

# TP 3



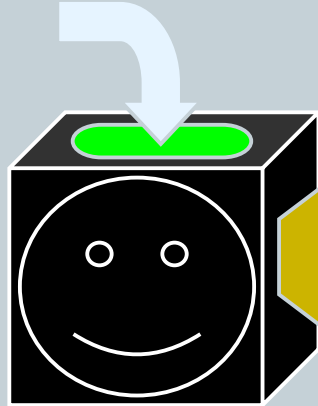
## LES CLASSES EN JAVA - CONSTRUCTEUR -

# Nos souvenirs – TP3

- Qu'est-ce qu'un objet?

→ Une boîte noire qui reçoit et envoie des messages

**Votre groupe  
SVP?**



**DSI 21!**

# Nos souvenirs – TP3



Que contient cette boîte?

Du code → Traitements composés d'instructions → **Comportements**

Des données → L'information traitée par les instructions et qui caractérise l'objet → **Etats** ou **Attributs**

→ Données et traitements sont donc indissociables

→ Les traitements sont conçus pour une boîte en particulier et ne peuvent pas s'appliquer à d'autres boîtes

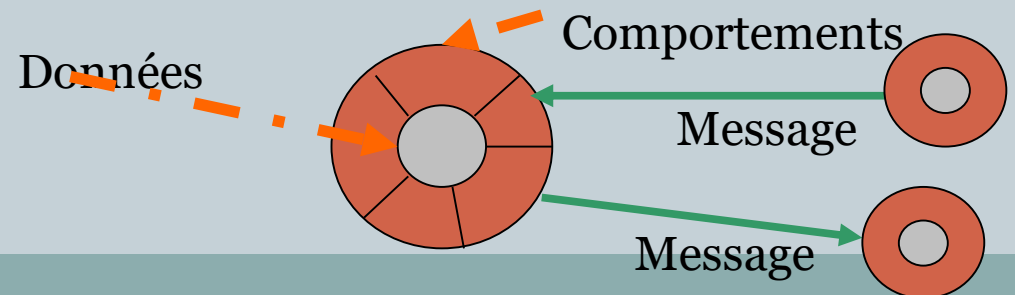
# Nos souvenirs – TP3



## Pourquoi une boîte noire?

- L'utilisateur d'un objet ne devrait jamais avoir à plonger à l'intérieur de la boîte
- Toute l'utilisation et l'interaction avec l'objet s'opère par messages
- Les messages définissent l'interface de l'objet donc la façon d'interagir avec eux
- Il faut et il suffit de connaître l'interface des messages pour pouvoir exploiter l'objet à 100%, sans jamais avoir à connaître le contenu exact de la boîte ni les traitements qui l'animent
- L'intégrité de la boîte et la sécurité de ses données peut être assurée
- Les utilisateurs d'un objet ne sont pas menacés si le concepteur de l'objet en change les détails ou la mécanique interne

### → ENCAPSULATION



# Nos souvenirs – TP3



## Comment les objets sont-ils définis?

- Par leur classe, qui détermine toutes leurs caractéristiques
  - Nature des attributs et comportements possibles
- La classe détermine tout ce que peut contenir un objet et tout ce qu'on peut faire de cet objet
- Classe = Moule, Définition, ou Structure d'un d'objet
- Objet = Instance d'une classe

# Nos souvenirs – TP3



Un programme orienté objet est uniquement constitué de classes interagissant par envoi de messages

L'intégralité du code d'un programme orienté objet se trouve donc à l'intérieur de classes

# Nos souvenirs – TP3



- Une variable est un endroit de la mémoire à laquelle on a donné un nom de sorte que l'on puisse y faire facilement référence dans le programme
- Une variable a une valeur, correspondant à un certain type
- La valeur d'une variable peut changer au cours de l'exécution du programme
- Une variable Java est conçue pour un type particulier de donnée

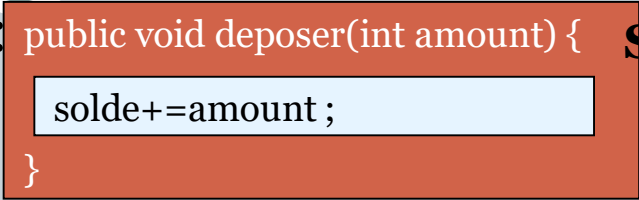
# Nos souvenirs – TP3



- Rappel: toute variable doit être déclarée et initialisée
- Les variables membres sont des variables déclarées à l'intérieur du corps de la classe mais à l'extérieur d'une méthode particulière, elles sont donc accessibles depuis n'importe où dans la classe.
- La signature de la variable :
  - ✦ Les modificateurs d'accès: indiquent le niveau d'accessibilité de la variable
  - optionnel** {
    - ✦ [static]: permet la déclaration d'une variable de classe
    - ✦ [final]: empêche la modification de la variable
    - ✦ [transient]: on ne tient pas compte de la variable en sérialisant l'objet
    - ✦ [volatile]: pour le multithreading
  - ✦ Le type de la variable (ex: int, String, double, RacingBike,...)
  - ✦ Le nom de la variable (identificateur)



# Nos souvenirs – TP3

- Une méthode est composée de:
  - 
  - Sa déclaration**
  - Son corps**
- Signature d'une méthode:
  - ✦ Modificateurs d'accès : public, protected, private, aucun
  - ✦ [modificateurs optionnels] : static, native, synchronized, final, abstract
  - ✦ Type de retour : type de la valeur retournée
  - ✦ Nom de la méthode (identificateur)
  - ✦ Listes de paramètres entre parenthèses (peut être vide mais les parenthèses sont indispensables)
  - ✦ [exception] (throws Exception)
- Au minimum:
  - La méthode possède un identificateur et un type de retour
  - Si la méthode ne renvoie rien → le type de retour est void
- Les paramètres d'une méthode fournissent une information depuis l'extérieur du "scope" de la méthode (idem que pour le constructeur)

# Nos souvenirs – TP3



- L'interface d'une méthode, c'est sa signature
- Cette signature, qui définit l'interface de la méthode, correspond en fait au message échangé quand la méthode est appelée
- Le message se limite de fait uniquement à la signature de la méthode
  - Type de retour
  - Nom
  - Arguments
- L'expéditeur du message n'a donc jamais besoin de connaître l'implémentation ou corps de la méthode
- On a donc:
  - Déclaration = Signature = Message de la méthode
  - Bloc d'instruction = Corps = Implémentation de la méthode

# *Problème 1–L’Enoncé–*



Il s'agit d'écrire le code en Java de la classe Compte.

# Problème 1–La Correction-



Déclaration  
de la classe

Variables d'instance  
ou « champs »

Définition du  
constructeur

Méthodes d'accès

Définition  
des  
méthodes

```
public class Compte{
```

Corps Class

```
private String nom;
```

```
private int solde ;
```

```
private int numero;
```

```
public Compte(String n,int s,int i){
```

```
    nom=n ;
```

```
    solde=s;
```

```
    numero=i;}
```

```
public String getName() { return nom; }
```

```
public void setName (String n) {nom= n;}
```

```
public void deposer(int amount) {
```

```
    solde += amount ;
```

```
}
```

```
public void retirer(int amount ){
```

```
    solde-=amount ;}
```

## *Problème 2 -Exercice 1–L’Enoncé-*



- Tous les documents possèdent un titre. Quand un document est créé, son titre est donné à la création et par la suite, il ne change plus.
- Définir la classe Document avec son constructeur public, et la propriété titre privée et son accesseur. (Les accesseurs sont des méthodes fournissant ou modifiant certaines des caractéristiques de la classe).

# *Problème 2 -Exercice 1–L'Enoncé-*



- On veut attribuer un numéro d'enregistrement unique dès que l'on crée un objet Document. On veut que le premier document créé ait le numéro 0, et que ce numéro s'incrémente de 1 à chaque création de document. Quelles propriétés faut-il ajouter à la classe Document ? Lesquelles doivent être static ? Les ajouter sans leurs accesseurs. Puis ajouter une méthode getNumero renvoyant le numéro d'enregistrement du document. **(Certains attributs peuvent être communs à tous les objets de la classe et exister indépendamment de tout objet de la classe. Ces attributs sont déclarés static. On peut trouver des variables static et des constantes static.)**

# *Problème 2 -Exercice 1–La correction-*



```
public class Document {  
    static private int numeroSuivant = 0;  
    private int numero;  
    private String titre;  
public Document(String titre) {  
    this.numero = numeroSuivant;  
    numeroSuivant++;  
    this.titre = titre;  
}  
public int getNumero(){ return numero;}  
public String getTitre(){ return titre; }  
}
```

## *Problème 2 -Exercice 1–L’Enoncé-*



- Ajouter un second constructeur public permettant de définir aussi, en plus du titre, le numéro d'enregistrement (supposé positif ou nul) lors de la création d'un document. Quelles restrictions peut-on prendre afin d'éviter que deux documents créés aient le même numéro ? Dans les cas de restriction, on prendra le numéro suivant prévu, à la place du numéro indiqué par l'utilisateur.



# *Problème 2 -Exercice 1–L’Enoncé-*



- Redéfinir la méthode toString renvoyant la chaîne de caractères constituée du numéro d'enregistrement et du titre du document.

# *Problème 2 -Exercice 1–La correction-*



```
public class Document {  
    static private int numeroSuivant = 0;  
    private int numero;  
    private String titre;  
    public Document(String titre) { this.numero = numeroSuivant;  
        numeroSuivant++; this.titre = titre; }  
public Document(String titre, int numero) {  
if (numero < numeroSuivant) { numero = numeroSuivant; }  
this.numero = numero; numeroSuivant = numero + 1;  
this.titre = titre; }  
    public int getNumero(){ return numero; }  
    public String getTitre(){ return titre; }  
public String toString() { return ("numero: "+numero+"  
titre: "+titre); }  
}
```

## *Problème 2 -Exercice 2–L'Enoncé-*



- On veut implémenter une liste de documents dans laquelle on veut pouvoir ajouter des documents petit à petit, et y accéder par leur numéro d'enregistrement. Mais plutôt que d'utiliser les structures de collections Java (ArrayList, Vector, etc.), on veut programmer soi-même une structure de données.
- Pour implémenter cette structure, on va créer une classe **ListeDeDocuments** avec un tableau de documents contenant les documents de la liste. Cependant, pour éviter d'avoir à augmenter la taille du tableau à chaque ajout, on va fixer initialement sa taille à 1 et doubler sa taille dès qu'on veut ajouter un document dont le numéro d'enregistrement est plus grand que la taille du tableau.

## *Problème 2 -Exercice 2–L’Enoncé-*



- Définir la classe **ListeDeDocuments** avec la propriété privée tableau et la méthode ajouter qui ajoute un document à la place indiquée par son numéro d'enregistrement, ainsi qu'une méthode getDocument qui prend en argument un numéro et renvoie le document de la liste ayant ce numéro (null s'il n'est pas dans la liste).

# *Problème 2 -Exercice 2–La correction-*



```
public class ListeDeDocuments {  
    private Document[] tableau;  
    public ListeDeDocuments() { tableau = new Document[1];}  
    public Document getDocument(int index) {  
        if (index >= tableau.length) { return null; }  
        return tableau[index]; }  
    public void ajouter(Document doc) {  
        if (tableau.length <= doc.getNumero()) {  
            Document[] t = new Document[tableau.length * 2];  
            for (int i = 0; i < tableau.length; ++i) { t[i] = tableau[i];}  
            tableau = t;}  
        tableau[doc.getNumero()] = doc;  
    }  
}
```

## *Problème 2 -Exercice 2–L’Enoncé-*



- Dans cette classe, redéfinir la méthode toString pour qu'elle renvoie les descriptions de tous les documents de la liste, séparées par un saut de ligne "`\n`".

# *Problème 2 -Exercice 2–La correction-*



```
public class ListeDeDocuments {  
    private Document[] tableau;  
    public ListeDeDocuments() { tableau = new Document[1];}  
    public Document getDocument(int index) {if (index >= tableau.length) { return null; }  
    return tableau[index];}  
    public void ajouter(Document doc) { if (tableau.length <= doc.getNumero()) {  
    Document[] t = new Document[tableau.length * 2];  
    for (int i = 0; i < tableau.length; ++i) { t[i] = tableau[i];} tableau = t;}  
    tableau[doc.getNumero()] = doc; }  
public String toString() {  
String resultat = "";  
for (int i = 0; i < tableau.length; ++i) {  
Document d = tableau[i];  
if (d != null) { resultat += d + "\n";}  
}return resultat;  
}
```