Méthodes, fonctions sous Processing et Java

Valeur d'accueil et de reconversion en informatique (VARI1)
Daniel Porumbel (dp.cnam@gmail.com)
http://cedric.cnam.fr/~porumbed/vari1/

Rappel : méthodes `setup` et `draw`

Le code ci-dessous permet de tracer des lignes sans arrêt (point de départ : le centre de la toile)

```
1 void setup(){
2     println("Salut toto");
3     size(600,600);
4 }
5 void draw(){
6     line(300,300,random(600),random(600));
7     println("rebonjour toto");
8 }
```

On peut dire qu'une méthode permet de donner un nom à plusieurs lignes de code

1. les lignes 2-3 : un bloc de code appelé `setup()`
2. les lignes 6-7 : un bloc de code appelé `draw()`

Rappel : méthodes `setup` et `draw`

Le code ci-dessous permet de tracer des lignes sans arrêt (point de départ : le centre de la toile)

```
1 void setup(){
2     println("Salut␣toto");
3     size(600,600);
4 }
5 void draw(){
6     line(300,300,random(600),random(600));
7     println("rebonjour␣toto");
8 }
```

On peut dire qu'une méthode permet de donner un nom à plusieurs lignes de code

1. les lignes 2-3 : un bloc de code appelé `setup()`
2. les lignes 6-7 : un bloc de code appelé `draw()`

Fonctions

Une fonction comporte :

une entrée on fait « rentrer » des informations dans la fonction (on lui donne les données avec lesquelles travailler).

un traitement grâce aux informations reçues en entrée, la fonction exécute un traitement, elle fait quelque chose

une sortie à la fin la fonction renvoie un résultat via return.

méthode = une fonction qui ne renvoie rien (void)

- Une méthode n'a pas besoin de return
- void= pas de sortie

Fonctions

Une fonction comporte :

une entrée on fait « rentrer » des informations dans la fonction (on lui donne les données avec lesquelles travailler).

un traitement grâce aux informations reçues en entrée, la fonction exécute un traitement, elle fait quelque chose

une sortie à la fin la fonction renvoie un résultat via return.

méthode = une fonction qui ne renvoie rien (void)

- Une méthode n'a pas besoin de return
- void= pas de sortie

Exemple de définition de fonction :

```
int calculerCube(int x){
    return x*x*x;
}
```

entrée : une variable entière

sortie : une variable entière

Exemple de définition de fonction qui ne renvoie rien :

```
void direBonjour(String nomPersonne){
    println("Bonjour "+nomPersonne);
}
```

Exemples d'appels de fonctions

```
...
int n = calculerCube(5);        //n vaut 125
direBonjour("Toto"); //Affiche "Bonjour Toto"
...
```

Exemple de définition de fonction :

```
int calculerCube(int x){
    return x*x*x;
}
```

entrée : une variable entière

sortie : une variable entière

Exemple de définition de fonction qui ne renvoie rien :

```
void direBonjour(String nomPersonne){
    println("Bonjour "+nomPersonne);
}
```

Exemples d'appels de fonctions

```
...
int n = calculerCube(5);        //n vaut 125
direBonjour("Toto"); //Affiche "Bonjour Toto"
...
```

Exemple de définition de fonction :

```
int calculerCube(int x){
    return x*x*x;
}
```

entrée : une variable entière

sortie : une variable entière

Exemple de définition de fonction qui ne renvoie rien :

```
void direBonjour(String nomPersonne){
    println("Bonjour "+nomPersonne);
}
```

Exemples d'appels de fonctions

```
...
int n = calculerCube(5);        //n vaut 125
direBonjour("Toto"); //Affiche "Bonjour Toto"
...
```

Définition de fonctions avec plusieurs entrées :

entrées : une chaine de caractères et une lettre

```
void direSalut(String nom, char initialePrenom){
    println("Salut␣"+initialePrenom+"␣"+nom);
}
```

Moyenne de 3 entiers :

sortie : float pas int

```
float calculerMoyenne(int a, int b, int c){
    return (float)(a+b+c)/3;//conversion float
}
```

Appeler ces fonctions :

apostrophes pour char

```
...
direSalut("Toto", 'T');//=> "Salut T Toto"
direSalut("Titi", 'S');//=> "Salut S Titi"
float m = calculerMoyenne(7,7,8);//m=7.33
...
```

Définition de fonctions avec plusieurs entrées :

entrées : une chaine de caractères et une lettre

```
void direSalut(String nom, char initialePrenom){
    println("Salut␣"+initialePrenom+"␣"+nom);
}
```

Moyenne de 3 entiers :

sortie : float pas int

```
float calculerMoyenne(int a, int b, int c){
    return (float)(a+b+c)/3; //conversion float
}
```

Appeler ces fonctions :

apostrophes pour char

```
...
direSalut("Toto", 'T'); //=> "Salut T Toto"
direSalut("Titi", 'S'); //=> "Salut S Titi"
float m = calculerMoyenne(7,7,8); //m=7.33
...
```

Définition de fonctions avec plusieurs entrées :

entrées : une chaine de caractères et une lettre

```
void direSalut(String nom,char initialePrenom){
    println("Salut␣"+initialePrenom+"␣"+nom);
}
```

Moyenne de 3 entiers :

sortie : float pas int

```
float calculerMoyenne(int a, int b, int c){
    return (float)(a+b+c)/3;//conversion float
}
```

Appeler ces fonctions :

apostrophes pour char

```
...
direSalut("Toto", 'T');//=> "Salut T Toto"
direSalut("Titi", 'S');//=> "Salut S Titi"
float m = calculerMoyenne(7,7,8);//m=7.33
...
```

`Processing` exécute `setup()` en premier

La première fonction exécutée est :

`setup()` sous `Processing`

`main()` sous `C/C++/Java`

Elle peut appeler d'autres fonctions

- fonctions prédéfinies : `size(...)`, `ellipse(...)`, etc.
- fonction `calculerMax(...)` définie par nous

Processing exécute setup() en premier

La première fonction exécutée est :

setup() sous Processing

main() sous C/C++/Java

```
int calculerMin(int a, int b){
    if(a<b)
        return a;
    else
        return b;
}
void setup(){
    int x = (int)random(200),y=(int)random(300);
    int rayon = calculerMin(x,y);
    size(600,600);
    ellipse(x,y,2*rayon,2*rayon);
}           //2*rayon=diamètre=>toucher bord
```

La fonction `draw()` déjà vu

- appelée de manière répétitive par `Processing`
  - Rappel : `setup()` est appelée une seule fois au début
- le nombre d'appels par seconde peut être modifié par un appel `frameRate(nombre)`
- Pour faire une animation : tracer une ligne à des coordonnées aléatoires à l'intérieur du `draw()`

```
void setup(){
    size(600,600);
    frameRate(1000);
}
void draw(){
    //tracer une ligne du pixel x au pixel y
    //les coordonnées x et y: aléatoires <600
    line(random(600),random(600), //pixel x
        random(600),random(600));//pixel y
}
```

La fonction `draw()` déjà vu

- appelée de manière répétitive par `Processing`
  - Rappel : `setup()` est appelée une seule fois au début
- le nombre d'appels par seconde peut être modifié par un appel `frameRate(nombre)`
- Pour faire une animation : tracer une ligne à des coordonnées aléatoires à l'intérieur du `draw()`

```
void setup(){
    size(600,600);
    frameRate(1000);
}
void draw(){
    //tracer une ligne du pixel x au pixel y
    //les coordonnées x et y: aléatoires <600
    line(random(600),random(600), //pixel x
         random(600),random(600));//pixel y
}
```

Mode actif et mode statique déjà vu

`Processing` connait deux modes :

mode actif On déclare des fonctions et toutes les calculs/affichages sont écrit dans les (corps des) fonctions

mode statique On ne déclare pas de fonction et on écrit une séquence d'instruction, comme jusqu'à aujourd'hui

Impossible de mélanger les deux modes : comment exécuter le programme ci-après ?

```
...
println("Toto");
void setup(){
    println("Dans le corps de la fonct. setup");
}
...
```

Mode actif et mode statique déjà vu

`Processing` connait deux modes :

mode actif On déclare des fonctions et toutes les calculs/affichages sont écrit dans les (corps des) fonctions

mode statique On ne déclare pas de fonction et on écrit une séquence d'instruction, comme jusqu'à aujourd'hui

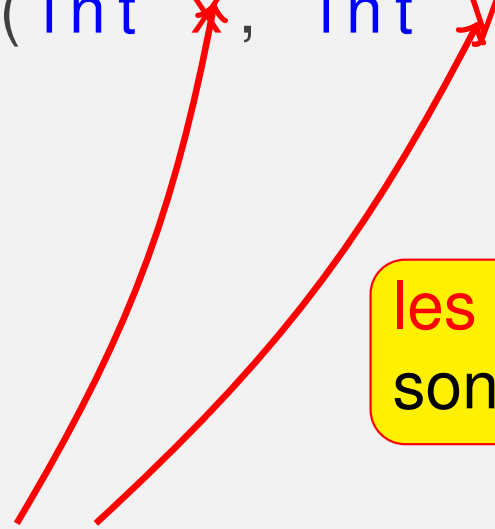
Impossible de mélanger les deux modes : comment exécuter le programme ci-après ?

```
...
println("Toto");
void setup(){
    println("Dans le corps de la fonct. setup");
}
...
```

Impossible de savoir quand afficher Toto : avant ou après l'appel à `setup()` ?

Tester le passage des paramètres par valeur

```
void testModif(int x, int y){
    x = 9;
    y = 12;
}
void setup(){
    int a=4;
    int b=5;
    testModif(a,b);
    println(a);
    println(b);
}
```

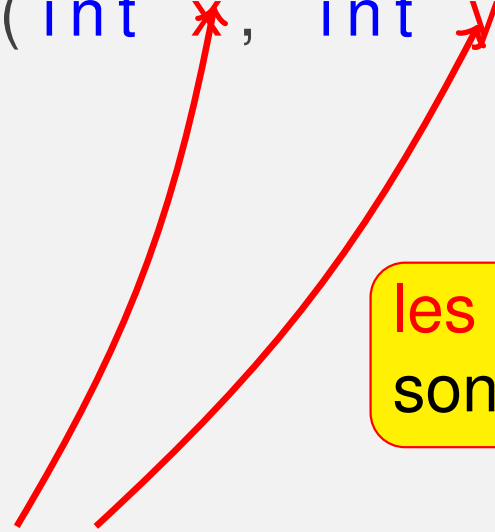


les valeurs de a et b
sont affectées à x et y

Qu'affiche ce programme ?

Tester le passage des paramètres par valeur

```
void testModif(int x, int y){
    x = 9;
    y = 12;
}
void setup(){
    int a=4;
    int b=5;
    testModif(a,b);
    println(a);
    println(b);
}
```



les valeurs de a et b
sont affectées à x et y

Qu'affiche ce programme ?
Exceptions à la régle plus haute du passage par valeur : les tableau et les objets (à voir plus tard).

Une fonction avec un tableau

```
int sommeTab(int[] tab){
    //tab: visible que dans sommeTab();
    return tab[0]+tab[1]+tab[2];
}
void setup(){
    //t: visible que dans setup()
    int[] t = new int[3];
    t[0] = 1;
    t[1] = 2;
    t[2] = 3;
    println(sommeTab(t));
}
```