Initiation à la programmation Java (2)
IP2

Yan Jurski

Université de Paris

Organisation de ce cours d'IP2

- Amphi 2h
- TD 2h
- TP 2h

Ce n'est plus un cours/TD comme IP1 l'était au premier semestre !

Page moodle *IF12Y010 Initiation à la Programmation 2*

- TDs à partir de demain
- les TPs commenceront la semaine prochaine
  - Notez qu'en TP 3 salles sont réservées pour 2 groupes
  - 2 enseignants se partageront les 3 salles.
    L'expérience montre une baisse de l'assiduité ...
    qui se répercute sur la réussite ...
    La moitié de ce qu'on vous demande est assez trivial :
    il faut pratiquer pour assimiler et restituer rapidement

Université de Paris

Contrôle des connaissances

plusieurs notes : en TD, TP, partiel, examen

- note $10\%TP + 20\%TD + 30\%Partiel + 40\%Exam\_mai$
- ou, si elle est meilleure : $10\%TP + 20\%TD + 70\%Exam\_mai$
- rattrapage : $max(examen\_juin; 10\%TP + 20\%TD + 70\%exam\_juin)$

Université de Paris

Contrôle des connaissances

plusieurs notes : en TD, TP, partiel, examen

- note $10\%\,TP + 20\%\,TD + 30\%\,Partiel + 40\%\,Exam\_mai$
- ou, si elle est meilleure : $10\%\,TP + 20\%\,TD + 70\%\,Exam\_mai$
- rattrapage : $max(examen\_juin; 10\%\,TP + 20\%\,TD + 70\%\,exam\_juin)$

Contrôle continu obligatoire

Absence de note en TD ou TP
$\Rightarrow$ pas de calcul de note finale
$\Rightarrow$ session de rattrapage

Université de Paris

Contrôle des connaissances

plusieurs notes : en TD, TP, partiel, examen

- note $10\%TP + 20\%TD + 30\%Partiel + 40\%Exam\_mai$
- ou, si elle est meilleure : $10\%TP + 20\%TD + 70\%Exam\_mai$
- rattrapage : $max(examen\_juin; 10\%TP + 20\%TD + 70\%exam\_juin)$

Contrôle continu obligatoire

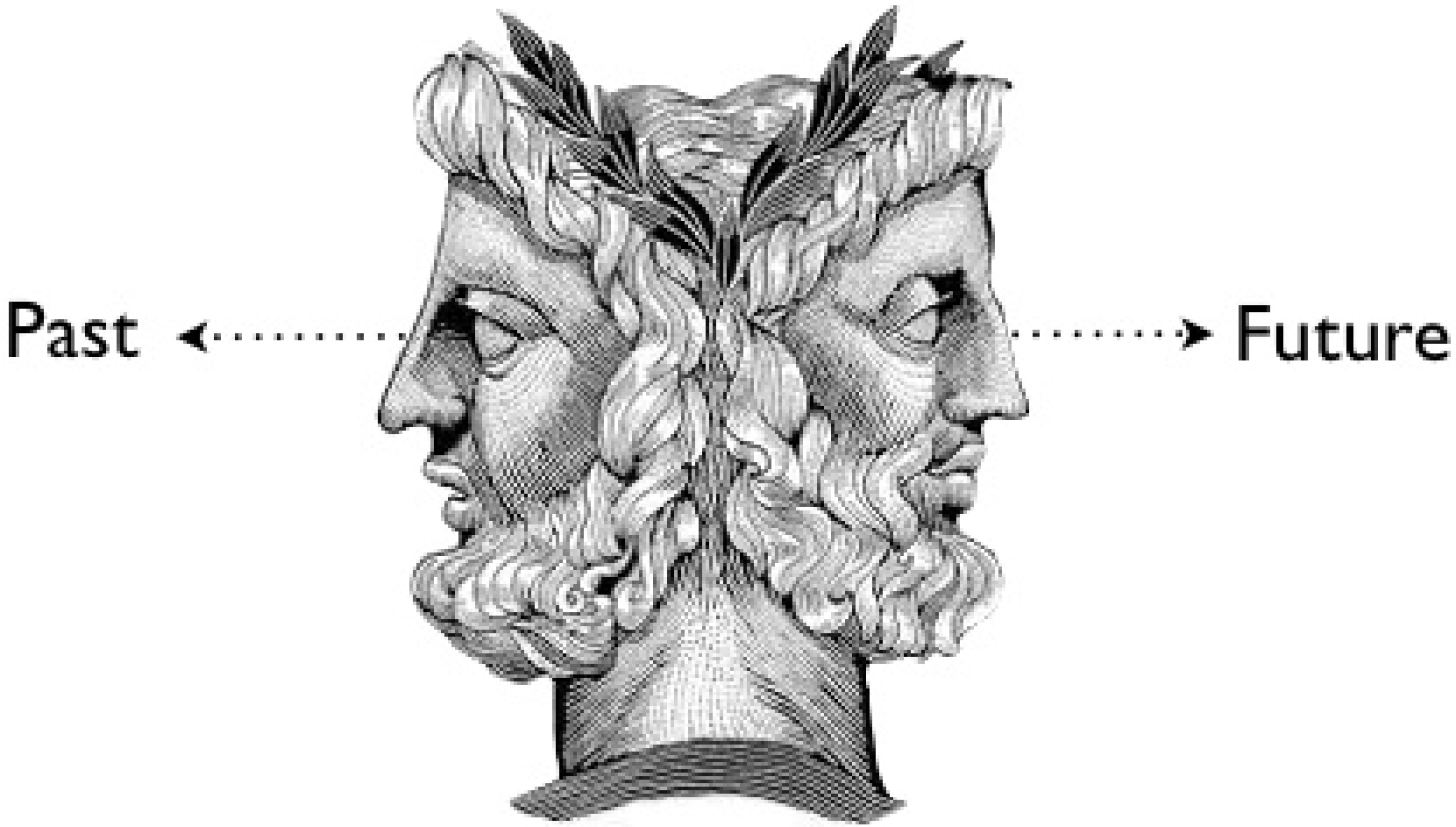
Absence de note en TD ou TP
$\Rightarrow$ pas de calcul de note finale
$\Rightarrow$ session de rattrapage

à priori : 2 contrôles de TD + 1 contrôle de TP

- Si vous anticipez une absence : alertez à l'AVANCE votre chargé de TD/TP
- Respectez les groupes de TD/TP correspondants à votre inscriptions administratives :
  - Les chargés de TD/TP annoncent les CC
  - Le TP noté a un nb de place limité

Université de Paris

Perspective de ce cours



Analogie ingénieur Automobile/Informatique

Automobile

- Conducteur : pilote, taxi ...

Informatique

- Utilisateur

Université de Paris

Analogie ingénieur Automobile/Informatique

Automobile

- Conducteur : pilote, taxi ...
- Bricoleur
  - vidange
  - plaquettes
  - ...

Informatique

- Utilisateur

Université de Paris

Analogie ingénieur Automobile/Informatique

Automobile

- Conducteur : pilote, taxi ...
- Bricoleur
  - vidange
  - plaquettes
  - ...

Informatique

- Utilisateur
- Bricoleur
  - installe logiciel
  - imprimante en wifi
  - choisir un cloud ...

Université de Paris

Analogie ingénieur Automobile/Informatique

Automobile

- Conducteur : pilote, taxi ...
- Bricoleur
  - vidange
  - plaquettes
  - ...
- Garagiste - Constructeur
  - monte / démonte
  - adapte

Informatique

- Utilisateur
- Bricoleur
  - installe logiciel
  - imprimante en wifi
  - choisir un cloud ...

Université de Paris

Analogie ingénieur Automobile/Informatique

Automobile

- Conducteur : pilote, taxi ...
- Bricoleur
  - vidange
  - plaquettes
  - ...
- Garagiste - Constructeur
  - monte / démonte
  - adapte

Informatique

- Utilisateur
- Bricoleur
  - installe logiciel
  - imprimante en wifi
  - choisir un cloud ...
- Développeur - Intégrateur
  - composition
  - manipulation concepts

Université de Paris

Analogie ingénieur Automobile/Informatique

Automobile

- Conducteur : pilote, taxi ...
- Bricoleur
  - vidange
  - plaquettes
  - ...
- Garagiste - Constructeur
  - monte / démonte
  - adapte
- Concepteur des pièces
  - spécialiste
  - prêtes à être branchées

Informatique

- Utilisateur
- Bricoleur
  - installe logiciel
  - imprimante en wifi
  - choisir un cloud ...
- Développeur - Intégrateur
  - composition
  - manipulation concepts

Université de Paris

Analogie ingénieur Automobile/Informatique

Automobile

- Conducteur : pilote, taxi ...
- Bricoleur
  - vidange
  - plaquettes
  - ...
- Garagiste - Constructeur
  - monte / démonte
  - adapte
- Concepteur des pièces
  - spécialiste
  - prêtes à être branchées

Informatique

- Utilisateur
- Bricoleur
  - installe logiciel
  - imprimante en wifi
  - choisir un cloud ...
- Développeur - Intégrateur
  - composition
  - manipulation concepts
- Développeur - Avancé
  - modélise réel $\rightarrow$ virtuel
  - produit une interface

Université de Paris

Analogie ingénieur Automobile/Informatique

Automobile

- Conducteur : pilote, taxi ...
- Bricoleur
  - vidange
  - plaquettes
  - ...
- Garagiste - Constructeur
  - monte / démonte
  - adapte
- Concepteur des pièces
  - spécialiste
  - prêtes à être branchées
- Recherche fondamentale
  - énergie
  - matériaux, ...

Informatique

- Utilisateur
- Bricoleur
  - installe logiciel
  - imprimante en wifi
  - choisir un cloud ...
- Développeur - Intégrateur
  - composition
  - manipulation concepts
- Développeur - Avancé
  - modélise réel $\rightarrow$ virtuel
  - produit une interface

Université de Paris

Analogie ingénieur Automobile/Informatique

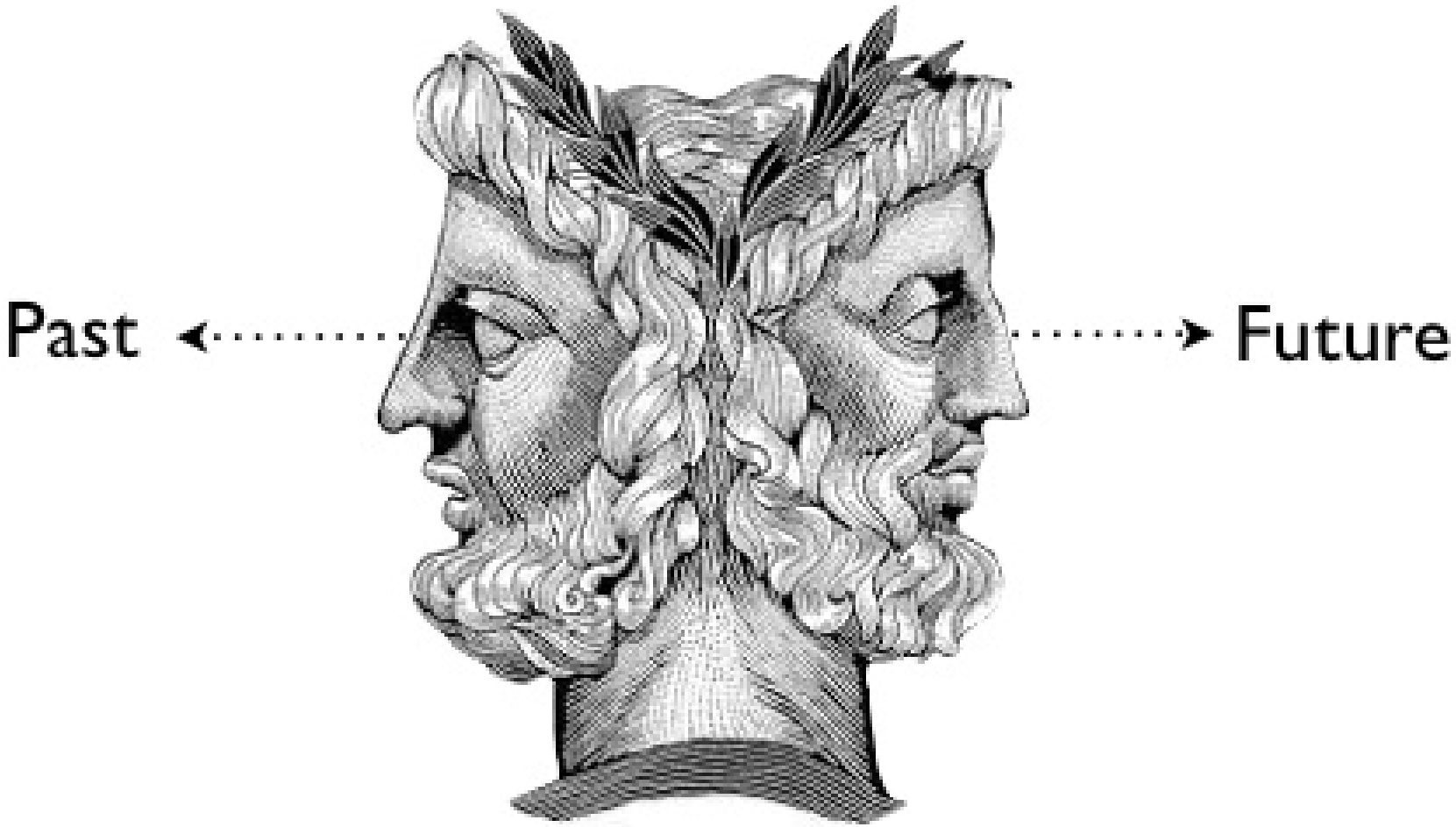
Automobile

- Conducteur : pilote, taxi ...
- Bricoleur
  - vidange
  - plaquettes
  - ...
- Garagiste - Constructeur
  - monte / démonte
  - adapte
- Concepteur des pièces
  - spécialiste
  - prêtes à être branchées
- Recherche fondamentale
  - énergie
  - matériaux, ...

Informatique

- Utilisateur
- Bricoleur
  - installe logiciel
  - imprimante en wifi
  - choisir un cloud ...
- Développeur - Intégrateur
  - composition
  - manipulation concepts
- Développeur - Avancé
  - modélise réel $\rightarrow$ virtuel
  - produit une interface
- Recherche fondamentale
  - calculabilité
  - compléxité, ...

Perspective de ce cours



IP1 résumé

Un programme en Basic, C, Java, Pascal, Php, Python . . . est une suite de

- déclarations variables `int x,y ;` (avec int, float, char, boolean)
- opérations mémoire `x=5 ; y=10 ; x=3+y ;`
- structures de contrôle

```
if (x<y) {x=2*x ;} else
{x=x-1 ;}
```

`while (x<y) {x++ ;}`

`for (int i=0 ; i < 10 ; i++ ) {y=y/x ;}`

- données plus complexes `int [ ] t1=new int [10] ; char [ ][ ] t2 ;`
- modularité `public static int methodeAdd(int a, int b){return a+b ;}`

(IP1 confort syntaxique - 1)

nuances `while` / `do while` :

```
int x=3; // une valeur initiale
do {
 System.out.println("Bonjour");;
 x = x - 1;
} while( x > 0 );
```

Si la valeur initiale de x est > 0, affichera x fois "Bonjour"
Et si elle est ≤ 0, affichera quand même une fois "Bonjour"

```
int x=3; // une valeur initiale
while( x > 0 ) {
 System.out.println("Bonjour");;
 x = x - 1;
};
```

Idem si la valeur initiale de x est > 0, mais si elle est ≤ 0, n'affichera rien

Université de Paris

(IP1 confort syntaxique - 1)

nuances `while` / `do while` :

```
int x=3; // une valeur initiale
do {
 System.out.println("Bonjour");
 x = x - 1;
} while( x > 0 );
```

Si la valeur initiale de x est > 0, affichera x fois "Bonjour"
Et si elle est ≤ 0, affichera quand même une fois "Bonjour"

```
int x=3; // une valeur initiale
if (x<=0) {
 System.out.println("Bonjour"); x = x - 1;
} else {
 while( x > 0 ) {
  System.out.println("Bonjour");
  x = x - 1;
 };
}
```

(IP1 confort syntaxique - 2)

Syntaxe alternative pour les énumérations de cas :

```
int num = 2;
String jour;
if (num==1) jour = "Lundi";
else if (num==2) jour = "Mardi";
else if (num==3) jour = "Mercredi";
else if (num==4) jour = "Jeudi";
else if (num==5) jour = "Vendredi";
else if (num==6) jour = "Samedi";
else if (num==7) jour = "Dimanche";
else jour = "Numéro Invalide";
```

Université de Paris

(IP1 confort syntaxique - 2)

Syntaxe alternative pour les énumérations de cas :

```
int num = 2;
String jour;
switch (num) {
 case 1: jour = "Lundi"; break;
 case 2: jour = "Mardi"; break;
 case 3: jour = "Mercredi"; break;
 case 4: jour = "Jeudi"; break;
 case 5: jour = "Vendredi"; break;
 case 6: jour = "Samedi"; break;
 case 7: jour = "Dimanche"; break;
 default: jour = "Numéro Invalide";
};
```

Université de Paris

(IP1 confort syntaxique - 2)

ou switch expression pour java >=12 :

```
int num = 2;
String jour;
jour = switch (num) {
    case 1 -> "Lundi";
    case 2 -> "Mardi";
    case 3 -> "Mercredi";
    case 4 -> "Jeudi";
    case 5 -> "Vendredi";
    case 6 -> "Samedi";
    case 7 -> "Dimanche";
    default -> "Numéro Invalide";
 };
```

Université de Paris

(IP1 "confort syntaxique" - 3)

Syntaxe alternative pour test/affectation rapides :

```
int min;
if (a<b) min=a;
else min=b;
```

```
int min = (a<b) ? a : b;
```

Université de Paris

(IP1 confort syntaxique - 4)

Syntaxe alternative pour parcourir tous les éléments de tableaux :

```
char [] t={'a','b','c'};
for (int i=0; i<t.length; i++){
 System.out.println( t[i] );
}
```

```
char [] t={'a','b','c'};
for (char val:t){
 System.out.println( val );
}
```

Bien sûr cela se généralise aux tableaux de tableaux :

```
int [][] t= new int[10][10];
for (int i=0;i<10;i++) t[i][i]=2*i; // par exemple pour initialiser
for (int [] x:t)
  for (int y:x)
    System.out.println( y );
```

(IP1 + de confort dans la programmation)

Au S1 vous appreniez le langage :

- l'utilisation du couple emacs/javac se justifiait

Vous êtes maintenant encouragés à utiliser **Eclipse**, **Netbeans**, **Intellij**, **Vscode** . . .qui sont des IDE (Environnement de Développement Intégré)

- environnement de travail moins austère que la console ou emacs
- erreurs de syntaxes détectées à la frappe : gain de temps !
- autre :
  - complétion
  - accès à la documentation java
  - navigation : organisation de votre travail (package etc)
  - . . .

A installer chez vous

Quand ? Ce soir !

Université de Paris

Analogie ingénieur Automobile/Informatique

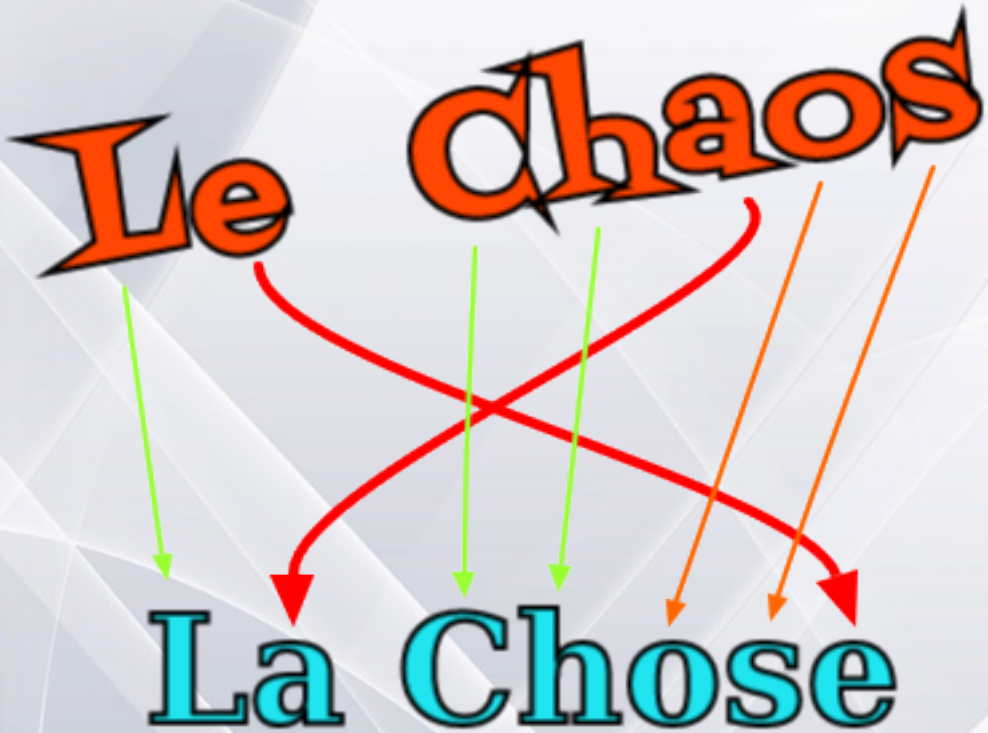
Automobile

- Conducteur
- Bricoleur
  - vidange
  - plaquettes
  - ...
- Garagiste - Constructeur
  - monte / démonte
  - adapte
- Concepteur des pièces
  - spécialiste
  - prêtes à être branchées
- Recherche fondamentale
  - énergie
  - matériaux

Informatique

- Utilisateur
- Bricoleur
  - installe logiciel
  - imprimante en wifi
  - choisit un cloud ...
- Développeur - Intégrateur
  - composition
  - manipulation concepts
- Développeur - Avancé
  - modélise réel $\rightarrow$ virtuel
  - produit une interface
- Recherche fondamentale
  - calculabilité
  - compléxité

Un programme ambitieux



Tout un travail conceptuel consiste à définir des objets.
L'idée originale est souvent confuse (ambigüe, imprécise).
Votre devez prendre le temps de poser des jalons, clarifier, organiser, donner des noms et affecter des rôles aux "choses" qui sortent du "chaos".

Université de Paris

Les objets encapsulent des données - Exemple d'un cercle

(on ne parle pas de l'héritage ce semestre)

fichier : Cercle.java

```
public class Cercle{
 int x,y; // coordonnées du centre
 int rayon;
}
```

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Cercle c;
 }
}
```

Conventions syntaxique

- majuscule au nom de la classe
- la classe est sauvegardée dans un fichier dont le nom est identique à celui de la classe

Les objets encapsulent des données - Exemple d'un cercle

fichier : Cercle.java

```
public class Cercle{
 int x,y; // coordonnées du centre
 int rayon;
}
```

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Cercle c1,c2,c3,c4;
 }
}
```

Université de Paris

Les objets encapsulent des données - Exemple d'un cercle

fichier : Cercle.java

```
public class Cercle{
 int x,y; // coordonnées du centre
 int rayon;
}
```

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Cercle c1,c2,c3,c4;
  Cercle [] tab;
 }
}
```

Université de Paris

Les objets encapsulent des données - Exemple d'un cercle

fichier : Cercle.java

```
public class Cercle{
 int x,y; // coordonnées du centre
 int rayon;
}
```

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Cercle c1,c2,c3,c4;
  Cercle [] tab;
 }
}
```

Ce sont des cercles différents (des instances différentes du même type)

Se pose la question de leur initialisation ...

Université de Paris

Les objets sont des types références (des pointeurs)

comme les tableaux

Leur nature : une référence (une adresse mémoire)

Ils peuvent être initialisés à `null` par exemple. C'est la valeur par défaut

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  int [] t=null;
  Cercle c=null;
  System.out.println(t); // affiche null
  System.out.println(c); // affiche null
 }
}
```

`null` est une valeur compatible pour tous les types références

(tableaux ou objets)

Université de Paris

Les objets sont des types références (des pointeurs)

comme les tableaux

Mais ce sont des mondes différents !

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  int [] t=null;
  Cercle c=null;
  System.out.println(t); // affiche null
  System.out.println(c); // affiche null
  if (c==t) System.out.println("c'est autorisé ça ?") // NON
 }
}
```

A la compilation :

Uncompilable source code - incomparable types : int[] and Cercle

Université de Paris

Les objets sont des types références (des pointeurs)

comme les tableaux

la construction de tableaux s'étend aux tableaux d'objets

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  int [] t = new int [10];
  Cercle [] tab = new Cercle [10];
 }
}
```

Les objets sont des types références (des pointeurs)

comme les tableaux

La construction de tableaux s'étend aux tableaux d'objets

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  int [] t = new int [10];
  Cercle [] tab = new Cercle [10];
  for(int val:t) System.out.println(val); // affiche des 0

  for(Cercle c:tab) System.out.println(c) // affiche des null

  // (rappel) la boucle précédente est équivalente à :
  for (int i=0;i<tab.length;i++) System.out.println(tab[i]);
 }
}
```

Mais on n'a pas encore construit un seul cercle !

Université de Paris

Le couple new / constructeur

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Cercle c;
  c = new Cercle(100,200,20); // en position (100,200) et de rayon 20
 }
}
```

Le constructeur avec ces types de paramètres doit être défini dans la classe Cercle !

fichier : Cercle.java

```
public class Cercle{
 int x,y; // coordonnées du centre
 int rayon;
 ...
}
```

Le couple new / constructeur

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Cercle c;
  c = new Cercle(100,200,20); // en position (100,200) et de rayon 20
 }
}
```

fichier : Cercle.java

```
public class Cercle{
 int x,y; // coordonnées du centre
 int rayon;
 Cercle (int a,int b, int c){ // même nom que la classe !
 // pas besoin de type retour
  x=a; y=b;
  rayon=c;
 }
}
```

Le couple new / constructeur

plusieurs constructeurs sont possibles

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Cercle c;
  c = new Cercle(20); // de rayon 20, centré où ?
 }
}
```

fichier : Cercle.java

```
public class Cercle{
 int x,y; // coordonnées du centre
 int rayon;
 ...
}
```

Université de Paris

Le couple new / constructeur

plusieurs constructeurs sont possibles

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Cercle c;
  c = new Cercle(20); // de rayon 20, centré où ?
 }
}
```

fichier : Cercle.java

```
public class Cercle{
 int x,y; // coordonnées du centre
 int rayon;
 Cercle (int d){ // même nom que la classe, pas de type retour
  x=0; y=0;
  rayon=d;
 }
}
```

Le possible "constructeur par défaut"

cas particulier

Dans le cas où **aucun** constructeur n'est défini, par exemple :

fichier : Cercle.java

```
public class Cercle{
    int x,y; // coordonnées du centre
    int rayon;
}
```

Java met en place un constructeur par défaut qui permet quand même de travailler sur ces objets. Il est donc possible d'écrire, en l'état :

fichier : Test.java

```
public class Test{
    public static void main(String [] args){
        Cercle c;
        c = new Cercle(); // par defaut
    }
}
```

Le possible "constructeur par défaut"

cas particulier

Dans le cas où **aucun** constructeur n'est défini, par exemple :

fichier : Cercle.java

```
public class Cercle{
    int x,y; // coordonnées du centre
    int rayon;
}
```

Cette construction via

```
    c = new Cercle(); // par defaut
```

ne peut pas faire grand chose d'autre qu'affecter une valeur triviale à x, y et *rayon*, en l'occurence 0 qui est la valeur par défaut des `int`

Université de Paris

Le possible "constructeur par défaut"

cas particulier

On peut rendre explicite ce constructeur
(et si on l'écrit, on ne l'appelera plus alors "par défaut")

fichier : Cercle.java

```
public class Cercle{
    int x,y; // coordonnées du centre
    int rayon;
    public Cercle () {x=0;y=0; rayon=0;}
}
```

Le possible "constructeur par défaut"

cas particulier

Notez bien que dès lors que vous écrivez un constructeur quelconque de `Cercle`, le compilateur java n'ajoute plus le constructeur par défaut.

fichier : Cercle.java

```
public class Cercle{
    int x,y; // coordonnées du centre
    int rayon;
    public Cercle (int r) { x=0; y=0; rayon=r;}
}
```

fichier : Test.java

```
public class Test{
    public static void main(String [] args){
        Cercle c;
        c=new Cercle(); // inconnu
    }
}
```

Le couple new / constructeur

effet sur la gestion de la mémoire

1. `new` va réserver un espace mémoire dont la taille dépend du type de l'objet. C'est pourquoi on lui précise le type de l'objet (et du nombre de ces objets si on parle de tableau). Cette réservation, dans le cas des cercles, nécessite 3 entiers par cercle.
   Cela explique la partie de la syntaxe : `"new Cercle...etc..."` ou `"new Cercle[10]"`
2. en java il a été décidé que la politique d'initialisation des attributs des cercles est prise en charge par ces méthodes spéciales que sont les constructeurs. Donc en quasiment en même temps que la réservation mémoire on propose l'appel aux constructeurs. Cela explique la partie de la syntaxe `new Cercle(10)`
   Remarquez que les parenthèses renvoient aux constructeurs, et les crochets aux tableaux.
3. Notez qu'il a été choisi de ne pas autoriser d'appels constructeurs+tableaux du genre new Cercle[10](10)...

Université de Paris

Exercice : modéliser un triangle

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Triangle t;
 }
}
```

fichier : Triangle.java

```
public class Triangle{
  ...
}
```

Université de Paris

Exercice : modéliser un triangle

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Triangle t;
 }
}
```

fichier : Triangle.java

```
public class Triangle{
 int x1,y1; // coordonnées A
 int x2,y2; // coordonnées B
 int x3,y3; // coordonnées C
 ...
}
```

Un peu lourd ... heureusement qu'on n'a pas modélisé un carré

Université de Paris

Exercice : modéliser un triangle

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Triangle t;
 }
}
```

fichier : Triangle.java

```
public class Triangle{
 int [] abscisses; // de taille 3
 int [] ordonnées; // de taille 3
  ...
}
```

moins lourd ... mais on peut faire plus clair

Université de Paris

Exercice : modéliser un triangle

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Triangle t;
 }
}
```

fichier : Triangle.java

```
public class Triangle{
 int [] coordonnées; // de taille 6
  ...
}
```

encore moins lourd ... mais encore moins clair

Université de Paris

Exercice : modéliser un triangle

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Triangle t;
 }
}
```

fichier : Triangle.java

```
public class Triangle{
 Point a,b,c;
  ...
}
```

Université de Paris

Exercice : modéliser un triangle

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Triangle t;
 }
}
```

fichier : Triangle.java

```
public class Triangle{
 Point [] tab;
  ...
}
```

Exercice : modéliser un triangle

fichier : Test.java

fichier : Triangle.java

```
public class Triangle{
 Point [] tab;
 ...
}
```

fichier : Point.java

```
public class Point{
 int x,y;
 Point(int a,int b){ // un constructeur
  x=a; y=b;
 }
}
```

Exercice : modéliser un triangle en plus du cercle

fichier : Test.java

fichier : Triangle.java

```
public class Triangle{
 Point [] tab;
 ...
}
```

fichier : Point.java

```
public class Point{
 int x,y;
 Point(int a,int b){
  x=a; y=b;
 }
}
```

fichier : Cercle.java

```
public class Cercle{
 int x,y;
 int rayon;
 Cercle (int a,int b,int c){
  x=a; y=b;
  rayon=c;
 }
}
```

Université de Paris

Exercice : modéliser un triangle en plus du cercle

fichier : Test.java

fichier : Triangle.java

```
public class Triangle{
 Point [] tab;
 ...
}
```

fichier : Point.java

```
public class Point{
 int x,y;
 Point(int a,int b){
  x=a; y=b;
 }
}
```

fichier : Cercle.java

```
public class Cercle{
 Point p;
 int rayon;
 Cercle (int a,int b,int c){
  p=new Point(a,b);
  rayon=c;
 }
}
```

Université de Paris

Exercice : modéliser un triangle en plus du cercle

fichier : Test.java

fichier : Triangle.java

```
public class Triangle{
 Point [] tab;
 ...
}
```

fichier : Point.java

```
public class Point{
 int x,y;
 Point(int a,int b){
  x=a; y=b;
 }
}
```

fichier : Cercle.java

```
public class Cercle{
 Point p;
 int rayon;
 Cercle (int a,int b,int c){
  p=new Point(a,b);
  rayon=c;
 }
 Cercle (Point x, int d){
  p=x; // partage de référence
  rayon=d;
 }
}
```

Université de Paris

Exercice : modéliser un triangle en plus du cercle

fichier : Test.java

fichier : Triangle.java

```
public class Triangle{
 Point [] tab;
 Triangle(Point a,Point b,Point c){
  tab=new Point[3];
  tab[0]=a;
  tab[1]=b; tab[2]=c;
 }
}
```

fichier : Point.java

fichier : Cercle.java

```
public class Cercle{
 Point p;
 int rayon;
 Cercle (int a,int b,int c){
  p=new Point(a,b);
  rayon=c;
 }
 Cercle (Point x, int d){
  p=x; // partage de référence
  rayon=d;
 }
}
```

Université de Paris

Objets et niveaux d'abstractions

- Manipuler un objet permet d'améliorer la compréhension globale.

fichier : Test.java

```
Cercle [] tab= new Cercle[2];
tab[0] = new Cercle (0,0,10);
tab[1] = new Cercle (1,1,5);
Cercle tmp=tab[0];
tab[0]=tab[1];
tab[1]=tmp;
```

- Un objet (conçu pour regrouper des informations) peut aussi être détaillé/décomposé (on regarde ce qu'il contient)

fichier : Test.java

```
Cercle c= new Cercle(0,0,10);
int diametre = c.rayon*2; // on pénètre la structure avec le .
Point centre = c.p;
int abscisse = c.p.x;
int ordonnée = centre.y;
centre.x=-1; // remarquez les changements induits (car reference)
```

Classes : définir un modèle et regrouper des méthodes

Le fichier définissant la classe d'objets :

- Porte le nom de la classe
- Contient les attributs/champs définissant les objets
- Contient le/les constructeurs (ou celui par défaut)
- Contient les méthodes qui "concernent" [1] ces objets

fichier : Point.java

```
public class Point{
 int x,y;
 Point (int a,int b) {x=a; y=b;}
 public static double distance( Point a , Point b){
  int dx = b.x - a.x;
  int dy = b.y - a.y;
  return  Math.sqrt ( dx*dx + dy*dy); // la racine carrée
 }
}
```

1. c'est parfois un peu subjectif

Classes : définir un modèle et regrouper des méthodes

- Contient les méthodes qui "concernent" [1] ces objets

fichier : Point.java

```
public class Point{
 int x,y;
 Point (int a,int b) {x=a; y=b;}
 public static double distance( Point a , Point b){
  int dx = b.x - a.x;
  int dy = b.y - a.y;
  return Math.sqrt ( dx*dx + dy*dy); // la racine carrée
 }
}
```

- Remarquez l'appel externe à la méthode statique sqrt de Math
- Le même mécanisme permettra Point.distance(arg1,arg2)

1. c'est parfois un peu subjectif

Classes : définir un modèle et regrouper des méthodes

Application - Exemple

Si on s'intéresse au périmètre d'un triangle :

- la méthode statique **périmètre** est à définir dans la classe **Triangle**
- elle fait appel à **distance** écrite dans **Point**
- elle même fait appel à **sqrt** écrite dans **Math**

fichier : Triangle.java

```
public class Triangle{
 Point [] tab;
 // ... et ici il y a toujours nos constructeurs ...
 public static double perimetre(Triangle t){ // nouvelle méthode
  double rep= Point.distance( t.tab[0] , t.tab[1] );
  rep += Point.distance( t.tab[1] , t.tab[2] );
  rep += Point.distance( t.tab[2] , t.tab[0] );
  return rep;
 }
}
```

Petit exercice - Complétez le code

fichier : Triangle.java

```
public class Triangle{
 // qui contient en particulier :
 public static double perimetre(Triangle t){ // etc }
}
```

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Point a=new Point(0,0);
  Point b=new Point(10,10);
  Point c=new Point(20,0);
  Triangle t=new Triangle(a,b,c);
  // calcul du périmètre de t
  double longueur = ... ; // à compléter
  System.out.println(" Le périmètre est " + longueur);
 }
}
```

Petit exercice

fichier : Triangle.java

```
public class Triangle{
 // qui contient en particulier :
 public static double perimetre(Triangle t){ // etc }
}
```

fichier : Test.java

```
public class Test{
 public static void main(String [] args){
  Point a=new Point(0,0);
  Point b=new Point(10,10);
  Point c=new Point(20,0);
  Triangle t=new Triangle(a,b,c);
  // calcul du périmètre de t
  double longueur = Triangle.perimetre(t);
  System.out.println(" Le périmètre est " + longueur);
 }
}
```

Exercices

- Remarquez que l'on peut aussi choisir de modéliser un cercle par 2 points quelconques d'un de ses diamètres.
  Définissez cette classe en suivant cette idée, et écrivez une méthode périmètre pour ces cercles.

Université de Paris

Exercices

- Réfléchissez à un environnement où on devrait manipuler à la fois une modélisation d'un permis de conduire à points et d'une police d'assurance. Combien de classes voudriez vous définir ?
  Comment retire t'on des points ? Jusqu'à quand ?
  Peut-on assurer un conducteur qui n'a plus de points à son permis ?

Université de Paris