrées au fur et à mesure des exercices (mot clé, syntaxe, signification, eventuellement un exemple).

1 Conditions

Pour commencer à utiliser les conditions, nous allons créer un programme Java qui analyse un nombre entier *N*. Dans un premier temps, *N* est fixé dans le programme, puis dans un deuxième temps, donné en paramètre. Le programme va afficher une série de message donnant certaines caractéristiques du nombre.

1.1 Copier le fichier **MonProgramme.java** sous le nom **Nombre.java** avec l'éditeur Kate. Modifier le code source étape par étape pour répondre aux questions suivantes. Ne pas oublier d'**enregistrer** le fichier, **compiler** le programme, **puis exécuter** à chaque modification pour tester le bon fonctionnement. *La modification du nom de fichier implique une modification dans le code source : laquelle ?*

1.2 Dans le programme, commencer par déclarer et initialiser la variable représentant le nombre (choisir un nom correct). Le premier message à afficher est : « Votre nombre est : *nombre* ». *Comment afficher ce message sur une ligne en une seule instruction ?*

1.3 Tester si le nombre est un multiple de 3. Dans ce cas, afficher le message: « Ce nombre est multiple de 3 ». *Quel opérateur mathématique permet ce test ? Comment exprimer la condition ?*

1.4 Tester si le nombre est un multiple de 5. Selon le cas, afficher un message « Ce nombre est ... » ou « Ce nombre n'est pas ... ». *Comment faire pour effectuer une seule fois le test ?*

1.5 Tester si le nombre est multiple de 4. Si c'est le cas, afficher un message comme précédemment. Sinon, tester si le nombre est multiple de 2 : afficher le message correspondant. *Essayer d'utiliser une condition imbriquée.*

1.6 Pour finir, utiliser la valeur du nombre donner en argument au programme. *Comment obtenir cette valeur ?*

2 Boucles

Pour bien comprendre les boucles, nous allons créer un programme Java qui calcule la somme et le produit des *N* premiers nombres entiers. Dans un premier temps, *N* est fixé dans le programme, puis dans un deuxième temps, donné en paramètre. Les deux types de boucle sont utilisés afin de pouvoir les comparer.

2.1 Copier le fichier **MonProgramme.java** sous le nom **SommeEtProduit.java** avec l'éditeur Kate. Modifier le code source pour répondre aux spécifications du programme. Compiler et exécuter au fur et à mesure en affichant des messages intermédiaires pour vérifier chaque étape. *La modification du nom de fichier implique une modification dans le code source : laquelle ?*

2.2 Fixer une valeur raisonnable pour *N* (inférieur à 10). Afficher chaque nombre de 1 à *N*. *Comment énumérer tous les nombres ? Quel type de boucle semble le plus adapté ?*

2.3 Calculer la somme et le produit de ces nombres, puis afficher ces deux résultats. *Comment effectuer ce calcul ? Comment stocker les résultats intermédiaires ?*

2.4 Essayer d'effectuer le même calcul en utilisant l'autre type de boucle. *Peut-on exprimer le même calcul ? Quelle forme de ce type de boucle est la plus semblable ?*

2.5 Modifier le programme pour récupérer *N* en paramètre. *A quel(s) endroit(s) faut-il modifier le programme ?*

2.6 Le choix du type de données peut s'avérer important. Utiliser des variables de type **byte** pour les calculs et observer le résultat de la somme pour *N*=15 et *N*=16. *Pourquoi le résultat est-il faux ? Une exception est-elle levée ? Quel type de donnée vaut-il mieux utiliser ?*

3 Boucles et Conditions – Suite

L'objectif du programme est de calculer les termes de la suite de Syracuse (suite entière) :

$$U_0 > 1 \qquad U_0 \in \mathbb{N}$$

$$U_n = \frac{U_{n-1}}{2} \qquad si \;\; U_{n-1} \;\; est\ pair$$

$$U_n = 3.U_{n-1} + 1 \qquad si \;\; U_{n-1} \;\; est\ impair$$

Il a été trouvé expérimentalement, mais jamais démontré, que cette suite finit toujours par atteindre la valeur 1 (en oscillant entre trois valeurs : 1, 4, 2). Dans un premier temps, le programme affiche la valeur des *N* premiers termes de la suite (U_0 fixé, puis passé en paramètre), *N* étant une valeur fixée assez grande. Dans un deuxième temps, le programme trouve et affiche le rang *N* où la

suite atteint la valeur 1 pour la première fois. *Comment exprimer la condition pair ou impair ? Quel type de boucle semble adapté pour le premier cas (N fixé) ? Pour le deuxième cas (N inconnu) ?*

4 Boucles et Conditions – Série

Nous allons maintenant écrire un programme permettant de calculer une valeur approchée de π à ϵ près, grâce à la formule :

$$\frac{\pi}{4} = \sum_{i=0}^{n} (-1)^i \frac{1}{2i+1}$$

Ce programme calcule la somme partielle de la série pour *N* croissant. Il compare le résultat à la valeur **connue** de π pour évaluer l'erreur et s'arrêter lorsque celle-ci est inférieure à ϵ . Le programme indique alors le rang *N* atteint. La valeur de ϵ est dans un premier temps fixée dans le programme, puis passée en paramètre.

4.1 Copier le fichier **MonProgramme.java** sous le nom **SuitePi.java** avec l'éditeur Kate.

4.2 Pour *N* allant de 1 jusqu'à 5, calculer les sommes partielles de la série et afficher les résultats. Pour calculer x puissance y, on peut utiliser la fonction **Math.pow(x,y)**, où x et y sont de type **double**. *Comment exprimer la somme pour éviter de faire plusieurs fois le même calcul ? Quel type de donnée utiliser ? Quel type de boucle semble adapté ?*

4.3 La valeur connue de π est **Math.PI**. Calculer l'erreur pour chaque somme partielle. Fixer une valeur pour ϵ et modifier le programme pour répondre aux spécifications (la fonction Valeur Absolue est **Math.abs(x)**). *Quel type de boucle semble adapté ?*

4.4 Récupérer la valeur de ϵ passée en paramètre. *Comment éviter une boucle infinie si la valeur donnée est trop petite pour être atteinte ?*