DUT MMI – IUT de Marne-la-Vallée
19/12/2014
M1202 - Algorithmique

Cours 5
Fonctions, entrées-sorties

UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

iut Marne-la-Vallée

Philippe Gambette

Plan du cours 5 – Les fonctions, les entrées/sorties

- Résumé de l'épisode précédent
- Les entrées/sorties
- Écriture de fonctions en Java, variables locales
- Les fonctions, entrées et sortie
- Combiner plusieurs fonctions pour en créer une nouvelle

Plan du cours 5 – Les fonctions, les entrées/sorties

- Résumé de l'épisode précédent
- Les entrées/sorties
- Écriture de fonctions en Java, variables locales
- Les fonctions, entrées et sortie
- Combiner plusieurs fonctions pour en créer une nouvelle

Résumé de l'épisode précédent

Tableaux :

- pour stocker un ensemble de valeurs **de même type**

- **une** valeur par case

- **nombre de cases fixé** à l'initialisation du tableau

- boucle pour **parcourir le tableau**

Boucles :

- attention à l'**initialisation** et la **condition d'arrêt** (premier et dernier passage dans la boucle)

- boucle **Tant que** et boucle **Pour tout**

La boucle “for” / “Pour tout...”

Pour **parcourir tous les entiers entre deux valeurs entières**.

En pseudo-code :

Pour tout entier *i* de 1 à 42 faire :

    …

Fin Pour

En Java :

```
for(int i=1;i<43;i++){
    …
}
```

La boucle “for” / “Pour tout...”

Pour **parcourir tous les entiers entre deux valeurs entières**.

En pseudo-code :

Pour tout entier *i* de 1 à 42 faire :

    …

Fin Pour

En Java :

```
for(int i=1;i<43;i++){
    …
}
```

En Java avec while :

```
int i=1;
while(i<43){
…
i++;
}
```

La boucle “for” / “Pour tout...”

Pour **parcourir tous les entiers entre deux valeurs entières**.

En pseudo-code :

Pour tout entier *i* de 1 à 42 faire :

…

Fin Pour

En Java :

```
for(int i=1;i<43;i++){
    …
}
```

En Java avec while :

```
int i=1;
while(i<43){
…
i++;
}
```

La boucle “for” / “Pour tout...”

Exemple : **parcours des cases d'un tableau**

En pseudo-code avec Tant que :

Variables : tableau d'entiers *tab*, entier *i*
i ← 1
Tant que *i* < **Longueur**(*tab*)+1 faire :
    [des choses avec la *i*-ième case du tableau **Case**(*tab*,*i*)...]
    i ← *i*+1
Fin Tant que

En pseudo-code avec Pour :

Variables : tableau d'entiers *tab*, entier *i*
Pour *i* de 1 à **Longueur**(*tab*) faire :
    [des choses avec la *i*-ième case du tableau **Case**(*tab*,*i*)...]
Fin Pour

La boucle “for” / “Pour tout...”

La boucle “for” / “Pour tout”

Une boucle pour **parcourir tous les entiers entre deux valeurs entières**.

Algorithme **DessineHistogramme**
Entrée : tableau de chaînes de caractères *Mots* et tableau d'entiers *NbApparitions*.
Variable : entier *compteur*
Début
compteur ← 1
Tant que *compteur* < **Longueur**(*Mots*)+1 faire :
    dessineRectanglePlein(*compteur**4-4,
        50-10***Case**(*NbApparitions*,*compteur*),
        4,10***Case**(*NbApparitions*,*compteur*),
        couleurRGB(0,0,255))
    compteur ← 1 + *compteur*
Fin TantQue
Fin

Algorithme **DessineHistogramme**
Entrée : tableau de chaînes de caractères *Mots* et tableau d'entiers *NbApparitions*.
Variable : entier *compteur*
Début
Pour *compteur* de 1 à **Longueur**(*Mots*) faire :
    dessineRectanglePlein(*compteur**4-4,
        50-10***Case**(*NbApparitions*,*compteur*),
        4,10***Case**(*NbApparitions*,*compteur*),
        couleurRGB(0,0,255))
Fin Pour
Fin

La boucle “for” / “Pour tout...”

La boucle “for” / “Pour tout”

Une boucle pour **parcourir tous les entiers entre deux valeurs entières**.

Algorithme **DessineHistogramme**
Entrée : tableau de chaînes de caractères *Mots* et tableau d'entiers *NbApparitions*.
Variable : entier *compteur*
Début
compteur ← 1
Tant que *compteur* < **Longueur**(*Mots*)+1 faire :
    dessineRectanglePlein(*compteur**4-4,
        50-10***Case**(*NbApparitions*,*compteur*),
        4,10***Case**(*NbApparitions*,*compteur*),
        couleurRGB(0,0,255))
    compteur ← 1 + *compteur*
Fin TantQue
Fin

En Java :

```
int compteur;
compteur=1;
while(compteur<mots.length+1) {
    …
}
```

Algorithme **DessineHistogramme**
Entrée : tableau de chaînes de caractères *Mots* et tableau d'entiers *NbApparitions*.
Variable : entier *compteur*
Début
Pour *compteur* de 1 à **Longueur**(*Mots*) faire :
    dessineRectanglePlein(*compteur**4-4,
        50-10***Case**(*NbApparitions*,*compteur*),
        4,10***Case**(*NbApparitions*,*compteur*),
        couleurRGB(0,0,255))
Fin Pour
Fin

```
for(int compteur=1;compteur<mots.length+1;compteur++){
    …
}
```

La boucle “for” / “Pour tout…”

La boucle “for” / “Pour tout”

Une boucle pour **parcourir tous les entiers entre deux valeurs entières**.

Algorithme **DessineHistogramme**
Entrée : tableau de chaînes de caractères *Mots* et tableau d'entiers *NbApparitions*.
Variable : entier *compteur*
Début
compteur ← 1
Tant que *compteur* < **Longueur**(*Mots*)+1 faire :
    dessineRectanglePlein(*compteur**4-4,
        50-10***Case**(*NbApparitions*,*compteur*),
        4,10***Case**(*NbApparitions*,*compteur*),
        couleurRGB(0,0,255))
    compteur ← 1 + *compteur*
Fin TantQue
Fin

En Java :

```
int compteur;
compteur=1;
while(compteur<mots.length+1) {
    …
}
```

Algorithme **DessineHistogramme**
Entrée : tableau de chaînes de caractères *Mots* et tableau d'entiers *NbApparitions*.
Variable : entier *compteur*
Début
Pour *compteur* de 1 à **Longueur**(*Mots*) faire :
    dessineRectanglePlein(*compteur**4-4,
        50-10***Case**(*NbApparitions*,*compteur*),
        4,10***Case**(*NbApparitions*,*compteur*),
        couleurRGB(0,0,255))
Fin Pour
Fin

déclaration + initialisation

```
for(int compteur=1;compteur<mots.length+1;compteur++){
    …
}
```

La boucle “for” / “Pour tout...”

La boucle “for” / “Pour tout”

Une boucle pour **parcourir tous les entiers entre deux valeurs entières**.

Algorithme **DessineHistogramme**
Entrée : tableau de chaînes de caractères *Mots* et tableau d'entiers *NbApparitions*.
Variable : entier *compteur*
Début
compteur ← 1
Tant que *compteur* < **Longueur**(*Mots*)+1 faire :
    dessineRectanglePlein(*compteur**4-4,
        50-10***Case**(*NbApparitions*,*compteur*),
        4,10***Case**(*NbApparitions*,*compteur*),
        couleurRGB(0,0,255))
    compteur ← 1 + *compteur*
Fin TantQue
Fin

En Java :

```
int compteur;
compteur=1;
while(compteur<mots.length+1) {
    …
}
```

Algorithme **DessineHistogramme**
Entrée : tableau de chaînes de caractères *Mots* et tableau d'entiers *NbApparitions*.
Variable : entier *compteur*
Début
Pour *compteur* de 1 à **Longueur**(*Mots*) faire :
    dessineRectanglePlein(*compteur**4-4,
        50-10***Case**(*NbApparitions*,*compteur*),
        4,10***Case**(*NbApparitions*,*compteur*),
        couleurRGB(0,0,255))
Fin Pour
Fin

déclaration + initialisation *condition d'arrêt*

```
for(int compteur=1;compteur<mots.length+1;compteur++){
    …
}
```

La boucle “for” / “Pour tout...”

La boucle “for” / “Pour tout”

Une boucle pour **parcourir tous les entiers entre deux valeurs entières**.

Algorithme **DessineHistogramme**
Entrée : tableau de chaînes de caractères *Mots* et tableau d'entiers *NbApparitions*.
Variable : entier *compteur*
Début
compteur ← 1
Tant que *compteur* < **Longueur**(*Mots*)+1 faire :
    dessineRectanglePlein(*compteur**4-4,
        50-10***Case**(*NbApparitions*,*compteur*),
        4,10***Case**(*NbApparitions*,*compteur*),
        couleurRGB(0,0,255))
    compteur ← 1 + *compteur*
Fin TantQue
Fin

En Java :

```
int compteur;
compteur=1;
while(compteur<mots.length+1) {
    …
}
```

Algorithme **DessineHistogramme**
Entrée : tableau de chaînes de caractères *Mots* et tableau d'entiers *NbApparitions*.
Variable : entier *compteur*
Début
Pour *compteur* de 1 à **Longueur**(*Mots*) faire :
    dessineRectanglePlein(*compteur**4-4,
        50-10***Case**(*NbApparitions*,*compteur*),
        4,10***Case**(*NbApparitions*,*compteur*),
        couleurRGB(0,0,255))
Fin Pour
Fin

déclaration + initialisation — *condition d'arrêt* — *mise à jour*

```
for(int compteur=1;compteur<mots.length+1;compteur++){
    …
}
```

La boucle “for” / “Pour tout...”

La boucle “for” / “Pour tout”

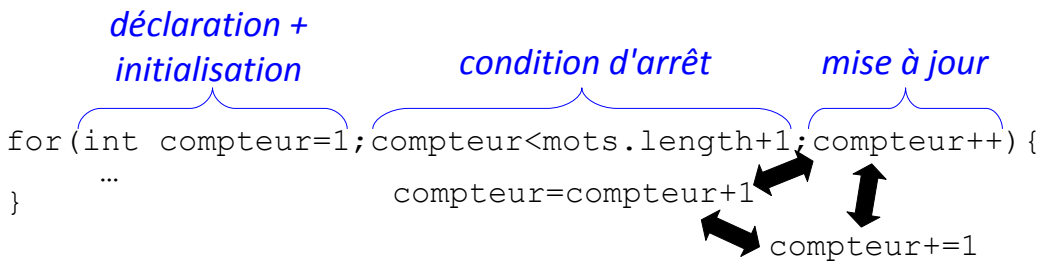
Une boucle pour **parcourir tous les entiers entre deux valeurs entières**.

Algorithme **DessineHistogramme**
Entrée : tableau de chaînes de caractères *Mots* et tableau d'entiers *NbApparitions*.
Variable : entier *compteur*
Début
compteur ← 1
Tant que *compteur* < **Longueur**(*Mots*)+1 faire :
    dessineRectanglePlein(*compteur**4-4,
        50-10***Case**(*NbApparitions*,*compteur*),
        4,10***Case**(*NbApparitions*,*compteur*),
        couleurRGB(0,0,255))
    compteur ← 1 + *compteur*
Fin TantQue
Fin

En Java :

```
int compteur;
compteur=1;
while(compteur<mots.length+1){
    …
}
```

Algorithme **DessineHistogramme**
Entrée : tableau de chaînes de caractères *Mots* et tableau d'entiers *NbApparitions*.
Variable : entier *compteur*
Début
Pour *compteur* de 1 à **Longueur**(*Mots*) faire :
    dessineRectanglePlein(*compteur**4-4,
        50-10***Case**(*NbApparitions*,*compteur*),
        4,10***Case**(*NbApparitions*,*compteur*),
        couleurRGB(0,0,255))
Fin Pour
Fin



Plan du cours 5 – Les fonctions, les entrées/sorties

- Résumé de l'épisode précédent
- Les entrées/sorties
- Écriture de fonctions en Java, variables locales
- Les fonctions, entrées et sortie
- Combiner plusieurs fonctions pour en créer une nouvelle

Les entrées-sorties

Entrées-sorties dans la **communication ordinateur – utilisateur** :

- périphériques de saisie d'entrées : clavier, souris, joystick, webcam, Wii remote, Kinect...

- périphérique d'affichage des sorties : écran, vidéo-projecteur, imprimante...

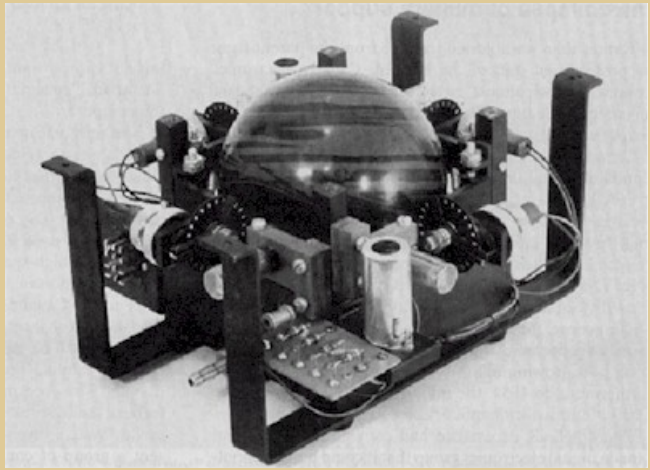
Différent des **variables d'entrée** / **variable de sortie** dans la “**communication entre algorithmes”**.

La souris

La “minute culturelle”

L'invention de la souris

1952 Trackball (boule de commande)
Tom Cranston et Fred Longstaff
(Marine Royale Canadienne)



1963 Souris mécanique
Douglas Engelbart et Bill English
(Stanford Research Institute)

1977 Souris optique
Jean-Daniel Nicoud et André Guignard
(Ecole polytechnique fédérale de Lausanne)

Photo : http://en.wikipedia.org/wiki/File:DATAR_trackball.jpg

La souris

La “minute culturelle”

L'invention de la souris

1952 Trackball (boule de commande)
Tom Cranston et Fred Longstaff
(Marine Royale Canadienne)

1963 Souris mécanique
Douglas Engelbart et Bill English
(Stanford Research Institute)

1977 Souris optique
Jean-Daniel Nicoud et André Guignard
(Ecole polytechnique fédérale de Lausanne)

Photo : http://cerncourier.com/cws/article/cern/28358/1/cernbooks2_12-00
Plus d'infos sur http://fr.wikipedia.org/wiki/Souris_(informatique)

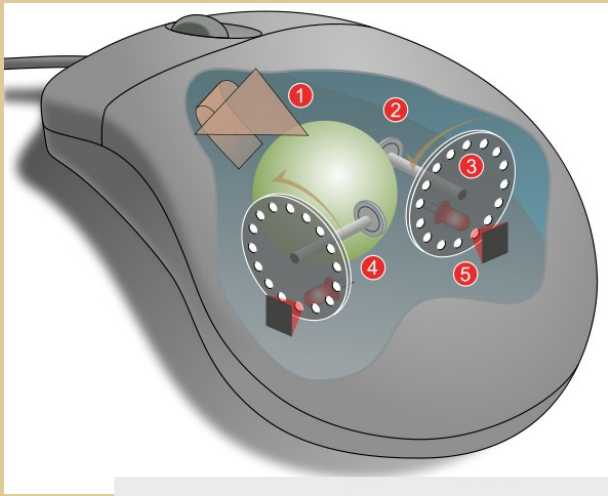
La souris

La “minute culturelle”

L'invention de la souris

1952 Trackball (boule de commande)
Tom Cranston et Fred Longstaff
(Marine Royale Canadienne)

1963 Souris mécanique
Douglas Engelbart et Bill English
(Stanford Research Institute)



1977 Souris optique
Jean-Daniel Nicoud et André Guignard
(Ecole polytechnique fédérale de Lausanne)

Image : Jeremykemp, Pbroks13, http://fr.wikipedia.org/wiki/Fichier:Mouse_mechanism_diagram.svg
Photo Stéphane Magnenat (User:Nct) http://en.wikipedia.org/wiki/File:SmakyMouseAG.jpeg

Autres périphériques d'entrée

La “minute culturelle”

Données transmises par la WiiRemote :

image de **profondeurs**

http://www.youtube.com/watch?v=0awjPUkBXOU

Autres périphériques d'entrée

La “minute culturelle”

Données transmises par la Kinect :

image de **couleurs** + **profondeurs**

http://www.youtube.com/watch?v=v0G0Cb7Zqjk

Les entrées-sorties

Entrées-sorties dans la **communication ordinateur – utilisateur**

Quel **type de données** utiliser en **algorithmique** pour coder les entrées-sorties ?

Périphérique	Type de données transmises
Clavier Souris à 1 bouton Webcam Kinect	
Ecran	

Les entrées-sorties

Entrées-sorties dans la **communication ordinateur – utilisateur**

Quel **type de données** utiliser en **algorithmique** pour coder les entrées-sorties ?

Périphérique	Type de données transmises
Clavier Souris à 1 bouton Webcam Kinect	chaîne de caractères
Ecran	

Les entrées-sorties

Entrées-sorties dans la **communication ordinateur – utilisateur**

Quel **type de données** utiliser en **algorithmique** pour coder les entrées-sorties ?

Périphérique	Type de données transmises
Clavier	chaîne de caractères
Souris à 1 bouton	deux entiers (abscisse et ordonnée) + un booléen (clic ou pas)
Webcam	
Kinect	
Ecran	

Les entrées-sorties

Entrées-sorties dans la **communication ordinateur – utilisateur**

Quel **type de données** utiliser en **algorithmique** pour coder les entrées-sorties ?

Périphérique	Type de données transmises
Clavier	chaîne de caractères
Souris à 1 bouton	deux entiers (abscisse et ordonnée) + un booléen (clic ou pas)
Webcam	image, donc tableau de tableaux de couleurs RGB
Kinect	
Ecran	

Les entrées-sorties

Entrées-sorties dans la **communication ordinateur – utilisateur**

Quel **type de données** utiliser en **algorithmique** pour coder les entrées-sorties ?

Périphérique	Type de données transmises
Clavier	chaîne de caractères
Souris à 1 bouton	deux entiers (abscisse et ordonnée) + un booléen (clic ou pas)
Webcam	image, donc tableau de tableaux de couleurs RGB
Kinect	image + tableau de tableaux d'entiers (profondeur)
Ecran	Si ligne de commande :

Les entrées-sorties

Entrées-sorties dans la **communication ordinateur – utilisateur**

Quel **type de données** utiliser en **algorithmique** pour coder les entrées-sorties ?

Périphérique	Type de données transmises
Clavier	chaîne de caractères
Souris à 1 bouton	deux entiers (abscisse et ordonnée) + un booléen (clic ou pas)
Webcam	image, donc tableau de tableaux de couleurs RGB
Kinect	image + tableau de tableaux d'entiers (profondeur)
Ecran	Si ligne de commande : chaîne de caractères Si interface graphique :

Les entrées-sorties

Entrées-sorties dans la **communication ordinateur – utilisateur**

Quel **type de données** utiliser en **algorithmique** pour coder les entrées-sorties ?

Périphérique	Type de données transmises
Clavier	chaîne de caractères
Souris à 1 bouton	deux entiers (abscisse et ordonnée) + un booléen (clic ou pas)
Webcam	image, donc tableau de tableaux de couleurs RGB
Kinect	image + tableau de tableaux d'entiers (profondeur)
Ecran	Si ligne de commande : chaîne de caractères Si interface graphique : image, donc tableau de tableaux de couleurs RGB

Les entrées-sorties

en pseudo-code

Entrées-sorties dans la **communication ordinateur – utilisateur**

Quel **type de données** utiliser en **algorithmique** pour coder les entrées-sorties ?

Périphérique	Type de données transmises
Clavier	chaîne de caractères chaîne de caractères **reponseALaQuestion**(*questionAAfficher*) affiche la question *questionAAfficher* et renvoie une chaîne de caractères.
Ecran	Si ligne de commande : chaîne de caractères

Les entrées-sorties

en pseudo-code

Entrées-sorties dans la **communication ordinateur – utilisateur**

Quel **type de données** utiliser en **algorithmique** pour coder les entrées-sorties ?

Périphérique	Type de données transmises
Clavier	chaîne de caractères chaîne de caractères **reponseALaQuestion**(*questionAAfficher*) affiche la question *questionAAfficher* et renvoie une chaîne de caractères. Exemple : **reponseALaQuestion**(“Quel est votre nom”) me laisse taper mon nom au clavier et renvoie “Gambette”
Ecran	Si ligne de commande : chaîne de caractères chaîne de caractères **Affiche**(*chaineAAfficher*) affiche la chaîne de caractères *chaineAAfficher* et ne renvoie rien.

Les entrées-sorties

en Java

Entrées-sorties dans la **communication ordinateur – utilisateur**

Quel **type de données** utiliser en **algorithmique** pour coder les entrées-sorties ?

Périphérique	Type de données transmises

Clavier	chaîne de caractères
	
	`Scanner lectureClavier = new Scanner(System.in);`
	`String chaineLue = lectureClavier.next();`
	LectureClavier.next() renvoie une chaîne de caractères.

Ecran	Si ligne de commande : chaîne de caractères

Les entrées-sorties

en Java

Entrées-sorties dans la **communication ordinateur – utilisateur**

Quel **type de données** utiliser en **algorithmique** pour coder les entrées-sorties ?

Périphérique	Type de données transmises

Clavier	chaîne de caractères
	
	`Scanner lectureClavier = new Scanner(System.in);`
	`String chaineLue = lectureClavier.next();`
	LectureClavier.next() renvoie une chaîne de caractères.
	
	`int entierLu = lectureClavier.nextInt();`
	LectureClavier.nextInt() renvoie un entier.

Ecran	Si ligne de commande : chaîne de caractères

Les entrées-sorties

en Java

Entrées-sorties dans la **communication ordinateur – utilisateur**

Quel **type de données** utiliser en **algorithmique** pour coder les entrées-sorties ?

Périphérique	Type de données transmises

Clavier	chaîne de caractères
	`Scanner lectureClavier = new Scanner(System.in);`
	`String chaineLue = lectureClavier.next();`
	LectureClavier.next() renvoie une chaîne de caractères.
	`int entierLu = lectureClavier.nextInt();`
	LectureClavier.nextInt() renvoie un entier.

Ecran	Si ligne de commande :
	chaîne de caractères
	`String chaineAAfficher="blabla";`
	`System.out.println(chaineAAfficher);`
	affiche la chaîne de caractères *questionAAfficher*, puis retourne à la ligne, mais ne renvoie rien.

Les entrées-sorties

en Java

Entrées-sorties dans la **communication ordinateur – utilisateur**

Quel **type de données** utiliser en **algorithmique** pour coder les entrées-sorties ?

Périphérique	Type de données transmises

Clavier	chaîne de caractères
	`Scanner lectureClavier = new Scanner(System.in);`
	`String chaineLue = lectureClavier.next();`
	LectureClavier.next() renvoie une chaîne de caractères.
	`int entierLu = lectureClavier.nextInt();`
	LectureClavier.nextInt() renvoie un entier.

Ecran	Si ligne de commande :
	chaîne de caractères
	`String chaineAAfficher="blabla";`
	`System.out.print(chaineAAfficher);`
	affiche la chaîne de caractères *questionAAfficher*, et ne renvoie rien.

Plan du cours 5 – Les fonctions, les entrées/sorties

- Résumé de l'épisode précédent
- Les entrées/sorties
- Écriture de fonctions en Java, variables locales
- Les fonctions, entrées et sortie
- Combiner plusieurs fonctions pour en créer une nouvelle

Les fonctions

Les **fonctions** = les **algorithmes**

Un ou plusieurs paramètres en entrée...

Une sortie...

Les fonctions

Les **fonctions** = les **algorithmes**

Un ou plusieurs paramètres en entrée...
ou aucun

Une sortie...
ou aucune

Les fonctions en Java – La fonction `main`

Toujours une fonction `main` qui ne renvoie rien

Déclaration des fonctions après la fonction `main`

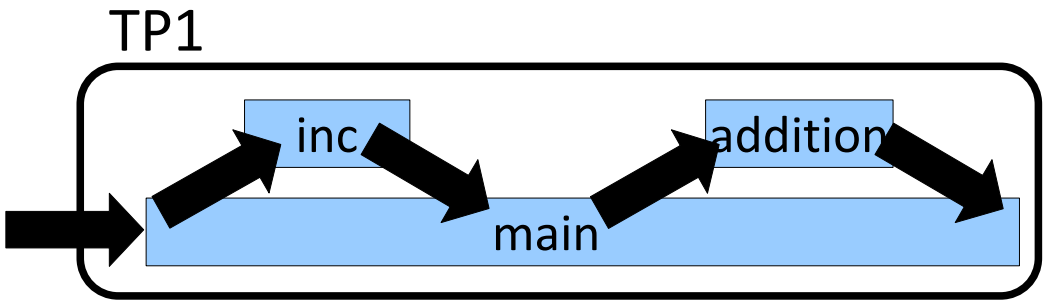
Les fonctions en Java – La fonction `main`

Algorithme pour **construire une maison** ?

→ faire appel à un **maître d'oeuvre**

Que fait le maître d'oeuvre ?
→ il réunit les informations sur la maison qui va être construite
→ il trouve des artisans
→ à chaque artisan, il donne des informations sur ce qu'il attend, et récupère le résultat
→ chaque artisan peut lui-même sous-traiter une partie de son travail à un autre artisan : il donne les informations sur ce qu'il attend, et récupère le résultat
→ la maison se construit

La construction de la maison : le programme TP1
Le maître d'oeuvre : la fonction main
L'artisan 1 : la fonction inc
L'artisan 2 : la fonction addition




```
public class TP1{
   public static void main(String[] arg){
      int i,j;
      i=5;
      j=34;
      System.out.print("i+1="+inc(i)+",i="+i+",
         j="+j+", somme : "+addition(i,j));
   }
   public static int inc(int i){
      i=i+1;
      return i;
   }
   public static int addition(int i, int j){
      return i+j;
   }
}
```

Les fonctions en Java – La fonction `main`

Toujours une fonction `main` qui ne renvoie rien
et qui prend en entrée les paramètres du programme

Déclaration des fonctions après la fonction `main`

```
public class TP1{
   public static void main(String[] arg){
      int i,j;
      i=5;
      j=34;
      System.out.print("i+1="+inc(i)+",i="+i+",
      j="+j+", somme : "+addition(i,j));
   }
   public static int inc(int i){
      i=i+1;
      return i;
   }
   public static int addition(int i, int j){
      return i+j;
   }
}
```

Les fonctions en Java – La fonction `main`

Toujours une fonction `main` qui ne renvoie rien
et qui prend en entrée les **paramètres du programme**

Déclaration des fonctions après la fonction `main`

```
public class TP1{
    public static void main(String[] arg){
        int i,j;
        i=5;
        j=34;
        System.out.print("i+1
        j="+j+", somme : "+ad
    }
    public static int inc(in
        i=i+1;
        return i;
    }
    public static int addition(int i, int j){
        return i+j;
    }
}
```

En ligne de commande :
`java TP1 toto 1 10 10.5`

Le tableau arg est alors :
`{"toto","1","10","10.5"}`

Les fonctions en Java – La fonction `main`

Toujours une fonction `main` **qui ne renvoie rien**
et qui prend en entrée les paramètres du programme

Déclaration des fonctions après la fonction `main`

```
public class TP1{
    public static void main(String[] arg){
        int i,j;
        i=5;
        j=34;
        System.out.print("i+1="+inc(i)+",i="+i+",
        j="+j+", somme : "+addition(i,j));
    }
    public static int inc(int i){
        i=i+1;
        return i;
    }
    public static int addition(int i, int j){
        return i+j;
    }
}
```

Les fonctions en Java – Autres fonctions

Toujours une fonction `main` **qui ne renvoie rien**
et qui prend en entrée les paramètres du programme

Déclaration des fonctions après la fonction `main`

```
public class TP1{
    public static void main(String[] arg){
        int i,j;
        i=5;
        j=34;
        System.out.print("i+1="+inc(i)+",i="+i+",
        j="+j+", somme : "+addition(i,j));
    }
    public static int inc(int i){
        i=i+1;
        return i;
    }
    public static int addition(int i, int j){
        return i+j;
    }
}
```

type de variable renvoyée par la fonction

Les fonctions en Java – Autres fonctions

Toujours une fonction `main` **qui ne renvoie rien**
et qui prend en entrée les paramètres du programme

Déclaration des fonctions après la fonction `main`

```
public class TP1{
    public static void main(String[] arg){
        int i,j;
        i=5;
        j=34;
        System.out.print("i+1="+inc(i)+",i="+i+",
        j="+j+", somme : "+addition(i,j));
    }
    public static int inc(int i){
        i=i+1;
        return i;
    }
    public static int addition(int i, int j){
        return i+j;
    }
}
```

type de variable renvoyée par la fonction

entrées de la fonction précédées de leur type

Les fonctions en Java – Autres fonctions

Toujours une fonction `main` **qui ne renvoie rien**
et qui prend en entrée les paramètres du programme

Déclaration des fonctions après la fonction `main`

```
public class TP1{
    public static void main(String[] arg){
        int i,j;
        i=5;
        j=34;
        System.out.print("i+1="+inc(i)+",i="+i+",
        j="+j+", somme : "+addition(i,j));
    }
    public static int inc(int i){
        i=i+1;
        return i;
    }
    public static int addition(int i, int j){
        return i+j;
    }
}
```

appel de la fonction addition

Les fonctions en Java – Autres fonctions

Seule **la fonction `main`** s'exécute quand on exécute le programme TP1.

Les **autres fonctions** ne s'exécutent que si elles sont **appelées** pendant l'exécution de la fonction `main`

```
public class TP1{
    public static void main(String[] arg){
        int i,j;
        i=5;
        j=34;
        System.out.print("i+1="+inc(i)+",i="+i+",
        j="+j+", somme : "+addition(i,j));
    }
    public static int inc(int i){
        i=i+1;
        return i;
    }
    public static int addition(int i, int j){
        return i+j;
    }
}
```

appel de la fonction inc

appel de la fonction addition

Les fonctions en Java

Visibilité des variables : toute variable déclarée à l'intérieur d'une fonction n'est valable **que dans cette fonction** et **ne peut pas être utilisée ailleurs**.

Variables **locales**

```
public class TP1{
    public static void main(String[] arg){
        int i,j;
        i=5;
        j=34;
        System.out.print("i+1="+inc(i)+",i="+i+",
        j="+j+", somme : "+addition(i,j));
    }
    public static int inc(int i){
        i=i+1;
        return i;
    }
    public static int addition(int i, int j){
        return i+j;
    }
}
```

Les fonctions en Java

Visibilité des variables : toute variable déclarée à l'intérieur d'une fonction n'est valable **que dans cette fonction** et **ne peut pas être utilisée ailleurs**.

Variables **locales**

```
public class TP1{
    public static void main(String[] arg){
        int i,j;
        i=5;
        j=34;
        System.out.print("i+1="+inc(i)+",i="+i+",
        j="+j+", somme : "+addition(i,j));
    }
    public static int inc(int i){
        i=i+1;
        return i;
    }
    public static int addition(int i, int j){
        return i+j;
    }
}
```

pas la même variable *i* même si elles ont la même valeur !

Les fonctions en Java

Visibilité des variables : toute variable déclarée à l'intérieur d'une fonction n'est valable **que dans cette fonction** et **ne peut pas être utilisée ailleurs**.

Variables **locales**

```
public class TP1{
    public static void main(String[] arg){
        int i,j;
        i=5;
        j=34;
        System.out.print("i+1="+inc(i)+",i="+i+",
        j="+j+", somme : "+addition(i,j));
    }
    public static int inc(int i){
        i=i+1;
        return i;
    }
    public static int addition(int i, int j){
        return i+j;
    }
}
```

définition de la fonction addition

Les fonctions en Java

Visibilité des variables : toute variable déclarée à l'intérieur d'une fonction n'est valable **que dans cette fonction** et **ne peut pas être utilisée ailleurs**.

Variables **locales**

```
public class TP1{
    public static void main(String[] arg){
        int i,j;
        i=5;
        j=34;
        System.out.print("i+1="+inc(i)+",i="+i+",
        j="+j+", somme : "+addition(i,j));
    }
    public static int inc(int i){
        i=i+1;
        return i;
    }
    public static int addition(int i, int j){
        return i+j;
    }
}
```

déclaration de la fonction addition

définition de la fonction addition

Les fonctions en Java

Visibilité des variables : toute variable déclarée à l'intérieur d'une fonction n'est valable **que dans cette fonction** et **ne peut pas être utilisée ailleurs**.

Variables **locales**

```
public class TP1{
    public static void main(String[] arg){
        int i,j;
        i=5;
        j=34;
        System.out.print("i+1="+inc(i)+",i="+i+",
        j="+j+", somme : "+addition(i,j));
    }
    public static int inc(int i){
        i=i+1;
        return i;
    }
    public static int addition(int i, int j){
        return i+j;
    }
}
```

déclaration de la fonction addition

renvoi du résultat en sortie

définition de la fonction addition

Les fonctions en Java

Visibilité des variables : toute variable déclarée à l'intérieur d'une fonction n'est valable **que dans cette fonction** et **ne peut pas être utilisée ailleurs**.

Variables **locales**

```
public class TP1{
    public static void main(String[] arg){
        int i,j;
        i=5;
        j=34;
        System.out.print("i+1="+inc(i)+",i="+i+",
        j="+j+", somme : "+addition(i,j));
    }
    public static int inc(int i){
        i=i+1;
        return i;
    }
    public static int addition(int i, int j){
        return i+j;
    }
}
```

appel de la fonction addition

déclaration de la fonction addition

renvoi du résultat en sortie

définition de la fonction addition

Plan du cours 5 – Les fonctions, les entrées/sorties

- Résumé de l'épisode précédent
- Les entrées/sorties
- Écriture de fonctions en Java, variables locales
- Les fonctions, entrées et sortie
- Combiner plusieurs fonctions pour en créer une nouvelle

Les fonctions

La “minute mathématique”

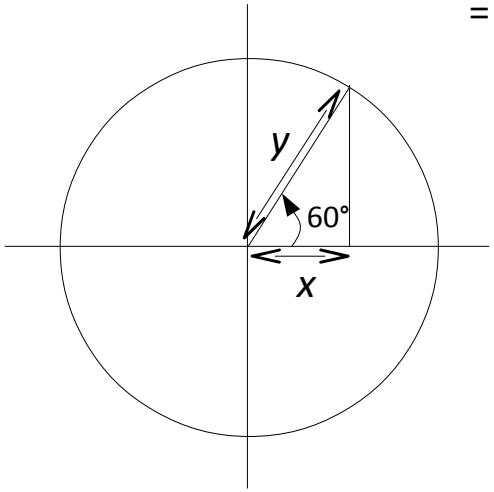
fonction	exemple	entrées possibles	sortie
cosinus	cosinus(1.047)=0.5		
somme			
opposé			
inverse			
différence			
estPositif			
partieEntière			
moyenne			
min			

Les fonctions

La “minute mathématique”

fonction	exemple	entrées possibles	sortie
cosinus	cosinus(1.047)=0.5		
somme			
opposé			
inverse			
différence			
estPositif			
partieEntière			
moyenne			
min			

cosinus(1.047) ≈ cosinus(π/3) = cosinus(60°)
= x/y
= 0.5/1



Les fonctions

La “minute mathématique”

fonction	exemple	entrées possibles	sortie
cosinus	cosinus(1.047)=0.5	flottant	flottant
somme			
opposé			
inverse			
différence			
estPositif			
partieEntière			
moyenne			
min			

Les fonctions

La “minute mathématique”

fonction	exemple	entrées possibles	sortie
cosinus	cosinus(1.047)=0.5	flottant	flottant
somme	somme(2,3)=5	2 entiers	entier
opposé			
inverse			
différence			
estPositif			
partieEntière			
moyenne			
min			

Les fonctions

La “minute mathématique”

fonction	exemple	entrées possibles	sortie
cosinus	cosinus(1.047)=0.5	flottant	flottant
somme	somme(2,3)=5	2 entiers	entier
opposé			
inverse			
différence			
estPositif			
partieEntière			
moyenne			
min			

Les “nombres à virgule” se notent avec un point aux Etats-Unis

La virgule sépare les paramètres d'une fonction

Les fonctions

La “minute mathématique”

fonction	exemple	entrées possibles	sortie
cosinus	cosinus(1.047)=0.5	flottant	flottant
somme	somme(2,3)=5	2 entiers 2 flottants	entier flottant
opposé			
inverse			
différence			
estPositif			
partieEntière			
moyenne			
min			

Les fonctions

La “minute mathématique”

fonction	exemple	entrées possibles	sortie
cosinus	cosinus(1.047)=0.5	flottant	flottant
somme	somme(2,3)=5	2 entiers 2 flottants	entier flottant
opposé	opposé(4)=-4	entier flottant	entier flottant
inverse			
différence			
estPositif			
partieEntière			
moyenne			
min			

Les fonctions

La “minute mathématique”

fonction	exemple	entrées possibles	sortie
cosinus	cosinus(1.047)=0.5	flottant	flottant
somme	somme(2,3)=5	2 entiers 2 flottants	entier flottant
opposé	opposé(4)=-4	entier flottant	entier flottant
inverse	inverse(10)=0.1	flottant	flottant
différence			
estPositif			
partieEntière			
moyenne			
min			

Les fonctions

La “minute mathématique”

fonction	exemple	entrées possibles	sortie
cosinus	cosinus(1.047)=0.5	flottant	flottant
somme	somme(2,3)=5	2 entiers 2 flottants	entier flottant
opposé	opposé(4)=-4	entier flottant	entier flottant
inverse	inverse(10)=0.1	flottant	flottant
différence	différence(2,3)=-1	2 entiers 2 flottants	entier flottant
estPositif			
partieEntière			
moyenne			
min			

Les fonctions

La “minute mathématique”

fonction	exemple	entrées possibles	sortie
cosinus	cosinus(1.047)=0.5	flottant	flottant
somme	somme(2,3)=5	2 entiers 2 flottants	entier flottant
opposé	opposé(4)=-4	entier flottant	entier flottant
inverse	inverse(10)=0.1	flottant	flottant
différence	différence(2,3)=-1	2 entiers 2 flottants	entier flottant
estPositif	estPositif(-5)=FAUX	entier	booléen
partieEntière			
moyenne			
min			

Les fonctions

La "minute mathématique"

fonction	exemple	entrées possibles	sortie
cosinus	cosinus(1.047)=0.5	flottant	flottant
somme	somme(2,3)=5	2 entiers 2 flottants	entier flottant
opposé	opposé(4)=-4	entier flottant	entier flottant
inverse	inverse(10)=0.1	flottant	flottant
différence	différence(2,3)=-1	2 entiers 2 flottants	entier flottant
estPositif	estPositif(-5)=FAUX	entier	booléen
partieEntière	partieEntière(5.6)=5	flottant	entier
moyenne			
min			

PartieEntière(10)=$\lfloor 10 \rfloor$=10

PartieEntière(5.6)=$\lfloor 5.2 \rfloor$=5

Les fonctions

La “minute mathématique”

fonction	exemple	entrées possibles	sortie
cosinus	cosinus(1.047)=0.5	flottant	flottant
somme	somme(2,3)=5	2 entiers 2 flottants	entier flottant
opposé	opposé(4)=-4	entier flottant	entier flottant
inverse	inverse(10)=0.1	flottant	flottant
différence	différence(2,3)=-1	2 entiers 2 flottants	entier flottant
estPositif	estPositif(-5)=FAUX	entier	booléen
partieEntière	partieEntière(5.6)=5	flottant	entier
moyenne	moyenne(2,4,6)=4	3 flottants	flottant
min			

Les fonctions

La “minute mathématique”

fonction	exemple	entrées possibles	sortie
cosinus	cosinus(1.047)=0.5	flottant	flottant
somme	somme(2,3)=5	2 entiers 2 flottants	entier flottant
opposé	opposé(4)=-4	entier flottant	entier flottant
inverse	inverse(10)=0.1	flottant	flottant
différence	différence(2,3)=-1	2 entiers 2 flottants	entier flottant
estPositif	estPositif(-5)=FAUX	entier	booléen
partieEntière	partieEntière(5.6)=5	flottant	entier
moyenne	moyenne(2,4,6)=4	3 flottants	flottant
min	min({6,2,4,3})=2	tableau d'entiers tableau de flottants	entier flottant

Plan du cours 5 – Les fonctions, les entrées/sorties

- Résumé de l'épisode précédent
- Les entrées/sorties
- Écriture de fonctions en Java, variables locales
- Les fonctions, entrées et sortie
- Combiner plusieurs fonctions pour en créer une nouvelle

Les fonctions

La “minute mathématique”

fonction	exemple	entrées possibles	sortie
somme	somme(2,3)=5	flottant entier	flottant entier
opposé	opposé(4)=-4	flottant	flottant
différence	différence(2,3)=-1	2 entiers 2 flottants	entier flottant

Fonction **différence**
Entrées : 2 entiers *a* et *b*
Sortie : entier
Début
    Renvoyer ...
Fin

Les fonctions

La "minute mathématique"

fonction	exemple	entrées possibles	sortie
somme	somme(2,3)=5	flottant entier	flottant entier
opposé	opposé(4)=-4	flottant	flottant
différence	différence(2,3)=-1	2 entiers 2 flottants	entier flottant

Fonction **différence**
Entrées : 2 entiers *a* et *b*
Sortie : entier
Début
    Renvoyer **somme**(*a*,**opposé**(*b*))
Fin

Attention, si on utilise la fonction somme qui renvoie un flottant, le type de sortie n'est pas correct pour la fonction différence !

Les fonctions

La "minute mathématique"

fonction	exemple	entrées possibles	sortie
somme	somme(2,3)=5	flottant entier	flottant entier
opposé	opposé(4)=-4	flottant	flottant
différence	différence(2,3)=-1	2 entiers 2 flottants	entier flottant

Fonction **différence**
Entrées : 2 flottants *a* et *b*
Sortie : flottant
Début
    Renvoyer **somme**(*a*,**opposé**(*b*))
Fin