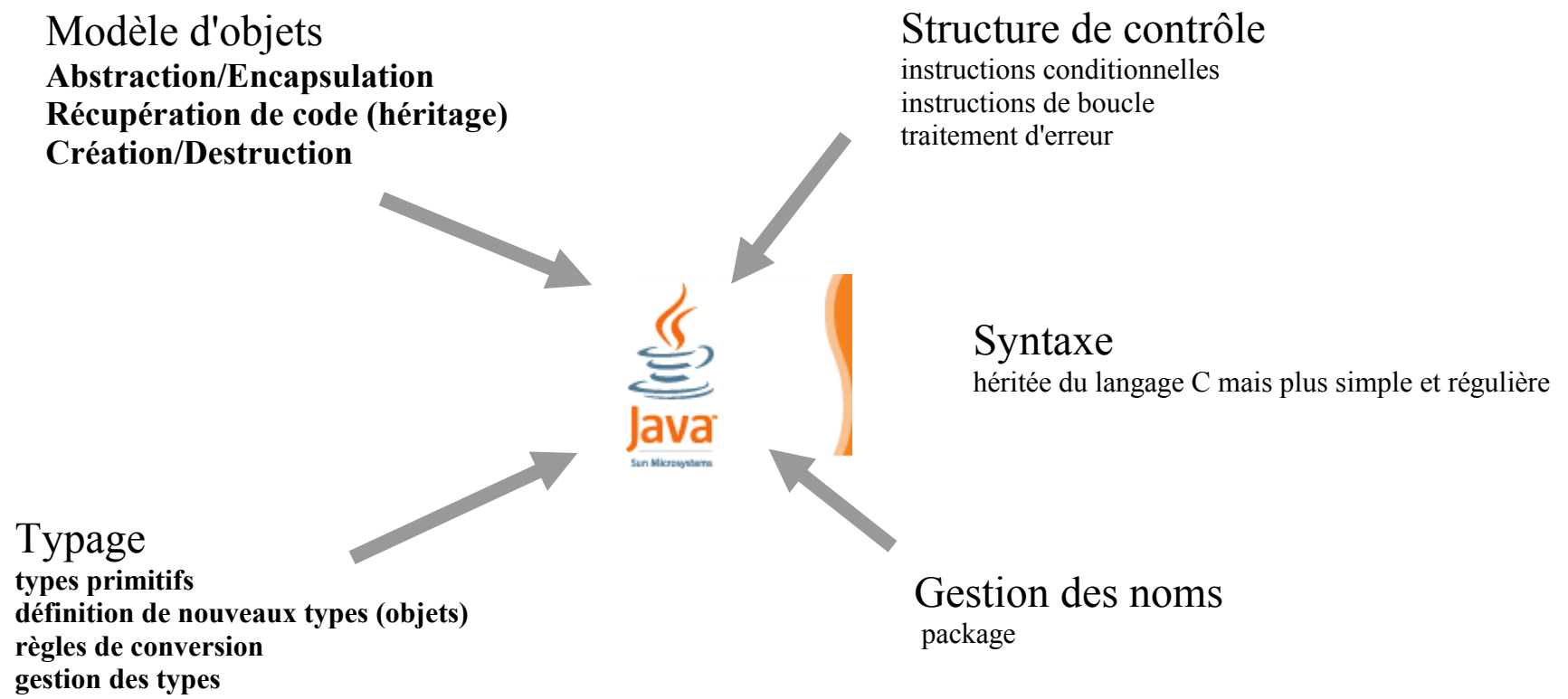


# présentation générale

java de base	chapitre 01
contenu de la section	
PRÉSENTATION GÉNÉRALE.....	1
CONTENU DE LA SECTION.....	2
LE LANGAGE JAVA .....	3
UNE INTRODUCTION AU JAVA FRAMEWORK (1).....	4
<i>les 3 éditions du "Java Framework" (Java 2 Platform)</i> .....	4
<i>une rapide introduction à J2SE</i> .....	4
UNE INTRODUCTION AU JAVA FRAMEWORK (2) .....	5
<i>les éléments intervenant dans l'exécution d'un programme</i> .....	5
QUELQUES RAPPELS SUR LES MODÈLES À OBJETS (1).....	6
<i>les objectifs et les principes</i> .....	6
<i>quelques termes de base</i> ... ..	6
QUELQUES RAPPELS SUR LES MODÈLES À OBJETS (2).....	7
<i>la conception/fabrication par assemblage</i> .....	7
<i>le contrôle d'accès</i> .....	7
QUELQUES RAPPELS SUR LES MODÈLES À OBJETS (3).....	8
<i>la réutilisation du code</i> .....	8
JAVA ET LES MODÈLES À OBJETS.....	9
<i>qu'est ce qu'un objet java, une classe, une interface, .... ?</i> .....	9
<i>le point de vue de l'utilisateur : les références</i> .....	9
<i>le point de vue du concepteur</i> .....	9
UNE VUE GÉNÉRALE DE JAVA (1).....	10
<i>un exemple : les points de vue de l'utilisateur et du concepteur</i> .....	10
UNE VUE GÉNÉRALE DE JAVA (2).....	11
<i>un exemple : l'utilisation d'une interface permettant de faire abstraction de l'implémentation</i> .....	11
UNE VUE GÉNÉRALE DE JAVA (3).....	12
<i>un exemple : la factorisation du code</i> .....	12
UNE VUE GÉNÉRALE DE JAVA (4).....	13
<i>la classe Object</i> .....	13
QUELQUES RAPPELS SUR LES STRUCTURES DE CONTRÔLE.....	14
<i>les instructions de contrôle</i> .....	14
UNE VUE GÉNÉRALE DE JAVA (5).....	15
<i>un exemple illustrant quelques instructions de contrôle</i> .....	15
version du 27/06/2010	page 2

# Le langage java



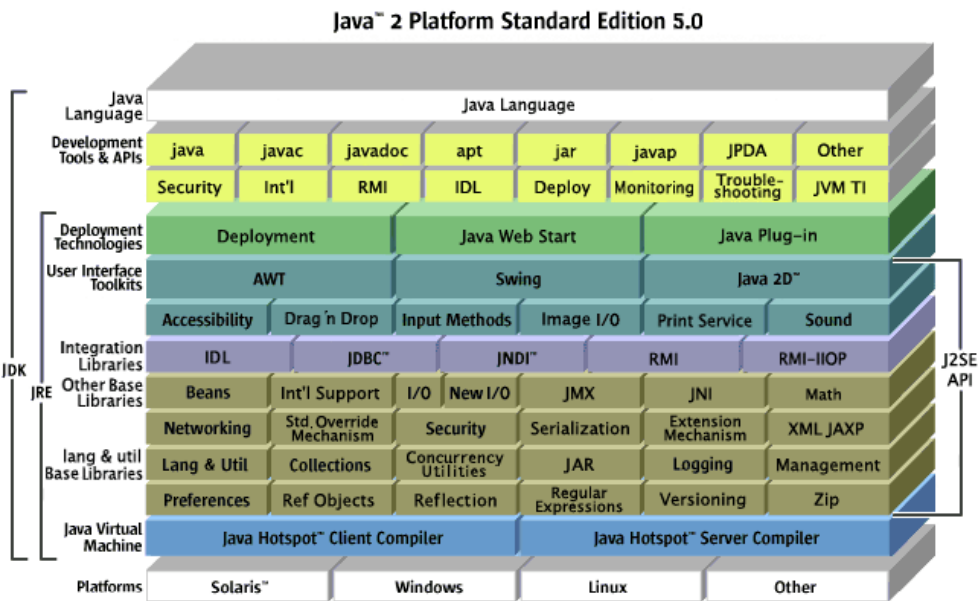
# Une introduction au Java Framework (1)

## les 3 éditions du "Java Framework" (Java 2 Platform)

- **J2ME** : *Java 2 Micro Edition* : développement d'applications embarquées (assistants personnels,terminaux mobiles)
  - **J2SE** : *Java 2 Standard Edition* : développement d'applications pour stations de travail
  - **J2EE** : *Java 2 Enterprise Edition* : *développement d'applications* avec mise en oeuvre de serveurs.
- Chaque édition propose un environnement complet pour le développement et l'exécution d'applications

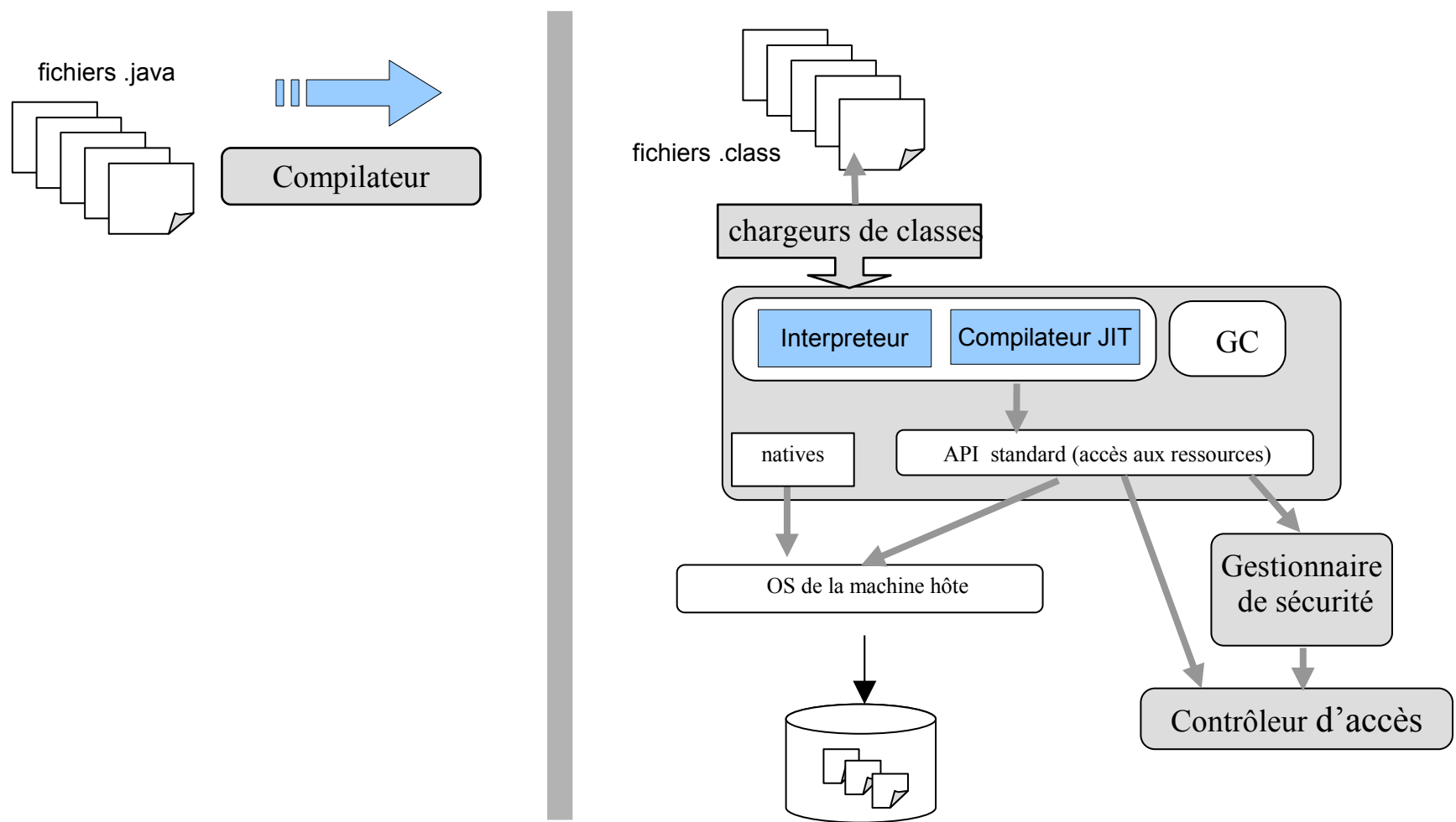
## une rapide introduction à J2SE

- un ensemble d'outils et d'API
  - pour le développement, le debug, la supervision, le monitoring, ..., le déploiement
  - l'intégration dans les systèmes existants
  - l'accès aux services (JNDI, JDBC, ...)
- des plateformes d'exécution
  - les JVM (client-serveur) pour divers OS



## Une introduction au Java Framework (2)

les éléments intervenant dans l'exécution d'un programme


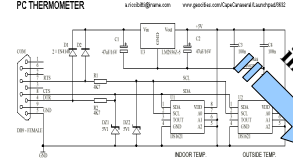




Quelques rappels sur les Modèles à Objets (1)

les objectifs et les principes

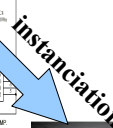
- la réutilisation du code : la définition d'entités logicielles réutilisables (première étape vers les composants logiciels)
- l'analogie avec les composants matériels réutilisables :
  - objets,
  - mode d'emploi (vision utilisateur : comment utiliser les objets),
  - plans de fabrication (vision fabriquant : comment fabriquer les objets),
- l'abstraction : la vision offerte à l'environnement, en particulier comment utiliser le composant
- l'encapsulation : la réalisation (qui satisfait l'abstraction)  
selon le langage : le moyen de stocker les données et de réaliser les traitements

quelques termes de base ...

interface	: <b>spécification abstraite (des interactions et des attributs)</b> partie du modèle indépendante de la réalisation (l'implémentation)	
classe	: modèle pour la réalisation (modèle d'objet)	
objet	: instance d'une classe (instanciation)	
champ	: donnée propre à une classe ou une instance	
méthode	: fonction propre à une classe ou une instance	
référence	: désignation d'un objet	



implementation

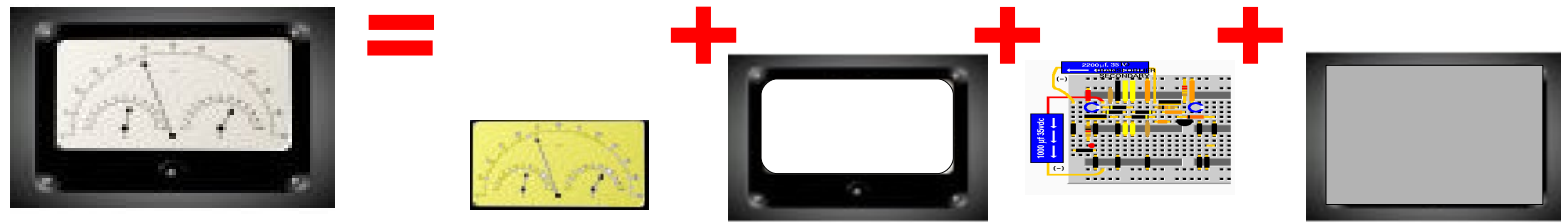


instanciation

## Quelques rappels sur les Modèles à Objets (2)

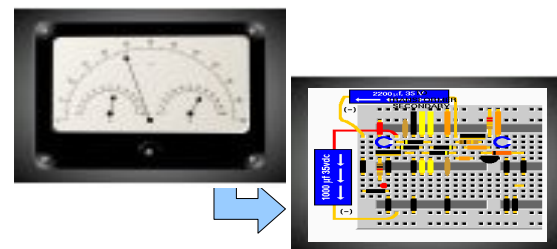
### la conception/fabrication par assemblage

- la conception/fabrication d'un système (logiciel) repose en partie sur l'assemblage d'objets : l'agrégation/composition



### le contrôle d'accès

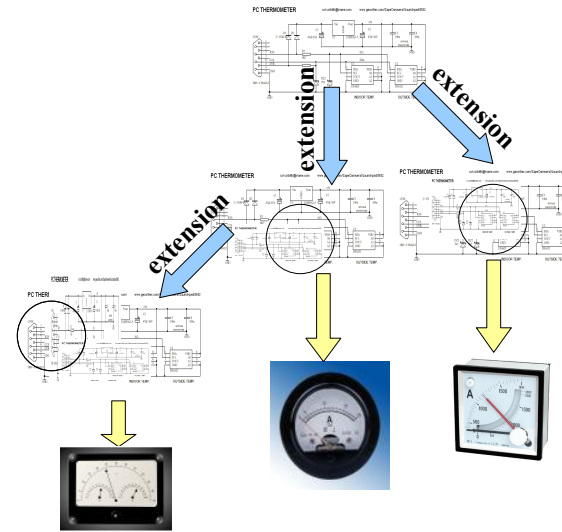
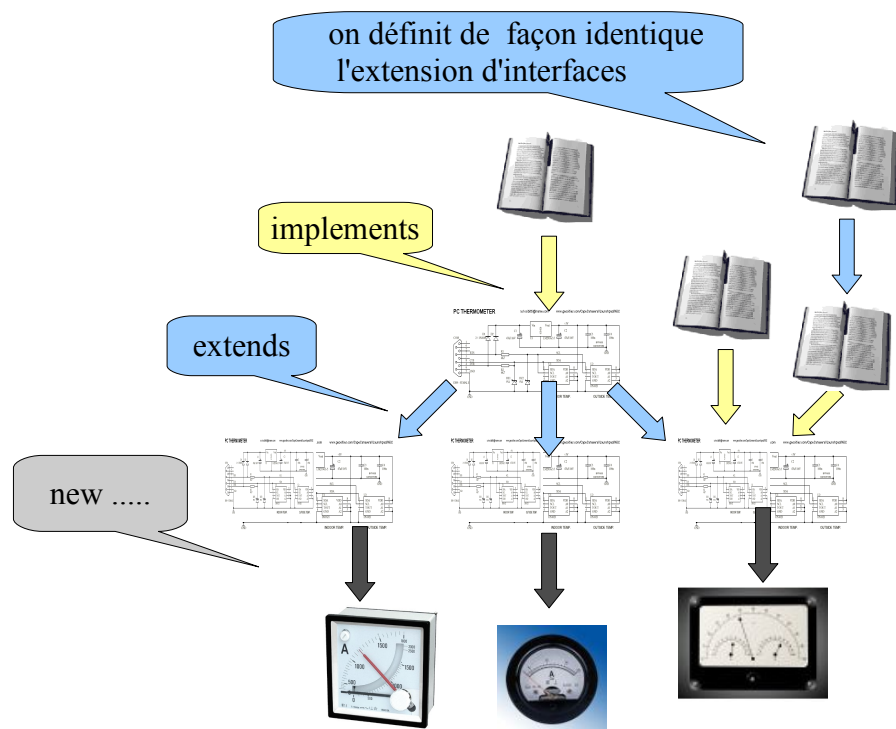
- Le contrôle d'accès a pour objectif de définir les façons légales d'utiliser/construire un objet :
  - les façons légales d'assembler les objets "composants"
  - les façons légales d'utiliser les objets
- les droits d'accès courants sont :  
**public**, **privé**, autorisé pour des "utilisateurs" privilégiés (*voir suite*)



## Quelques rappels sur les Modèles à Objets (3)

### la réutilisation du code

- permettre la factorisation du code : composants logiciels
- elle introduit la notion de classe abstraite
- elle est mise en oeuvre en java par l'**extension de classes**





## Java et les Modèles à Objets

### qu'est ce qu'un objet java, une classe, une interface, .... ?

- un objet = un conteneur d'information + un ensemble d'opérations permettant de les utiliser (connaître/modifier leur valeur, ...)
  - une opération se manifeste par une méthode (une fonction)
- une classe = un modèle de réalisation (les moyens de stocker les informations, comment réaliser les opérations)
- une interface = l'ensemble des opérations mises à la disposition des utilisateurs

### le point de vue de l'utilisateur : les références

- un objet est **TOUJOURS** manipulé par le biais d'une référence
  - une référence : une désignation d'un objet
  - l'utilisation des objets
    - invocation des méthodes
    - accès aux champs
- un objet est **TOUJOURS** créé de façon explicite
  - l'initialisation des objets : les constructeurs

#### //création des références

```
Voltmetre p1;  
Voltmetre p2;  
.....
```

#### //création des instances avec initialisation (constructeur)

```
p1=new Voltmetre(sensibilite1,vMax1);  
p2=new Voltmetre(sensibilite2,vMax2);  
.....
```

#### //utilisation des instances via les références

```
double v = p1.mesure();  
.....  
v =p2.mesure();  
.....
```


### le point de vue du concepteur

- un objet est un ensemble de :
  - variables : espaces mémoire (généralement typés) utilisés pour le stockage de l'information
  - fonctions : blocs d'instructions nommés [utilisant des valeurs reçues en paramètre]
- la définition d'une classe consiste à décrire les éléments constitutifs d'un objet (comment construire un objet)

## Une vue générale de Java (1)

### un exemple : les points de vue de l'utilisateur et du concepteur

```
class Test {                                     //classe utilisant la classe Point2D
    public static void main(String[] args) {
        Point2D p1, p2;                         //définition de références de classe Point2D
        p1 = new Point2D();                     //création d'instances de la classe Point2D
        p2 = new Point2D(10, 20);
        p1.deplacer(10,10);                     //utilisation des instances : invocation de la méthode deplacer
        System.out.println("le point est en " + p1.getX() + " et " + p1.getY());
        System.out.println("le nombre de points créés est " + Point2D.getNbPoints());
    }
}
```


utilisateur

```
class Point2D {                                 //classe dont le nom est Point2D
    private static int nbPoints;                //champs de classe
    private int x;                             //champs d'instance
    private int y;                             //idem

    public Point2D(int a0, int a1) { x = a0; y = a1; nbPoints++; } //constructeur
    public Point2D() { x = 0; y = 0; nbPoints++; } //idem

    public void deplacer(int dx,int dy) { x += dx; y += dy; }      //méthodes d'instance
    public int getX() { return x; }                                //idem
    public int getY() { return y; }

    public static int getNbPoints() { return nbPoints; }           //méthodes de classe
}
```

concepteur

## Une vue générale de Java (2)

un exemple : l'utilisation d'une interface permettant de faire abstraction de l'implémentation

```
class Test {  
    public static void main(String[] args) {  
        IForme2D p1 = new Ligne();           //définition de références d'interface IPoint2D  
        p1.addSommet(new Point2D());  
        p1.addSommet(new Point2D(10, 20)); //utilisation des instances  
        .....  
        System.out.println("la longueur de la ligne est " + p1.getLongueur ());  
    }  
}
```

Seul compte pour l'utilisateur,  
la façon d'utiliser un objet

```
interface IForme2D {  
    public void addSommet(Point2D p);  
    public void removeSommet(Point2D p);  
    public int getNbSommets();  
    public double getLongueur();  
    .....  
}
```

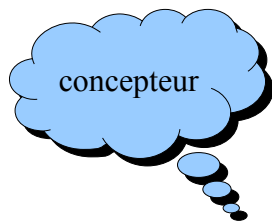
Seul compte pour l'utilisateur,  
la façon d'utiliser un objet

```
class Ligne implements IForme2D {  
    private ArrayList sommets;  
  
    public Ligne() { sommets = new ArrayList(); }  
    public void addSommet(Point2D s) { sommets.add(s); }  
    public void removeSommet(Point2D s) { sommets.remove(s); }  
    public int getNbSommets() { return sommets.size(); }  
  
    public double getLongueur() { ..... }  
    public Point2D[] getSommets() { ..... }  
    .....  
}
```

indique la conformité au point  
de vue "utilisateur" exprimé par  
l'interface IPoint2D

## Une vue générale de Java (3)

### un exemple : la factorisation du code



```
class Point2DColor extends Point2D {
    private int couleur;

    public Point2DColor(int _x, int _y, int c) { super(_x, _y); couleur=c; }
    public Point2DColor() { super(0, 0) ; couleur=0; }

    public void setCouleur(int c) { couleur=c ; }
    public int getCouleur() { return couleur; }
}
```

```
class Point2D {
    private static int nbPoints;
    private int x;
    private int y;

    public Point2D(int _x, int _y) { x = _x; y = _y; nbPoints++; }
    public Point2D() { x = 0 ; y = 0; nbPoints++; }
    public void deplacer(int dx,int dy) { x += dx; y += dy; }
    public int getX() { return x; }
    public int getY() { return y; }

    public static int getNbPoints() { return nbPoints; }
}
```

//super(\_x,\_y) référence le constructeur Point2D(x,y)  
 //super(0,0) référence le constructeur Point2D(0,0)

//définition de nouvelles méthodes  
 //idem

```
class Test
    public static void main(String[ ] args) {
        Point2DColor p1;
        p1 = new Point2DColor(3, 15, 2);
        System.out.println("le point est en " + p1.getX() + " et " + p1.getY());
        System.out.println(" la couleur est " + p1.getCouleur() );
        p1.deplacer(10,10);
        p1.setCouleur(3);
    }
}
```

## Une vue générale de Java (4)

### la classe Object

- il existe en Java la classe Object qui est étendue (extended) par toutes les autres classes
  - toutes les classes récupèrent les méthodes définies dans Object (en particulier toString et equals)

```
class Point2D {
    private static int nbPoints;
    private int x;
    private int y;

    public Point2D(int _x, int _y) { x = _x; y = _y; nbPoints++; }
    public Point2D() { x = 0 ; y = 0; nbPoints++; }
    public void deplacer(int dx,int dy) { x += dx; y += dy; }
    public int getX() { return x; }
    public int getY() { return y; }

    public static int getNbPoints() { return nbPoints; }
}
```

```
class Test
    public static void main(String[] args) {
        Point2D p1;
        p1= new Point2D();
        System.out.println(p1.toString());
    }
}
```

## Quelques rappels sur les structures de contrôle

### les instructions de contrôle

- les instructions conditionnelles :
  - if (*condition\_booléenne*) instruction ou if (*condition\_booléenne*) instruction1 else instruction2
- les instructions de boucle :
  - for (*expression\_initialisation*; *expression\_continuation*; *expression\_increment*) instruction
  - while(*expression\_booléenne*) instruction
  - do instruction while(*expression\_booléenne*);
- les instructions de branchement :
  - return; ou return expression;
  - switch(expression\_entière) { case constante entière : ..... default : }
  - break; ou break *etiquette*;
  - continue; ou continue *etiquette*;
- les instructions pour le traitement d'erreur
  - throw

associé à :  
try  
    bloc\_instructions  
catch(Exception e) bloc\_instructions

## Une vue générale de Java (5)

### un exemple illustrant quelques instructions de contrôle

```
class Loterie {
    private long somme=1000 ;

    public int jouerUneFois() { return (int) (Math.random()*100)-50 ; }

    public void jouerNFois(int nf) {
        for (int i=0; i<nf; i++) { somme+=jouerUneFois() ; }
    }

    public int jouerJusquaUnCoupGagnant() {
        int gain, nFois=0 ;
        do {
            nFois++ ;
            gain=jouerUneFois() ;
            somme=somme+gain ;
        } while (gain<=0) ;
        return nFois;
    }

    public int jouerTantQueSommePositive() {
        int nFois=0 ;
        while (somme>0) {
            nFois++ ;
            somme=somme+ jouerUneFois() ;
        }
        return nFois;
    }
    .....
}
```