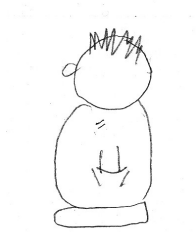
Java à l'école

Exercices et proposition de solutions

11TG, Première année de programmation



Version 1.3.4b, Jens Getreu

Table des matières

1 Introduction

Les exercices de ce recueil sont basés sur un concept de programmation appelé « Architecture Modèle/Vue/Contrôleur»[1], consistant entre autres à séparer l'interface utilisateur d'un programme de sa logique interne. Une interface utilisateur est la partie du programme qui assure la communication avec son utilisateur. Le plus simple à réaliser est l'interface textuelle basée sur les deux instructions Java suivantes:
L'instruction `System.out.println(s1)` affiche le texte de la variable `s1` sur l'écran, tandis que l'instruction `String s2 = sc.nextLine()` permet d'assigner un texte saisi par l'utilisateur sur clavier à la variable `s2`. La dernière instruction existe en 3 variantes selon le type de donnée saisie:

- `String s2 = sc.nextLine()` pour les chaînes de caractères,
- `int n1 = sc.nextInt()` pour les nombres entiers et
- `double n2 = sc.nextDouble()` pour les nombre réels.

Toute communication avec l'utilisateur est programmée dans la classe `InterfaceUtilisateur`. Ainsi, les deux instructions ci-dessous ne peuvent apparaître que dans cette classe. Voici le modèle de la classe applicable pour tous les exercices :

```
import java.util.*;
public class InterfaceUtilisateur
{
   public static void main(String[] args)
   {
      Scanner sc = new Scanner(System.in);
      <insérez votre code source ici>
   }
}
```

Bien que la communication avec l'utilisateur soit assurée par la classe `InterfaceUtilisateur` , la logique du programme est réalisée principalement dans une classe à part.

Exercice d'introduction

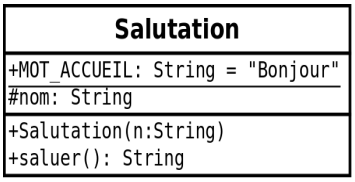
Nous réalisons une variante du célèbre programme « *hello world* »: Le programme demande d'abord à l'utilisateur de saisir son nom (par exemple « *Lina* ») et affiche ensuite une salutation personnalisée: « *Bonjour Lina. Comment allez-vous ?* », comme le montre la capture d'écran ci-contre.

```
SALUTATION
Veuillez saisir votre nom : Lina
Bonjour Lina. Comment allez-
vous ?
```

Solution

La logique du programme est codée dans la classe `Salutation`.

Comme le montre le diagramme de classe ci-contre, toutes les instances (objets) de la classe `Salutation` possèdent une constante `MOT_ACCUEIL` du type String (chaîne de caractères) , un attribut (variable d'instance) `Nom` du type String, un constructeur `Salutation()` avec un paramètre `n` du type String et une méthode `saluer()` renvoyant également un String.



1 Voir : http://fr.wikipedia.org/wiki/Modèle-Vue-Contrôleur

La figure suivante montre le code source de la classe donnant des informations plus détaillées sur son fonctionnement :

```
001: public class Salutation
002: {
003:     public static final String MOT_ACCUEIL = "Bonjour ";
004:
005:     protected String nom;
006:
007:     public Salutation(String n)
008:     {
009:        nom = n;
010:     }
011:
012:     public String saluer()
013:     {
014:         return MOT_ACCUEIL + nom + ". Comment allez-vous ?";
015:     }
016: }
```

Linge	Remarque
001-016	Définition de la classe `Salutation`.
003	Déclaration de la constante `MOT_ACCUEIL` initialisée à *"Bonjour "*. *`MOT_ACCUEIL`* est un attribut de la classe `Salutation`.
005	Déclaration de l'attribut `nom`.
007-010	Définition du constructeur de la classe `Salutation`: Lors de la création d'une instance (objet) de la classe `Salutation` , nous devons passer un paramètre `n` au constructeur contenant une chaîne de caractères. Un constructeur, ici `Salutation()`, est une méthode spéciale qui est exécutée au moment de la création (instanciation) d'un objet. Un constructeur n'a jamais de valeur de retour.
009	Cette chaîne de caractères est ensuite stockée dans une variable propre à l'objet: `nom`. Une telle variable est appelée « attribut » ou « variable d'instance ».
012-15	Définition de la méthode `saluer()`. Elle utilise l'attribut `nom` pour composer un message personnalisé en concaténant le mot "*Bonjour*", la chaîne de caractères référencée par la variable `nom` et le texte *"Comment allez-vous ?*" Le résultat de la concaténation est renvoyé comme valeur de retour du type `String`.

Voici le code source de la classe `InterfaceUtilisateur` assurant la communication avec l'utilisateur:

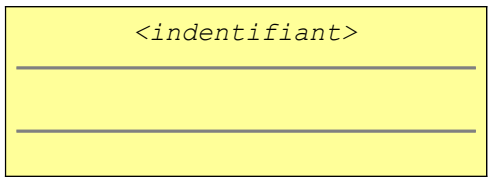
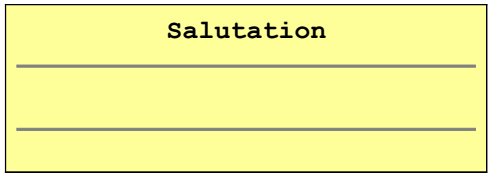
```
001: import java.util.*;
002: public class InterfaceUtilisateur
003: {
004:    public static void main(String[] args)
005:    {
006:          Scanner sc = new Scanner(System.in);
007:
008:          System.out.println("SALUTATION");
009:          System.out.print("Veuillez saisir votre nom : ");
010:          String n = sc.nextLine();
011:
012:          Salutation s = new Salutation(n);
013:          System.out.println(s.saluer() );
014:    }
015: }
```

Linge	Remarque
001-006	Instructions standards et identiques au modèle présenté ci-dessus. [2]
008-009	Les messages « *SALUTATION* » et « *Veuillez saisir votre nom :* » sont affichés à l'écran.
010	L'instruction `String n = sc.nextLine();` affecte la chaîne de caractères saisie au clavier à la variable `n`.
012	L'instruction `Salutation s = new Salutation(n);` crée une instance de la classe `Salutation` (=objet) appelée `s` par la suite. L'objet est créé en passant la variable `n` au constructeur.
013	La variable `s` sert à référencer l'objet créé et permet l'accès à sa méthode par l'expression: `s.saluer()` . Le résultat de la méthode `saluer()`, c'est-à-dire *"Bonjour Lina. Comment allez-vous ?",* est affiché par l'instruction `System.out.println()`.

2 De plus amples informations en classe de 12ème.

Éléments du langage Java

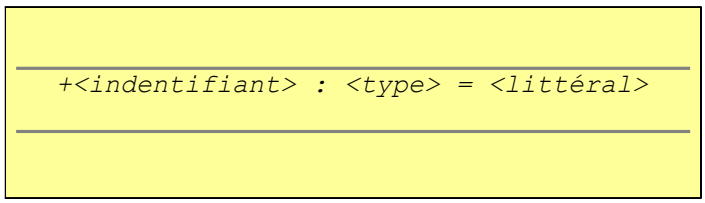
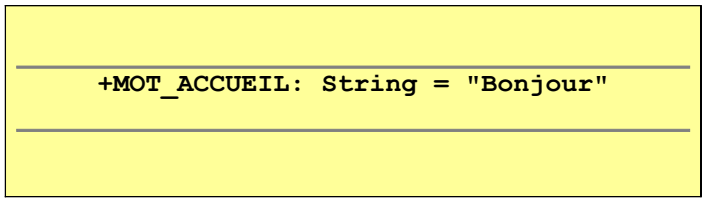
Les tableaux récapitulatifs ci-dessous montrent les conventions utilisées dans ce recueil d'exercices en se référant à l'exercice d'introduction.

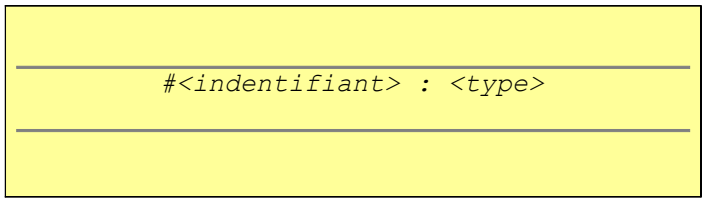
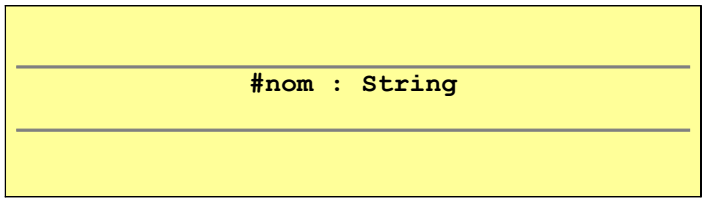
Classes	
Définition du terme	Une classe est une description abstraite des données et du comportement d'objets, ces objets étant similaires. Les représentants de la classe sont appelés des instances ou simplement objets.
Contexte	Définition à l'intérieur d'un fichier `.java` portant le même nom que la classe.
Nommage	• 1ère lettre en majuscule • Mélange de minuscules, de majuscules avec la première lettre de chaque mot en majuscule • Donner des noms simples et descriptifs
Exemples de noms (identifiants)	`InterfaceUtilisateur, Salutation, NombrePositif, Piste, NombrePremier`
Syntaxe diagramme de classe UML	
Exemple de diagramme de classe	

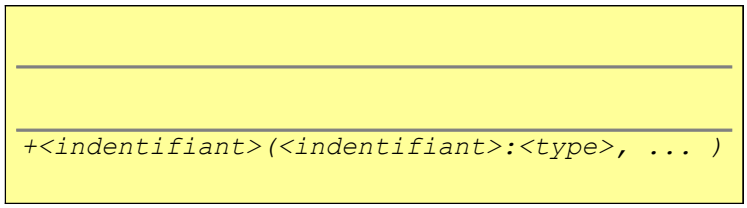
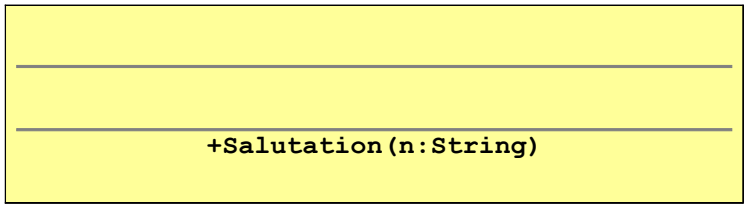
Syntaxe de définition	`public class <indentifiant>`
	`{ ...`
	`}`

Exemples de définition	`public class Salutation`
	`{ ...`
	`}`
	
	`public class InterfaceUtilisateur`
	`{ ...`
	`}`

Exemples d'utilisation	`Salutation s = new Salutation(n);` `System.out.println(s.saluer() );`

Constantes	
Définition du terme	Un attribut spécial dont la valeur reste invariable après son initialisation.
Contexte	Définition à l'intérieur d'une classe.
Nommage	• Tout en majuscules • Séparer les mots par un souligné (underscore) : _ • Donner des noms simples et descriptifs
Exemples de noms (identifiants)	`MOT_ACCUEIL, POS_MAX, POS_MIN, NON_VALIDE`
Syntaxe diagramme de classe UML	
Exemple de diagramme de classe	
Syntaxe de déclaration avec initialisation	`public static final <type> <identifiant> = <littéral>;`
Exemples de déclaration avec initialisation	`public static final String MOT_ACCUEIL = "Bonjour ";` `public static final int POS_MAX = 20;`
Exemples d'utilisation	`return MOT_ACCUEIL + nom + ". Comment allez-vous ?";` `if (i < POS_MAX) ...`

Attributs (synonyme : variables d'instances)	
Définition du terme	Une variable d'instance précise l'état d'un objet auquel elle se réfère. Deux objets différents, même appartenant à la même classe, peuvent avoir des valeurs différentes dans leurs variables d'instance respectives. « Ce qu'un objet d'une classe a ou caractérise».
Contexte	Définition à l'intérieur d'une classe.
Nommage	• 1ère lettre en minuscule • Mélange de minuscules, de majuscules avec la première lettre de chaque mot en majuscule • Donner des noms simples et descriptifs • Ne pas commencer les noms avec '$' ou '_' bien que ce soit possible.
Exemples de noms (identifiants)	`nom, coordX, coordY, nombreZéros`
Syntaxe diagramme de classe UML	
Exemple de diagramme de classe	
Syntaxe de déclaration	`protected <type> <identifiant>;`
Exemple de déclaration	`protected String nom;`
Exemples d'utilisation	`nom=n;` `return MOT_ACCUEIL + nom + ". Comment allez-vous ?";`

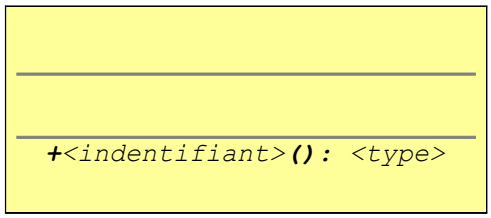
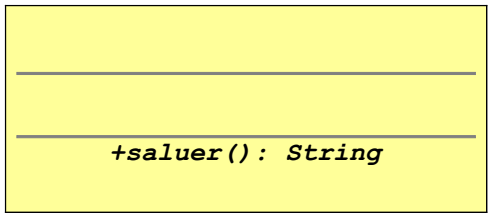
Constructeurs	
Définition du terme	Le constructeur est une méthode spéciale appelé lors de la création de l'objet.
Contexte	Définition à l'intérieur d'une classe.
Nommage	• Les noms des constructeurs doivent être identiques au nom de la classe.
Exemples de noms (identifiants)	`Salutation(), NombrePositif(), Piste(), NombrePremier()`
Syntaxe diagramme de classe UML	
Exemple de diagramme de classe	

Syntaxe de définition[3]	`public <identifiant>(<identifiant>:<type>, ...)`
	`{`
	`  ...`
	`}`

Exemple de définition	`public Salutation(String n)`
	`{`
	`  Nom = n;`
	`}`

Exemples d'utilisation	`Salutation s = new Salutation(n);`

3 Notons qu'en règle générale les constructeurs n'ont pas de la valeur de retour. Il n'est pas nécessaire d'indiquer cette absence par le mot clé `void`.

Méthodes	
Définition	Descriptif et implémentation des fonctionnalités d'une classe. « Ce qu'un objet d'une classe peut faire ».
Contexte	Définition à l'intérieur d'une classe.
Nommage	• Les noms de méthodes (fonctions) doivent exprimer une action. • Choisir de préférence des verbes. • 1ère lettre en minuscule • Mélange de minuscules, de majuscules avec la première lettre de chaque mot en majuscule
Exemples de noms (identifiants)	`saluer(); afficher(); dessiner(); getCoordX(); setCoordY(); isPremier(); isValide()`
Syntaxe diagramme de classe UML	
Exemple de diagramme de classe	

Syntaxe de définition[4]	`public <type> <identifiant>()`
	`{`
	`  ...`
	`}`

Exemple de définition[5]	`public String saluer()`
	`{`
	`  return MOT_ACCUEIL + nom + ". Comment allez-vous ?";`
	`}`

Exemple d'utilisation	`System.out.println( s.saluer() );`

4 Pour réduire la complexité de ce chapitre, les méthodes exposées ici n'ont pas de paramètres. Pour plus d'informations sur le passage des paramètres, voir les pages « Constructeur » et « Variables locales ». Pour définir un paramètres il suffit d'introduire dans les parenthèses vides la déclaration d'une variable locale, par exemple `public String saluer(int n){...}` et donc `System.out.println(s.saluer(3))`

5 Dans cet exemple la méthode retourne une chaîne de caractères composée de trois chaînes. Cette valeur de retour est du type `String`.

Variables locales	
Définition	Une variable locale est une variable qui ne peut être utilisée que dans la fonction ou le bloc où elle est définie.
Contexte	Définition à l'intérieur d'une méthode ou comme paramètre d'une méthode.
Nommage	• Voir « Attributs » • Variable d'une seule lettre minuscule de préférence.
Exemples de noms (identifiants)	`i, n, x, y, s`
Syntaxe de déclaration avec initialisation	*`<type> <identifiant> = <expression>;`*
Exemple de déclaration avec initialisation	`String n = sc.nextLine();`
Exemple d'utilisation	`Salutation s = new Salutation(n)`
Exemple de déclaration avec initialisation	`int i = 0;`
Exemple d'utilisation	`i = i+1;`

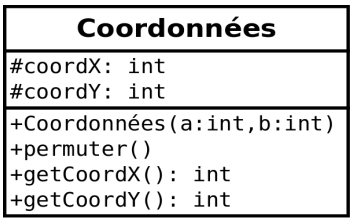
Exemple de déclaration avec utilisation[6]	`public double additionner(double y)`
	`{`
	`  mémoireInterne = mémoireInterne + y;`
	`}`

6 Une variable locale peut aussi servir à transmettre des informations à une méthode lors de son appel. On parle ainsi d'un *paramètre* ou *argument*.
La règle générale de Java est que les paramètres (ou arguments) sont passés par valeur. Cela signifie que l'appel de méthode se fait **par copie** des valeurs passées en argument et que chaque appel de méthode dispose de sa propre version des paramètres.

2 Permutation

Écrire un programme permettant de lire deux valeurs pour les variables x et y. Permuter ensuite les contenus des deux variables. Afficher les contenus des deux variables avant et après l'opération de permutation.

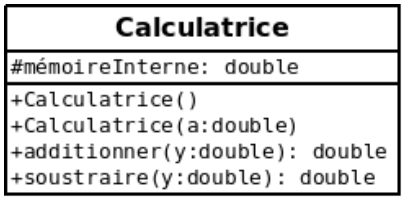
```
PERMUTATION

Veuillez saisir un nombre entier x : 3
Veuillez saisir un nombre entier y : 6

Les variables avant la permutation :
y=6 x=3
Les variables après la permutation :
y=3 x=6
```



3 Mini-calculatrice

L'utilisateur entre deux nombres x et y sur le clavier. Ensuite, un des caractères « + » ou « - » est saisi. L'utilisateur est guidé lors de la saisie par des messages adéquats. Selon le caractère entré, les nombres x et y sont additionnés ou soustraits. Le résultat de l'opération s'affiche à l'écran.

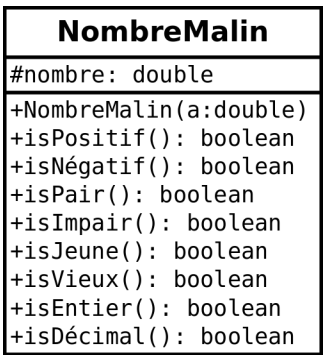
```
MINICALCULATRICE
Veuillez saisir un nombre réel x : 2.3
Veuillez saisir un nombre réel y : 1.2
Veuillez saisir un opérateur '+' ou '-':
+
x+y = 3.5

MINICALCULATRICE
Veuillez saisir un nombre réel x : 4.2
Veuillez saisir un nombre réel y : 3.0
Veuillez saisir un opérateur '+' ou '-':
-
x-y = 7.2

MINICALCULATRICE
Veuillez saisir un nombre réel x : 3.2
Veuillez saisir un nombre réel y : 9
Veuillez saisir un opérateur '+' ou '-':
xyz
L'opérateur 'xyz' n'est pas défini.
```



4 Nombre malin

Écrire une classe `NombreMalin` représentant un nombre réel. Développer des méthodes qui renseignent sur ce nombre, par exemple: s'il est pair ou impair, positif ou négatif, inférieur ou supérieur à 30, ou s'il est entier ou avec une partie fractionnaire. Le programme à réaliser permet à l'utilisateur de saisir un nombre réel, à partir duquel une instance de la classe est créé et un sous-ensemble des messages suivants est affiché:
Je suis négatif, je suis pair , je suis jeune (<30), je suis un nombre entier, je suis positif , je suis impair, je suis vieux (>=30), j'ai une partie fractionnaire.
Attention: Seuls les nombres entiers sont pairs ou impairs!

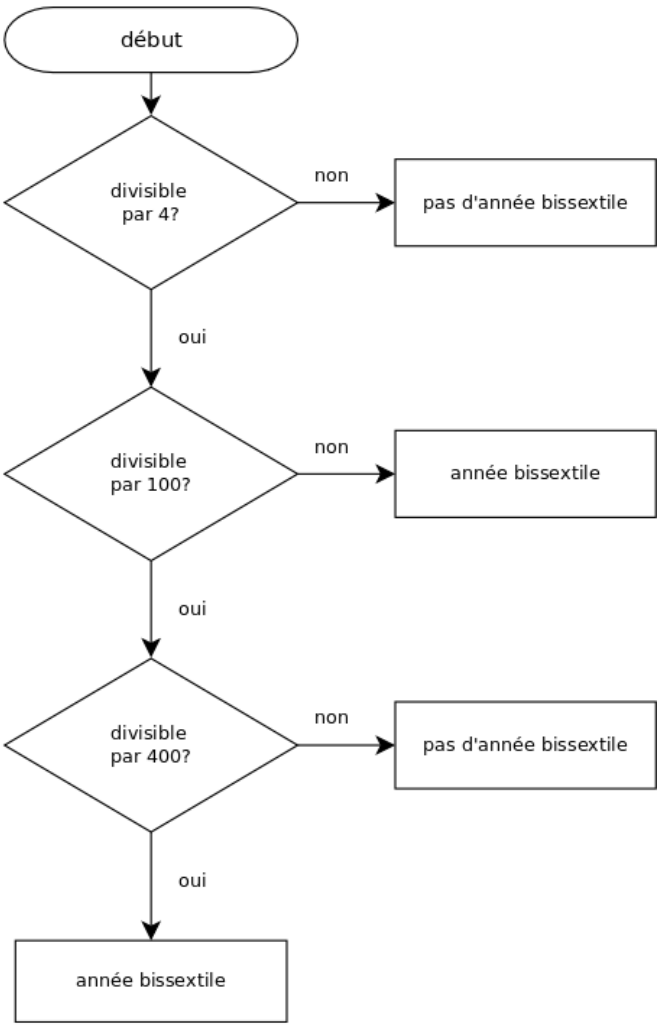
```
NOMBRE MALIN
Veuillez saisir un nombre réel : -8
Bonjour, je suis un nombre malin.
Je me connais bien:
Je suis négatif.
Je suis pair.
Je suis jeune (<30).
Je suis un nombre entier.

NOMBRE MALIN
Veuillez saisir un nombre réel : 32,5
Bonjour, je suis un nombre malin.
Je me connais bien:
Je suis positif.
Je suis vieux (>=30).
J'ai une partie fractionnaire.
```



5 Année bissextile

Réaliser un programme permettant à l'utilisateur d'entrer une année sous forme de nombre entier (par exemple 2005). Le programme affiche ensuite sous forme de message si l'année est une année bissextile ou non.

Détermination des années bissextiles



Organigramme de programmation (Programmablaufplan)

```
ANNÉE BISSEXTILE
Veuillez saisir une année : 2007
Il ne s'agit pas d'une année bissextile.

ANNÉE BISSEXTILE
Veuillez saisir une année : 2004
Il s'agit d'une année bissextile.
```

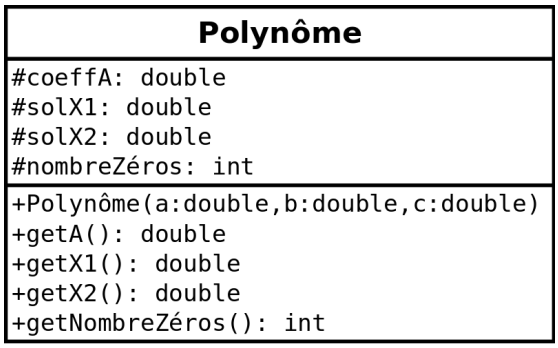
Année
#an: int
+Année(x:int) +isAnnéeBissextile(): boolean

6 Équation du second degré

Le programme à réaliser doit résoudre une équation du second degré de la forme: $ax^2+bx+c=0$. Les coefficients a, b et c sont entrés par l'utilisateur. La solution de l'équation s'affiche à l'écran. Rechercher une représentation interne du polynôme initial ax^2+bx+c sous sa forme factorisée $a(x-x1)(x-x2)$. Vérifiez votre programme avec les valeurs ci-dessous.

a	b	c	a	x1	x2	Nombre de racines (=nombre de zéros)
2	4	-30	2	3	-5	2
2	8	8	2	-2	-2	1
2	1	1	2	-	-	0

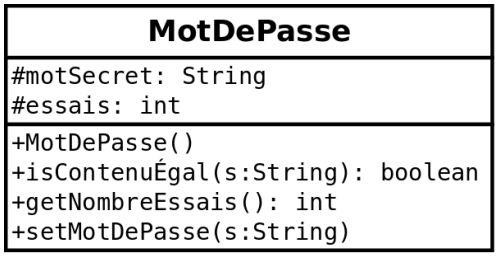
```
RÉSOUDRE L'ÉQUATION DE SECOND DEGRÉ
 a*X^2 + b*X + c = 0

Veuillez saisir le coefficient a : 1
Veuillez saisir le coefficient b : 3
Veuillez saisir le coefficient c : -4
Deux solutions : 1 et -4

RÉSOUDRE L'ÉQUATION DE SECOND DEGRÉ
 a*X^2 + b*X + c = 0

Veuillez saisir le coefficient a : 1
Veuillez saisir le coefficient b : -6
Veuillez saisir le coefficient c : 9
Une solution : 3
```



7 Vérifier un mot de passe

Réaliser un programme permettant de vérifier un mot de passe secret (« hommik »). L'utilisateur a 3 essais pour entrer le mot de passe correct. Il est guidé lors de la saisie par des messages adéquats. Extension possible: L'utilisateur doit d'abord changer son mot de passe.

```
VÉRIFIER LE MOT DE PASSE
Veuillez saisir votre mot de passe : bon
Erreur. Il vous reste 2 essais.
Veuillez saisir votre mot de passe : jour
Erreur. Il vous reste 1 essais.
Veuillez saisir votre mot de passe : bonjour
Erreur. Il vous reste 0 essais.
Le mot de passe n'est pas correct.


VÉRIFIER LE MOT DE PASSE
Veuillez saisir votre mot de passe : öö
Erreur. Il vous reste 2 essais.
Veuillez saisir votre mot de passe : hommik
Le mot de passe est correct.
```



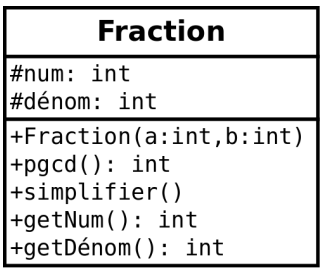
8 Simplifier une fraction

Réaliser un programme permettant de saisir le numérateur et le dénominateur d'une fraction. L'utilisateur est guidé lors de la saisie par des messages adéquats. Le programme affiche la fraction simplifiée. Utiliser l'algorithme d'Euclide.

Exemple 1: *pgcd(24,9)*	Exemple 2: *pgcd(12,7)*
24 modulo 9 = 6 9 modulo 6 = 3 6 modulo 3 = 0 *résultat = 3*	12 modulo 7 = 5 7 modulo 5 = 2 5 modulo 2 = 1 2 modulo 1 = 0 *résultat = 1*

Remarque : réaliser et tester d'abord la méthode pgcd().
Extension possible: ajouter une méthode permettant de porter la fraction à une plus grande expression.

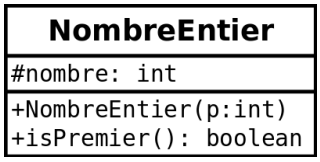
```
SIMPLIFIER UNE FRACTION
Veuillez saisir le numérateur n : 12
Veuillez saisir le dénominateur d : 20

Le numérateur simplifié n = 3
Le dénominateur simplifié d = 5
```



9 Nombre premier

Écrire un programme permettant de tester si un nombre saisi est un nombre premier ou non. Un nombre premier est un nombre entier supérieur à 1 ayant seulement deux diviseurs : 1 et lui-même. Remarque : cette exercice peut être traité indépendamment ou comme extension de l'exercice « nombre malin ».

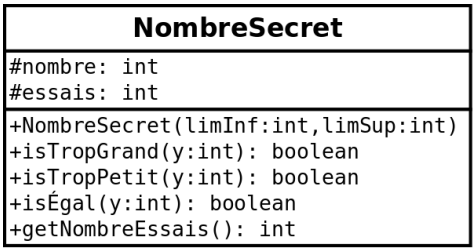
```
NOMBRE PREMIER
Veuillez saisir un nombre entier p : 12
p n'est pas nombre premier.

NOMBRE PREMIER
Veuillez saisir un nombre entier p : 13
p est nombre premier.
```



10 Jeu « deviner nombre »

Développer un jeu où l'ordinateur choisit un nombre aléatoire compris entre deux limites entrées par l'utilisateur. L'utilisateur doit deviner ce nombre en trois essais tout en étant guidé lors de la saisie par des messages adéquats. L'ordinateur donne les indications suivantes « trop grand », « trop petit », « vous avez gagné » et « vous avez perdu ».

```
DEVINER NOMBRE:
Limite inférieur : 3
Limite supérieur : 6
*** Le jeu démarre, vous avez 3 essais ***
Votre estimation : 3
   trop petit ...
Votre estimation : 4
   trop petit ...
Votre estimation : 6
   trop grand...
Vous avez perdu.
```



11 Jeu du loup

Le joueur (pion X) essaie d'attraper un loup (pion O). Avant de commencer la course, le joueur donne au loup une certaine avance. A chaque étape, le joueur indique le nombre de pas à faire pour avancer tout en sachant que le loup avance lui-même de 1 à 5 pas aléatoirement. Le jeu se termine avec le message *vous avez rattrapé le loup* ou le message *le loup vous à échappé* si un des coureurs à quitté la piste avant.

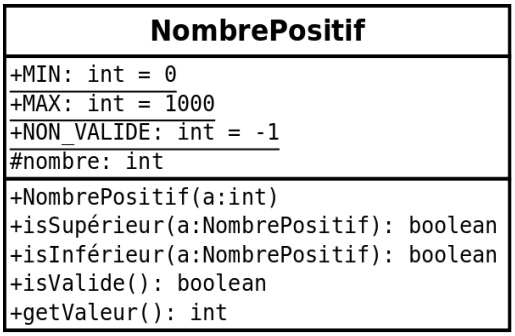
```
JEU DU LOUP
Veuillez saisir l'avance de O sur X : 5
|X____O_____________| Votre pas : 5
|_____X___O_________| Votre pas : 5
|_________ *________|
Vous avez rattrapé le loup!

JEU DU LOUP
Veuillez saisir l'avance de O sur X : 8
|X_______O__________| Votre pas : 5
|_____X___O_________| Votre pas : 4
|_________X_O_______| Votre pas : 3
|_____________XO_____| Votre pas : 4
|__________________X_O| Votre pas : 3
|____________________X
Le loup vous a échappé
```

Piste
+POS_MAX: int = 20 +PAS_MAX: int = 5 +PAS_MIN: int = 1 #posO: int #posX: int
+Piste(avanceO:int) +avancer(pasX:int): boolean +dessiner(): String +getPosX(): int +getPosO(): int

12 Minimum-Maximum

Écrire un programme affichant le minimum et le maximum d'une série de nombres entiers positifs entre 0 et 1000 saisie par l'utilisateur. L'utilisateur indique la fin de la saisi en saisissant -1.

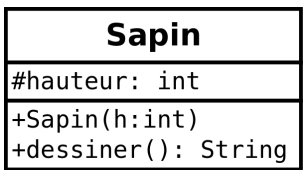
```
MIN-MAX
Veuillez saisir un nombre entier entre 0 et 1000 (-1 pour terminer) : 4
Veuillez saisir un nombre entier entre 0 et 1000 (-1 pour terminer) : 9
Veuillez saisir un nombre entier entre 0 et 1000 (-1 pour terminer) : 3
Veuillez saisir un nombre entier entre 0 et 1000 (-1 pour terminer) : -1
Le minimum = 3
Le maximum = 9
```



13 Sapin de Noël

Écrire un programme affichant la figure suivante à `m` lignes représentant un sapin de Noël. Le paramètre `m` est saisi par l'utilisateur.

```
SAPIN DE NOËL
Veuillez saisir la hauteur : 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

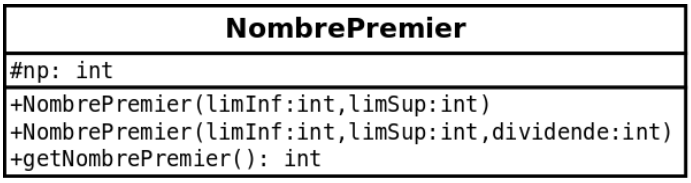


14 Décomposition en produit de facteurs_premiers

L'utilisateur entre un nombre entier au clavier. Le programme affiche sa décomposition en produit de facteurs premiers.

Remarques: Ce programme est l'extension de l'exercice « Nombre Premier ». Réaliser d'abord une classe `NombrePremier` représentant un nombre premier avec un constructeur `NombrePremier(int limInf, int limSup, int dividende)`. Ce constructeur génère le plus petit nombre premier entre `limInf` et `limSup` à condition qu'il est diviseur de `dividende`. Si aucun nombre premier vérifie ces conditions la valeur est 0 par convention.

```
DÉCOMPOSITION EN PRODUIT DE FACTEURS PREMIERS
Veuillez saisir un nombre entier p : 170
p = 1*2*5*17
DÉCOMPOSITION EN PRODUIT DE FACTEURS PREMIERS
Veuillez saisir un nombre entier p : 450
p = 1*2*3*3*5*5
```



Version 1.3.4b, Jens Getreu