Schéma itératif [it]
Exercices de cours

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 17 mai 2018

Table des matières

Java - Exercices de cours (Solution)

Mots-Clés Schéma itératif ■
Difficulté ● ● ○ (2 h) ■

Remarque
Les exercices **n'utilisent pas** le module @[Algorithmes paramétrés].

1 Appréhender le cours

1.1 Le nombre de poissons / pgnpoissons

Objectif
Un poissonnier sert un client qui a demandé x Kg de poisson. Il pèse successivement différents poissons et s'arrête dès que le poids total égale ou dépasse x Kg. Cet exercice donne le nombre de poissons servis. Exemple d'exécution :

```
Poids voulu (en gr)? 1000
Poids du poisson? 350
==> 1 poisson(s) pour un poids total de 350 gr
Poids du poisson? 280
==> 2 poisson(s) pour un poids total de 630 gr
Poids du poisson? 375
==> 3 poisson(s) pour un poids total de 1005 gr
```

Comment faut-il procéder ?

Solution simple
Il faut :

- Demander le poids voulu
- Initialiser le poids total (actuel) et le nombre de poissons (actuel), tous deux à zéro.
- Tant que le poids total est inférieur au poids voulu, il faut :
  - Saisir le poids du poisson
  - Cumuler le poids du poisson au poids total
  - Incrémenter le nombre de poissons

Écrivez un programme qui saisit le poids voulu de poissons (réel).
Affichez l'invite :

```
Poids voulu (en gr)?
```

Déclarez puis initialisez à zéro les variables du poids total (réel) et du nombre de poissons (entier).

Demandez successivement le poids du poisson (réel), actualisez vos variables et affichez l'état actuel jusqu'à ce que le poids total égale ou dépasse le poids voulu (où `[x]` désigne le contenu de `x`) :

```
==> [npoissons] poisson(s) pour un poids total de [poidsTotal]
```

Testez.

Validez votre programme avec la solution.

Solution Java @[pgnpoissons.java]

```java
import java.util.Scanner;
import java.util.Locale;

class PGNpoissons {

public static void main(String[] args)
{
  Scanner input = new Scanner(System.in);
  input.useLocale(Locale.US);
  double poidsVoulu;
  System.out.print("Poids voulu (en gr)? ");
  poidsVoulu = input.nextDouble();
  double poidsTotal = 0.0;
  int npoissons = 0;
  while (poidsTotal < poidsVoulu)
  {
    double poidsPoisson;
    System.out.print("Poids du poisson? ");
    poidsPoisson = input.nextDouble();
    poidsTotal += poidsPoisson;
    ++npoissons;
    System.out.println("==> " + npoissons + " poisson(s) pour un total de " + poidsTotal
    + " gr");
  }
}

}
```

Solution commentée

`poidsTotal` est le poids total des `j` premiers poissons, `poidsPoisson` le poids du `j`-ème poisson en grammes, `npoissons` le nombre de poissons vendus après la `j`-ème itération et `poidsVoulu` le poids souhaité par l'utilisateur.

1.2 Carré d'un entier / pgcarres

Objectif
Une manière originale de calculer le carré d'un nombre entier n est de faire la **somme** des n premiers **nombres impairs**. Exemples :

- $1^2 = 1$: addition du premier impair
- $2^2 = 1 + 3 = 4$: addition des deux premiers impairs
- $3^2 = 1 + 3 + 5 = 9$: addition des trois premiers impairs
- $4^2 = 1 + 3 + 5 + 7 = 16$: addition des quatre premiers impairs
- $5^2 = 1 + 3 + 5 + 7 + 9 = 25$: addition des cinq premiers impairs

Écrivez un programme qui saisit un entier positif, variable `n`, puis affiche **tous** les carrés de 1 à `n` en utilisant la méthode des impairs. Exemples :

```
4 ==> 1 4 9 16
10 ==> 1 4 9 16 25 36 49 64 81 100
```

Contrainte : Votre programme ne doit pas comporter de boucles imbriquées.

Testez. Exemple d'exécution :

```
n? 10
1^2 = 1
2^2 = 4
3^2 = 9
4^2 = 16
5^2 = 25
6^2 = 36
7^2 = 49
8^2 = 64
9^2 = 81
10^2 = 100
```

Validez votre programme avec la solution.

Solution Java @[pgcarres.java]

```java
import java.util.Scanner;

class PGCarres {

public static void main(String[] args)
{
  Scanner input = new Scanner(System.in);
  int n;
  System.out.print("n? ");
  n = input.nextInt();
  int rs = 0;
```

```
  for (int j = 1; j <= n; ++j)
  {
    rs += 2 * j - 1;
    System.out.println(j + "^2 = " + rs);
  }
}

}
```

1.3 Calcul par récurrence d'un minimum / pgvmin

Objectif
Cet exercice calcule par récurrence le minimum d'une suite d'entiers.
Exemple d'exécution :

```
n? 5
9
4
12
-6
3
==> Le minimum est -6
```

Comment trouver le minimum de n entiers donnés au fur et à mesure ?

Solution simple
C'est le plus petit des deux nombres : minimum des $n-1$ premiers entiers donnés, n-ème entier donné.

Donnez une définition par récurrence du minimum.

Solution simple
On a :

$$minimum_n = \begin{cases} nombre_n & \text{si } nombre_n < minimum_{n-1} \\ minimum_{n-1} & \text{sinon} \end{cases}$$

Comment initialiser la suite ?

Solution simple
Une solution consiste à initialiser le minimum au premier terme saisi et à commencer la formule générale de récurrence à l'indice 2. Il s'agit d'une **initialisation à un terme utile**.

Écrivez un programme qui saisit le nombre d'entiers (dans `n`), calcule le minimum des `n` entiers donnés au fur et à mesure (dans `vmin`) puis affiche (où `[x]` désigne le contenu de `x`) :

```
==> Le minimum est [vmin]
```

Testez.

Validez votre programme avec la solution.

Solution Java @[pgvmin.java]

```java
import java.util.Scanner;

class PGVmin {

public static void main(String[] args)
{
  Scanner input = new Scanner(System.in);
  int n;
  System.out.print("n? ");
  n = input.nextInt();
  int nombre;
  nombre = input.nextInt();
  int vmin = nombre;
  for (int j = 2; j <= n; ++j)
  {
    nombre = input.nextInt();
    if (nombre < vmin)
    {
      vmin = nombre;
    }
  }
  System.out.println("==> Le minimum est " + vmin);
}

}
```

1.4 Placement d'un capital / pgcapital

Objectif
Le 1er janvier de l'année à venir, on place un capital pendant un certain nombre d'années à un certain taux. Le capital évolue suivant le système des intérêts composés.

Écrivez un programme qui saisit :

- Le capital de départ.
- Le nombre d'années.
- Le taux de placement (en %).

Puis affiche pour chaque année : la date, le capital ainsi que les intérêts obtenus au 1er janvier de cette année. Exemple d'exécution :

```
Capital de depart? 10000
Nombre d'annees? 5
Taux de placement? 0.05
Annee 1: 10500 dont 500 interets
Annee 2: 11025 dont 525 interets
Annee 3: 11576.25 dont 551.25 interets
Annee 4: 12155.0625 dont 578.8125 interets
Annee 5: 12762.815625 dont 607.753125 interets
```

Testez.

Validez votre programme avec la solution.

Solution Java @[pgcapital.java]

```java
import java.util.Scanner;
import java.util.Locale;

class PGCapital {

public static void main(String[] args)
{
  Scanner input = new Scanner(System.in);
  input.useLocale(Locale.US);
  double capital;
  System.out.print("Capital de depart? ");
  capital = input.nextDouble();
  int n;
  System.out.print("Nombre d'annees? ");
  n = input.nextInt();
  double taux;
  System.out.print("Taux de placement? ");
  taux = input.nextDouble();

  for (int j = 1; j <= n; ++j)
  {
```

```
    double interets = capital * taux;
    capital += interets;
    System.out.println("Annee " + j + ": " + capital + " dont " + interets + "
    interets");
  }
}

}
```

1.5 Prêt à taux constant / pgemprunt

Objectif

Une banque fait un prêt à une personne X pour un montant total de s_0 euros. Cette personne rembourse chaque mois un montant fixe r et paye un intérêt variable $i = t \cdot s$ avec t le taux d'intérêt mensuel (fixe) et s la somme restant à rembourser (avant déduction du remboursement mensuel). Cet exercice détermine la somme des intérêts encaissés par la banque une fois le prêt remboursé.

Écrivez un programme qui saisit les valeurs du montant de prêt s_0, du remboursement mensuel r et du taux $0 < t < 1$, puis calcule et affiche la somme des intérêts encaissés et la durée du remboursement.

Testez. Exemple d'exécution :

```
Somme pretee (0==fin)? 30000
Remboursement mensuel (>0)? 1200
Taux d'interet %? 1
Interets encaisses: 3900 (sur 25 mois)
```

Validez votre programme avec la solution.

Solution Java @[pgemprunt.java]

```java
import java.util.Scanner;
import java.util.Locale;

class PGEmprunt {

public static void main(String[] args)
{
  Scanner input = new Scanner(System.in);
  input.useLocale(Locale.US);
  double s0;
  System.out.print("Somme pretee? ");
  s0 = input.nextDouble();
  double rb;
  System.out.print("Rembousement mensuel ?");
  rb = input.nextDouble();
  double ir;
  System.out.print("Taux d'interet %? ");
  ir = input.nextDouble();
  ir /= 100.0;

  int n = (int)Math.ceil(so / rb);
  double cumul = 0.0;
  double s = s0;
  // Itère tantque le solde est positif
  while (s > 0.0)
  {
    cumul += ir * s;
```

```
    s -= rb;
  }
  // Résultat
  System.out.println("Interets encaisses: " + cumul + " (sur " + n + " mois)");
}

}
```

2 Appliquer le cours

2.1 Comptage d'entiers / pgcomptage

Objectif

Demander une série d'entiers puis compter le nombre de valeurs positives, de valeurs négatives ainsi que la plus grande et plus petite valeur. Exemple d'exécution :

```
Entrez une serie d'entiers (Finir par 0)
5 3 8 -5 2 -2 0
Nombre de positifs est 4
Nombre de negatifs est 2
Plus petite valeur est -5
Plus grande valeur est 8
```

Définissez la constante `sentinelle=0`.

Écrivez un programme qui saisit une série d'entiers jusqu'à ce que l'utilisateur introduit la valeur `sentinelle`.

Testez.

Déclarez et initialisez les compteurs du nombre de valeurs positives `npos` (entier) et celui des valeurs négatives `nneg` (entier).

Complétez la structure `TantQue` afin d'actualiser le compteur concerné selon le signe du nombre saisi.

Solution simple

Notez que le test « `Si nombre = sentinelle` » n'est jamais réalisé dans la boucle puisque c'est la condition d'arrêt du `TantQue`.

Affichez (où `[x]` désigne le contenu de `x`) :

```
Nombre de positifs est [npos]
Nombre de negatifs est [nneg]
```

Testez.

Complétez votre programme afin qu'il détermine :

- La plus grande des valeurs saisies dans `vmax`.
- Et la plus petite dans `vmin`.

Aide simple

Il faudra déclarer les variables ainsi que les initialiser (ici à la première valeur saisie).

Testez.

Validez votre programme avec la solution.

Solution Java @[pgcomptage.java]

```java
import java.util.Scanner;

class PGComptage {
final static int sentinelle = 0;

public static void main(String[] args)
{
  Scanner input = new Scanner(System.in);
  System.out.println("Entrez une serie d'entiers (Finir par " + sentinelle + ")");
  int npos = 0, nneg = 0;
  int nombre;
  nombre = input.nextInt();
  int vmin = nombre, vmax = nombre;
  while (nombre != sentinelle)
  {
    if (nombre > 0)
    {
      ++npos;
    }
    else if (nombre < 0)
    {
      ++nneg;
    }
    if (nombre < vmin)
    {
      vmin = nombre;
    }
    if (vmax < nombre)
    {
      vmax = nombre;
    }
    nombre = input.nextInt();
  }
  System.out.println("Nombre de positifs est " + npos);
  System.out.println("Nombre de negatifs est " + nneg);
  System.out.println("Plus petite valeur est " + vmin);
  System.out.println("Plus grande valeur est " + vmax);
}

}
```

2.2 Le prix de la communication / pgcxinet

Objectif

Cet exercice aide un utilisateur d'Internet à calculer le prix de sa connexion téléphonique. Pour ce le programme doit lui demander d'entrer l'heure à laquelle il s'est connecté et l'heure à laquelle il s'est déconnecté. Vous ferez les simplifications suivantes :

- Calculez avec des heures entières uniquement (pas de minutes).
- L'utilisateur n'est jamais connecté de avant minuit jusqu'à après minuit.
- La durée maximale d'une connexion est de 23 heures.

Les tarifs de connexion sont les suivants :

- Tarif 1 de 7h à 17h : 10 €/heure
- Tarif 2 de 0h à 7h et de 17h à 24h : 5 €/heure

Écrivez un programme qui demande les données et vérifie leur cohérence, puis calcule et affiche les résultats selon l'exemple d'exécution :

```
** hdebut et hfin? 10 5
OUPS... Heure debut apres la fin!
** hdebut et hfin? 10 10
OUPS... Pas connecte du tout!
** hdebut et hfin? 10 25
OUPS... Heure(s) hors journee!
** hdebut et hfin? 6 20
Connection 14 h pour 120 Euros
```

Validez votre programme avec la solution.

Solution Java @[pgcxinet.java]

```java
import java.util.Scanner;

class PGCxinet {
final static int HRDEB = 7;
final static int HRFIN = 17;
final static double TARIF_H1 = 10.0;
final static double TARIF_H2 = 5.0;

public static void main(String[] args)
{
  Scanner input = new Scanner(System.in);
  int hd = -1, hf = -1;
  boolean ok = false;
  // Saisie contraintes des heures
  while (!ok)
  {
    System.out.print("** hdebut et hfin? ");
    hd = input.nextInt();
    hf = input.nextInt();
    if (hd == hf)
```

```
    {
      System.out.println("OUPS... Pas connecte du tout!");
    }
    else if (hf < hd)
    {
      System.out.println("OUPS... Heure debut apres la fin!");
    }
    else if (hd < 0 || 24 < hf)
    {
      System.out.println("OUPS... Heure(s) hors journee!");
    }
    else
    {
      ok = true;
    }
  }
  // Nombre d'heures tarif1
  int nht1 = 0;
  for (int h = hd; h < hf; ++h)
  {
    if (HRDEB <= h && h < HRFIN)
    {
      ++nht1;
    }
  }
  // Nombre d'heures tarif2
  int nht2 = (hf - hd) - nht1;
  double prix = nht1 * TARIF_H1 + nht2 * TARIF_H2;
  System.out.println("Connection " + (nht1 + nht2) + " h pour " + prix + " Euros");
}

}
```

3 Approfondir le cours

3.1 Approximation du sinus / pgapproxsin

Propriété

Pour un réel donné x, la valeur de $\sin(x)$, représentant un angle en radians, peut être approximé par le développement limité suivant :

$$S_n = x - \frac{x^3}{3!} + \frac{x^5}{5!} - ... + (-1)^i \frac{x^{2i+1}}{(2i+1)!} + ... + (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!}$$

où n est le nombre de termes.

Écrivez un programme permettant de calculer la valeur approchée de $\sin(x)$, où x est un réel représentant un angle en radians, pour un nombre de termes donné.

Solution simple

L'approximation du sinus, pour un angle, s'obtient, selon la formule donnée, par des additions successives de termes, positifs ou négatifs. La suite récurrente $(sinus_i)_{i\in\mathbb{N}}$ est définie par :

$$\begin{cases} sinus_0 = x \\ sinus_i = sinus_{i-1} + (-1)^i \dfrac{x^{2i+1}}{(2i+1)!} \text{ pour } i \geq 1 \end{cases}$$

Chaque nouveau terme de la suite récurrente nécessite donc le calcul d'un terme du type :

$$(-1)^i \frac{x^{2i+1}}{(2i+1)!}$$

Il serait donc nécessaire de calculer la puissance d'une valeur donnée, ainsi que le factoriel d'un entier. Ces calculs sont gourmands en ressources, mais nous remarquons que le calcul d'un terme $(-1)^i \frac{x^{2i+1}}{(2i+1)!}$ peut être déduit du terme précédent $(-1)^{i-1} \frac{x^{2i-1}}{(2i-1)!}$. Effectivement :

$$2i+1 = 2 + 2i - 1 \text{ et donc } x^{2i+1} = x^2 x^{2i-1}$$
$$\text{et} \qquad (2i+1)! = (2i+1)2i(2i-1)!$$

Ceci permet de définir une deuxième suite récurrente pour le calcul de ces termes :

$$\begin{cases} t_0 = x \\ t_i = -t_{i-1} \times \frac{x^2}{2i(2i+1)} \text{ pour } i \geq 1 \end{cases}$$

Finalement, le calcul s'exprime sous la forme de deux suites récurrentes, $(sinus_i)_{i\in\mathbb{N}}$ et $(t_i)_{i\in\mathbb{N}}$ (la dernière étant donnée ci-dessus) :

$$\begin{cases} sinus_0 = t_0 \\ sinus_i = sinus_{i-1} + t_i \text{ pour } i \geq 1 \end{cases}$$

Conclusion : quand i augmente d'une unité, on passe au terme en i au suivant, d'une part en ajoutant deux facteurs x au numérateur, et d'autre part, en ajoutant les deux prochains nombres dans le calcul du factoriel au dénominateur.

La valeur approchée de $\sin(x)$ est calculée avec n termes de la somme, donc le cas d'arrêt est donné par l'expression $i = n$.

Le nombre de termes de la suite est connu à l'avance, donc une répétitive `Pour` peut être utilisée.

Testez. Exemple d'exécution :

```
x (en radians)? 0.5
Nombre de termes? 5
sin(0.5) = 0.4794255386164159
sin math = 0.479425538604203
```

Validez votre programme avec la solution.

Solution Java @[pgapproxsin.java]

```
import java.util.Scanner;
import java.util.Locale;

class PGApproxsin {

public static void main(String[] args)
{
  Scanner input = new Scanner(System.in);
  input.useLocale(Locale.US);
  double x;
  System.out.print("x (en radians)? ");
  x = input.nextDouble();
  int n;
  System.out.print("Nombre de termes? ");
  n = input.nextInt();
  double terme = x;
  double approx = terme;
  for (int j = 1; j < n; ++j)
  {
    terme = -terme * x * x / (2 * j * (2 * j + 1));
    approx += terme;
  }
  System.out.println("sin(" + x + ") = " + approx);
  System.out.println("sin math = " + Math.sin(x));
}

}
```