Module 1 –Informatique – Algorithmique et Programmation Objet

Travaux Pratiques (11), Licence 1ère Année

La récursivité

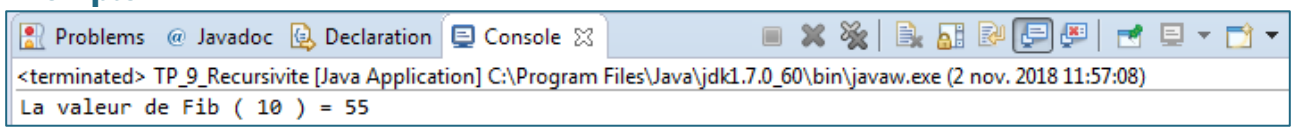
Exercice 1 Fibonacci

Soit la suite suivante, appelée suite de Fibonacci, dont la définition est :

$F_0 = 0$
$F_1 = 1$
$F_n = F_{n-1} + F_{n-2}$

Écrire une méthode **long fibonacci(int n)** qui calcule et retourne la valeur de F_n avec deux appels récursifs.

Exemple



Exercice 2 Puissance

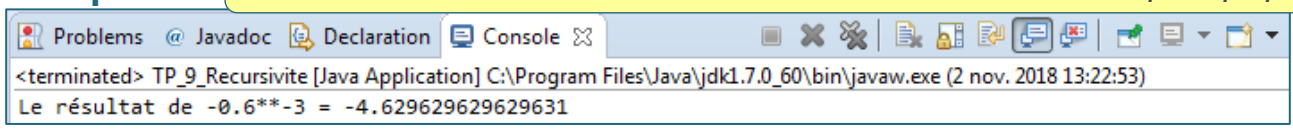
La mise à la puissance d'un réel **R** à un exposant entier **N** peut être définie par la récurrence suivante :

$R^0 = 1$
$R^N = R*R^{N-1}$

Écrire une méthode récursive **double puissance(double r, int n)** qui calcule la puissance à un exposant entier **n** (positif, nul ou négatif) d'un nombre réel **r** telle que définie précédemment.

Rappel : si **n** est un entier naturel (positif ou nul) alors $r^{-n} = \frac{1}{r^n} = \frac{1}{r} * \frac{1}{r^{n-1}}$

Exemple



Écrire une méthode récursive **double puissancePaireImpaire(double r, int n)** qui reprendre l'exercice précédent en utilisant la récurrence suivante :

$R^0 = 1$
$R^N = R*R^{N-1}$ si N est impair
$R^N = (R*R)^{N/2}$ si N est pair

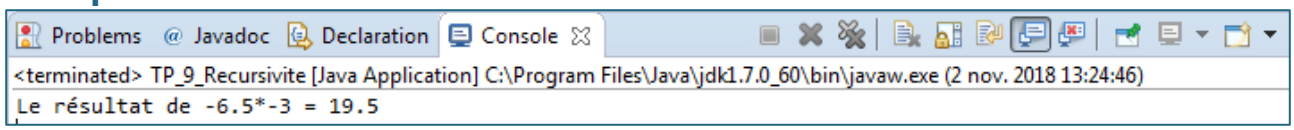
Exercice 4 Multiplication

La multiplication d'un nombre réel **R** par un entier **N** peut être réalisée par une succession d'additions selon la récurrence suivante :

$R*0 = 0$
$R*1 = R$
$R*N = R + R*(N-1)$

Écrire une méthode récursive **double multiplication(double r, int n)** qui calcule la multiplication par un entier **n** (positif, nul ou négatif) d'un nombre réel **r** telle que définie précédemment.

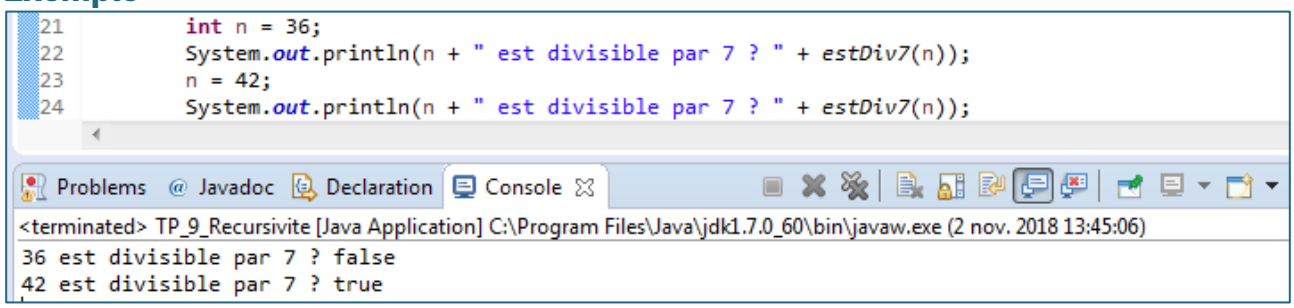
Exemple



Exercice 5 Multiple de 7

Écrire une méthode récursive **boolean estDiv7(int n)** de paramètre un entier **n** qui détermine si **n** est un multiple de **7**, sans utiliser ni **%** (le modulo) ni **/** (la division) mais uniquement par des opérations de soustractions successives.

Exemple

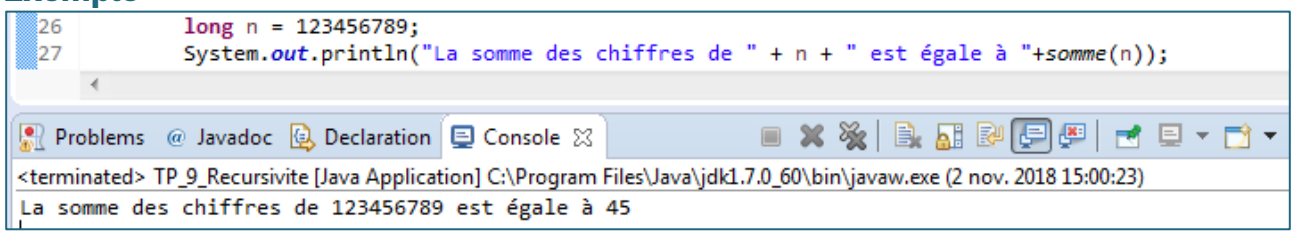


Exercice 6 Somme

6.1. Écrire une méthode récursive **long somme(long n)** de paramètre un entier naturel **n**, permettant de calculer la somme des chiffres de **n**.

Note : Pour cela on remarquera que si **n<10**, il suffit de renvoyer **n**, et que si **n≥10** on renvoie le chiffre des unités de **n** plus la somme des chiffres de **n/10**.

Exemple



On sait qu'un entier est divisible par **3** si la somme des chiffres qui le composent (et successivement la somme des chiffres des sommes intermédiaires) est divisible par **3**. Les seuls entiers compris entre **0** et **10** divisibles par **3** sont **3**, **6** et **9**. D'autre part, la somme des chiffres d'un entier supérieur à **10** est toujours strictement plus petite que cet entier.

Exemple : si **n** = **999999999999993** alors
somme(**n**) = **129**
somme(**129**) = **12**
somme(**12**) = **3** qui appartient à **[3, 6, 9]** donc **n** est divisible par **3**.

6.2. Grâce aux observations précédentes, écrire une méthode récursive **boolean estDiv3(long n)** de paramètre un entier **n**, qui détermine si **n** est divisible par 3 ou non.

Exemple

```
30        long n = 9999999999999931;
31        System.out.println(n + " est divisible par 3 ? " + estDiv3(n));
32        n = 9999999999999951;
33        System.out.println(n + " est divisible par 3 ? " + estDiv3(n));
```

Problems @ Javadoc Declaration Console

```
<terminated> TP_9_Recursivite [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (2 nov. 2018 15:07:52)
999999999999993 est divisible par 3 ? true
999999999999995 est divisible par 3 ? false
```

Exercice 7 Jeu du jackpot

On considère un jeu où l'utilisateur gagne s'il obtient trois symboles identiques. À chaque fois qu'il tente sa chance il dépense 1 euro. Lorsqu'il gagne, le jackpot est entre 10 et 100 euros (nombre aléatoire). On suppose que l'utilisateur rejoue jusqu'à ce qu'il gagne.

7.1. Écrire un programme avec une méthode **String jeu()**, sans paramètre, qui renvoie une chaîne de caractères composée de quatre traits verticaux '|' entre lesquels se trouve un symbole qui peut être soit '*' soit '(' soit 'o' avec équiprobabilité des trois symboles. Par exemple, la chaîne peut être "|*|o|*|".

Exemple

```
35        System.out.println("Chaîne retournée par l'appel à jeu() = "+jeu());
```

Problems @ Javadoc Declaration Console

```
<terminated> TP_9_Recursivite [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (2 nov. 2018 15:21:38)
Chaîne retournée par l'appel à jeu() = |*|o|*|
```

7.2. Ajouter une méthode **boolean gagne(String s)**, de paramètre une chaîne de caractères **s**, qui retourne **true** si le **2**ème, le **4**ème et le **6**ème caractères de la chaîne **s** sont tous les trois identiques, **false** sinon.

Exemple

```
37  String s = "|o|*|o|";
38  System.out.println("Pour la chaîne \"" + s + "\" gagne(\"" + s + "\") retourne " + gagne(s));
39   s = "|o|o|o|";
40  System.out.println("Pour la chaîne \"" + s + "\" gagne(\"" + s + "\") retourne " + gagne(s));
```

Problems @ Javadoc Declaration Console

```
<terminated> TP_9_Recursivite [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (2 nov. 2018 21:06:37)
Pour la chaîne "|o|*|o|" gagne("|o|*|o|" retourne false
Pour la chaîne "|o|o|o|" gagne("|o|o|o|" retourne true
```

7.3. Ajouter une méthode récursive **void joue(int essai)**, de paramètre un entier **essai** correspondant au nombre de fois où l'utilisateur a joué, qui appelle la méthode **jeu()**, affiche la chaîne obtenue, puis si l'utilisateur gagne la récursivité s'arrête et on affiche la somme gagnée (entier aléatoire entre **10** et **100**) ainsi que la somme dépensée, sinon on rejoue.

Exemple

```
Problems  Javadoc  Declaration  Console
<terminated> TP_9_Recursivite [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (2 nov. 2018 21:42:58)
|o|(|o|
|*|(|*|
|o|o|o|
Jackpot de 55 euro(s)
Vous avez dépensé 3 euro(s)
```

Exercice 8 Palindrome

On appelle palindrome un texte dont l'ordre des lettres reste le même qu'on le lise de gauche à droite ou de droite à gauche.

Principe : une chaîne **s** est un palindrome si le premier et le dernier caractères de **s** sont identiques **et** que la chaîne **s'** obtenue de **s** en excluant ces deux caractères est elle-même un palindrome. On répète le principe jusqu'à ce qu'on arrive à une chaîne de dimension une ou deux (cas d'arrêt pour une chaîne initiale de taille respectivement impaire ou paire).

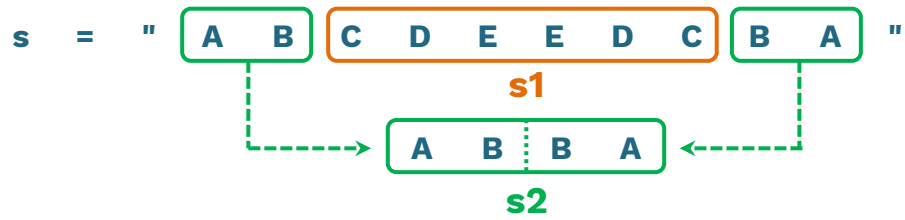
s est un palindrome si **s.charAt(0) == s.charAt(s.length()-1)** **ET** **s'** est un palindrome
[on réitère le même raisonnement sur **s'** et ainsi de suite jusqu'aux cas de base]

8.1. Écrire une méthode récursive **boolean estPalindrome(String s)** qui accepte en paramètre une chaîne **s** et qui indique si cette dernière est un palindrome ou pas selon le principe précédent.

Exemple

```
Problems  Javadoc  Declaration  Console
<terminated> TP_9_Recursivite [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (2 nov. 2018 22:51:10)
"selles" est un palindrome ? true
"celles" est un palindrome ? false
```

Une seconde solution consiste à couper le texte au quart et au trois-quarts de sa longueur et d'effectuer deux appels récursifs. L'un sur la partie centrale et l'autre sur les parties extrêmes concaténées. On répète le principe jusqu'à ce qu'on arrive à des chaînes de dimension une ou deux (cas d'arrêt).



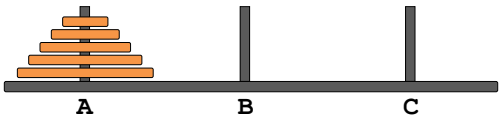
s est un palindrome si **s1** **est un palindrome** **ET** **s2** est un palindrome
[on réitère le même principe sur **s1** **et** **s2** et ainsi de suite jusqu'aux cas de base]

8.2. Réécrire la méthode précédente, en la renommant **estPalindrome2**, de façon qu'elle mette en application la deuxième solution mentionnée ci-dessus.

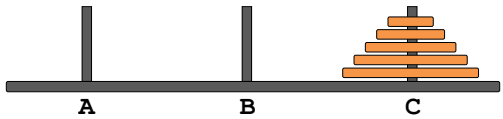
Exercice 9 Tour de Hanoï

Les tours de Hanoï sont constituées de trois (3) piquets **A**, **B** et **C**, de même diamètre dans lesquels on peut enfiler des disques ayant un trou en leur centre et de diamètres tous distincts.
Au début, **N** disques sont rangés sur le piquet **A** de manière ordonnée, le plus grand à la base et le plus petit au sommet. On désire déplacer l'ensemble de ces disques sur le piquet **C**, en utilisant le piquet **B** comme intermédiaire, rangés dans le même ordre. On ne déplace qu'un disque à la fois d'un piquet vers un autre, sachant qu'on ne peut poser un disque que si le disque sur lequel il va reposer (s'il existe) a un diamètre plus grand.

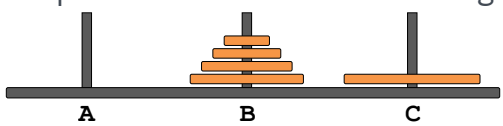
Ci-dessous les configurations de départ et d'arrivée dans le cas de cinq (5) disques avec exhibition d'une configuration intermédiaire.



Configuration de départ

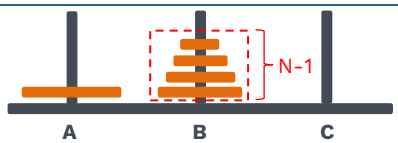


Configuration d'arrivée

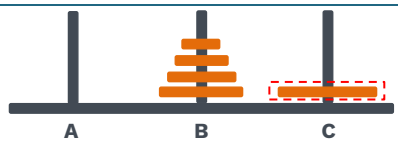


Configuration Intermédiaire

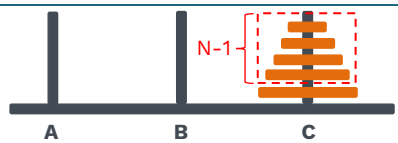
Principe : supposez que vous savez déplacer à la fois les **N-1** disques supérieurs. Ce qui donne :



Je déplace les **N-1** disques du piquet **A** vers le piquet **B**



Je déplace le **N**ème disque du piquet **A** vers le piquet **C**



Je déplace les **N-1** disques du piquet **B** vers le piquet **C**

Appliquer le même principe sur les N-1 disques et ainsi de suite jusqu'au cas de base.

Écrire une méthode récursive **void tourHanoi(int disques, char deb, char inter, char fin)** qui accepte quatre paramètres : **disques** qui indique le nombre de disques initialement, le piquet **deb** de départ, le piquet d'arrivée **fin** et le piquet intermédiaire **inter**. La méthode doit afficher les déplacements résolvant le problème.

Exemple