Amphi 5

Les énumérations et révision

Plan

- Les énumérations
- Modélisation en programmation d'objet/ Révision
- Examen

Les énumérations en Java

- Les énumération définissent des types qui contiennent un ensemble figé de constantes simplement désignées par les noms littéraux.
- Exemples:

```
public enum Jour  { LUNDI, MARDI, ...}

public enum Saison  { PRINTEMPS, ETE, ...}

public enum Couleur  { COEUR, CARREAU, ...}
```

- A partir de là, on peut définir des variables qui relèvent d'une énumération

```
Jour j =  LUNDI;          Couleur c =  CARREAU;
```

Les énumérations en Java

- On peut utiliser ces variables dans des expressions Java
  - affectation `Jour j = Jour.LUNDI;`
  - comparaison `if (j == Jour.LUNDI)`
  - comparaisons multiples

```
switch (couleur){
  case COEUR: ...
  case CARREAU: ...
  ...
}
```

Les énumérations en Java

- Exemple avec la définition de l'énumération dans un fichier à part
  - Fichier Couleur.java

```
public enum Couleur  { COEUR, CARREAU, PIQUE, TREFLE}
```

  - Fichier Carte.java

```
public class Carte{
  private Couleur couleur;
  private int rang;
  private int force;

  public Carte (Couleur c, int r){
     this.couleur = c;
     this.rang = (r >= 1 && r <= 13 ? r : 0);
     this.force = r;
  }
}
```

Les énumérations en Java

- Exemple avec la définition de l'énumération dans un même fichier

  - Fichier Carte.java

```
public enum Couleur  { COEUR, CARREAU, PIQUE, TREFLE}

public class Carte{
  private Couleur couleur;
  private int rang;
  private int force;

  public Carte (Couleur c, int r){
     this.couleur = c;
     this.rang = (r >= 1 && r <= 13 ? r : 0);
     this.force = r;
  }
}
```

Ce qu'il faut retenir

Structure des classes

```
class X {
```

attributs

constructeurs

méthodes (d'instance)

```
}
```

Ce qu'il faut retenir

Structure des classes

```
class X {
```

attributs

constructeurs

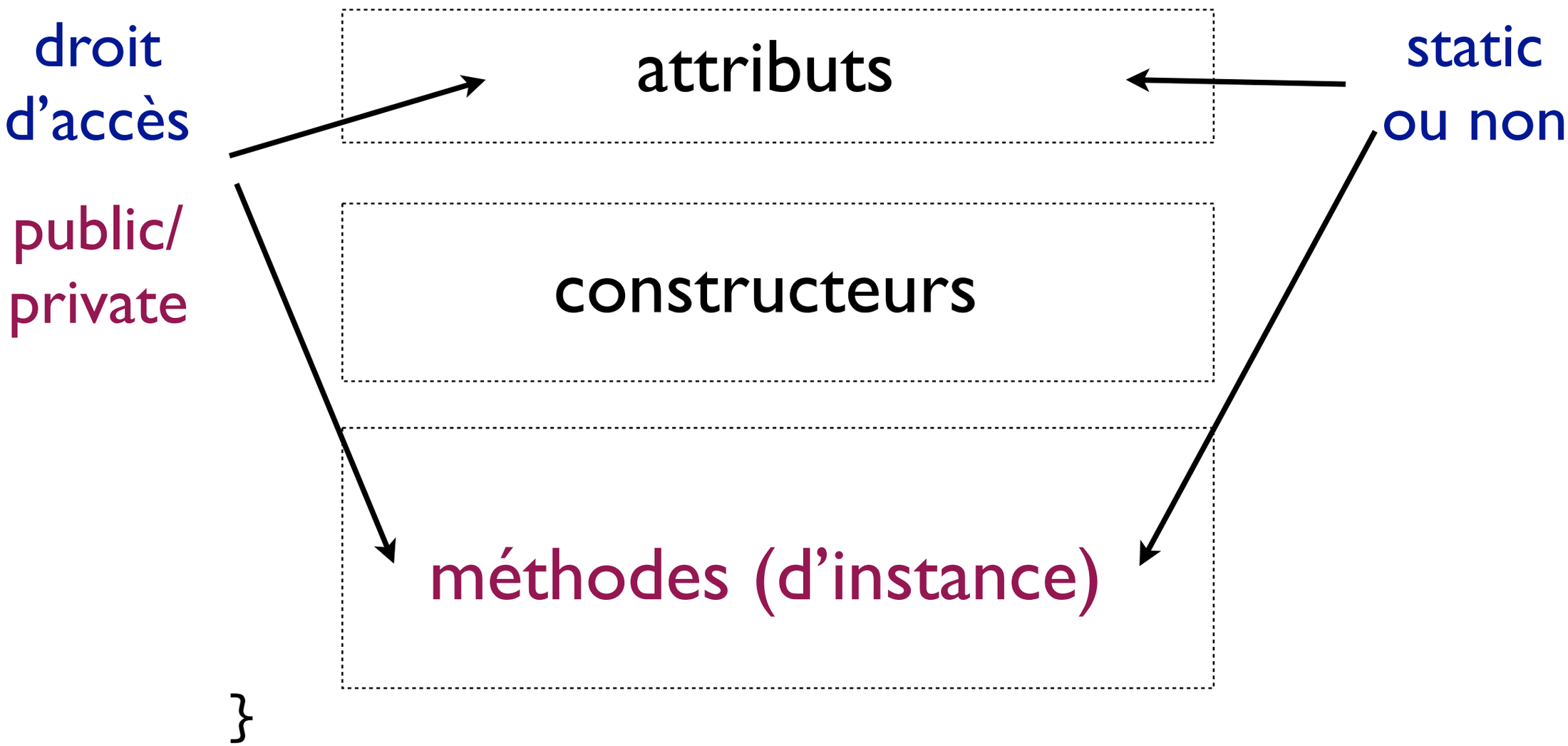
méthodes (d'instance)

static ou non

```
}
```

Ce qu'il faut retenir

Structure des classes

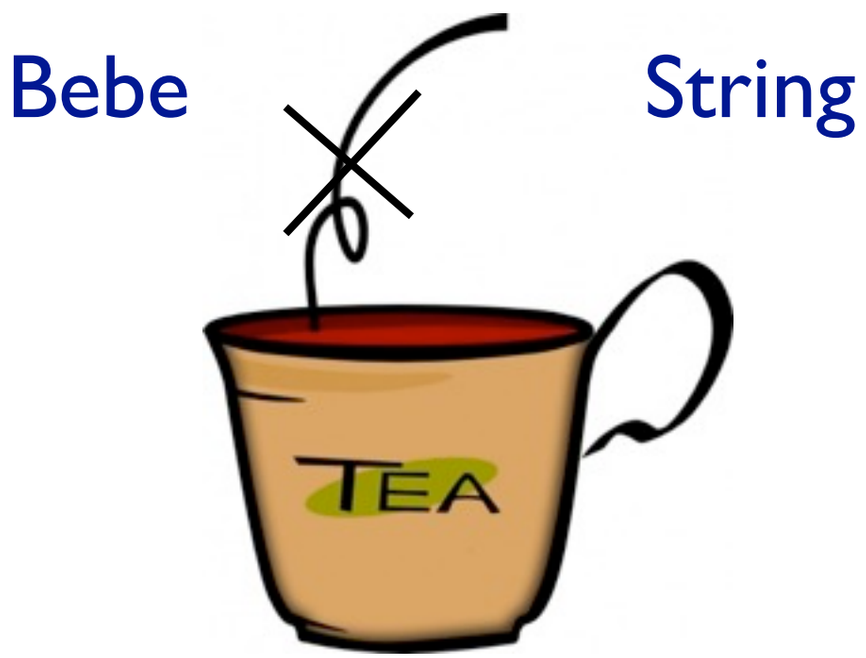


```java
public class Book {

    private String title;
    private boolean borrowed;
    public static int nombre;

    public Book(String bookTitle) {
     this.title = bookTitle;
      this.nombre ++;
    }

    public void borrowed() {
     this.borrowed = true;
    }

    public void returned() {
     this.borrowed = false;
    }

    public boolean isBorrowed() {
     return this.borrowed;
    }
}
```

Ce qu'il faut retenir

Références/Stockage en Java

- Les objets sont "trop" grands
  - stocker ailleurs
  - Variable stocke un numéro (adresse) pour localiser l'objet



Ce qu'il faut retenir

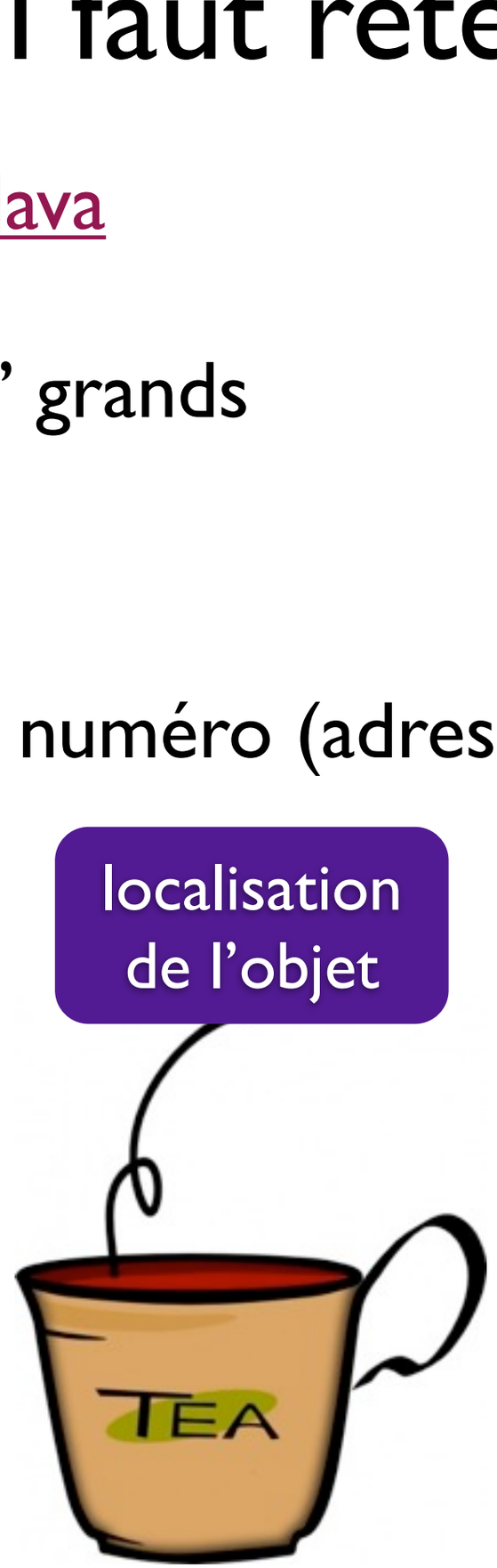
Références/Stockage en Java

- Les objets sont "trop" grands
  - stocker ailleurs
  - Variable stocke un numéro (adresse) pour localiser l'objet

Ce qu'il faut retenir

Références/Stockage en Java

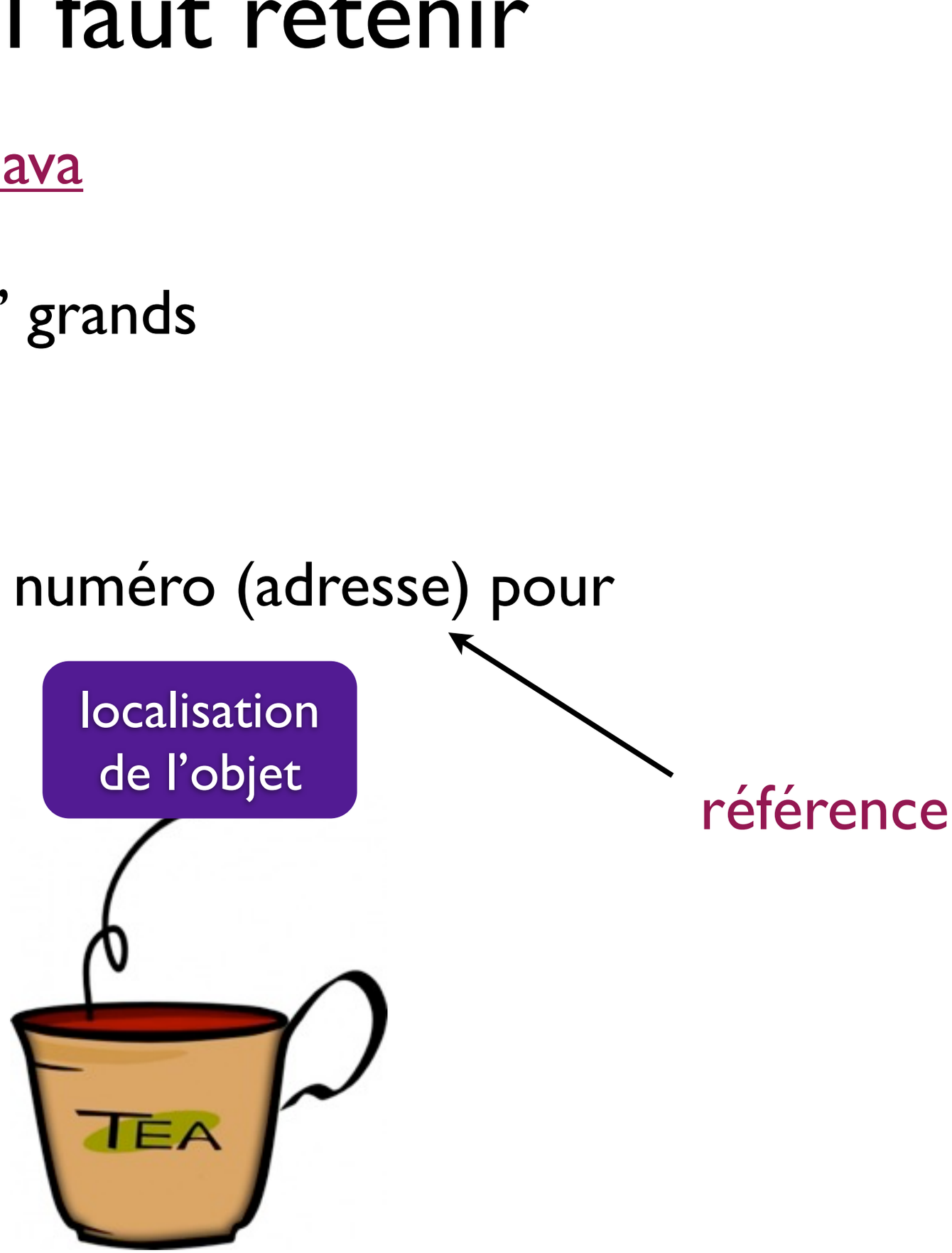
- Les objets sont "trop" grands
  - stocker ailleurs
  - Variable stocke un numéro (adresse) pour localiser l'objet



Ce qu'il faut retenir

Références/Stockage en Java

- Les objets sont "trop" grands
  - stocker ailleurs
  - Variable stocke un numéro (adresse) pour localiser l'objet



Références

- “==” compare les références

```
Bebe b1 = new Bebe ("Bryan", true, 5.0);
Bebe b2 = new Bebe ("Bryan", true, 5.0);
```

b1 == b2?

Références

- “==” compare les références

```
Bebe b1 = new Bebe ("Bryan", true, 5.0);
Bebe b2 = new Bebe ("Bryan", true, 5.0);
```

b1 == b2?

Non

référence b1

référence b2

b1

b2

Références

```
Bebe b1 = new Bebe ("Bryan", true, 5.0);
```

référence b1

b1

nom = Bryan

Références

```
Bebe b1 = new Bebe ("Bryan", true, 5.0);
b1.nom = Antoine;
```

référence b1

b1

nom = Bryan

Références

```
Bebe b1 = new Bebe ("Bryan", true, 5.0);
b1.nom = Antoine;
```

référence b1

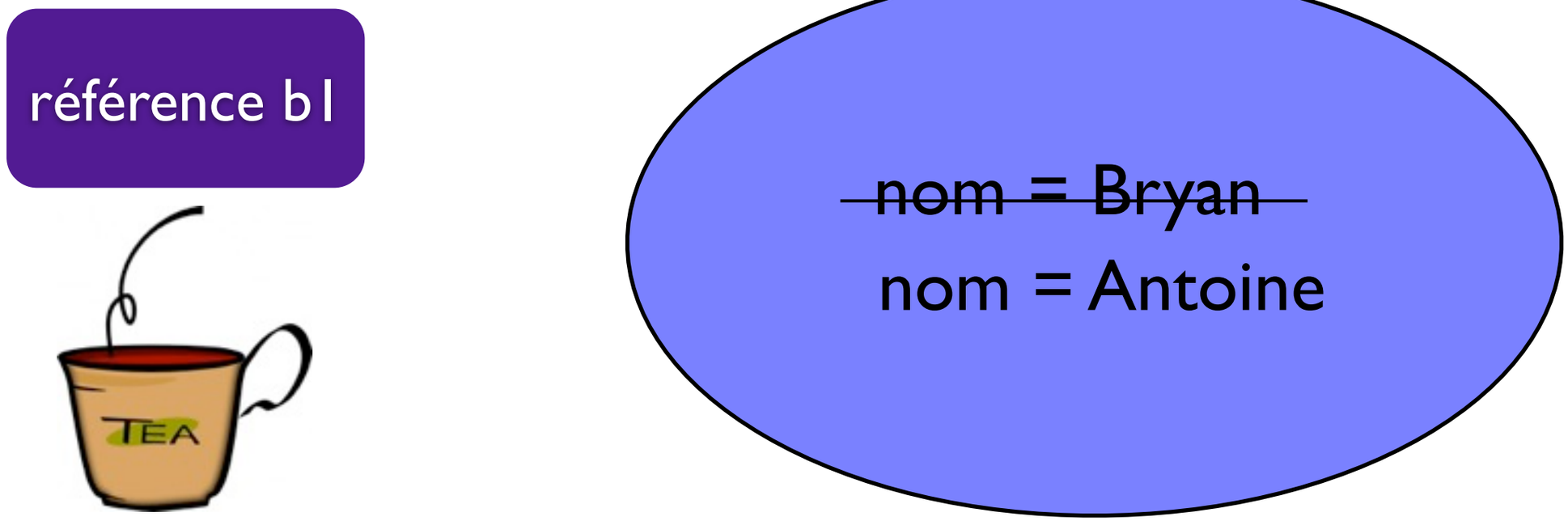
b1

~~nom = Bryan~~

nom = Antoine

Références

```
Bebe b1 = new Bebe ("Bryan", true, 5.0);
b1.nom = Antoine;
```



Utiliser “=” pour mettre à jour les attributs

Références

Utiliser "=" pour mettre à jour les références

```
b1 = b2;
```

référence b1

référence b2

b1

b2

nom = Bryan

objet b1

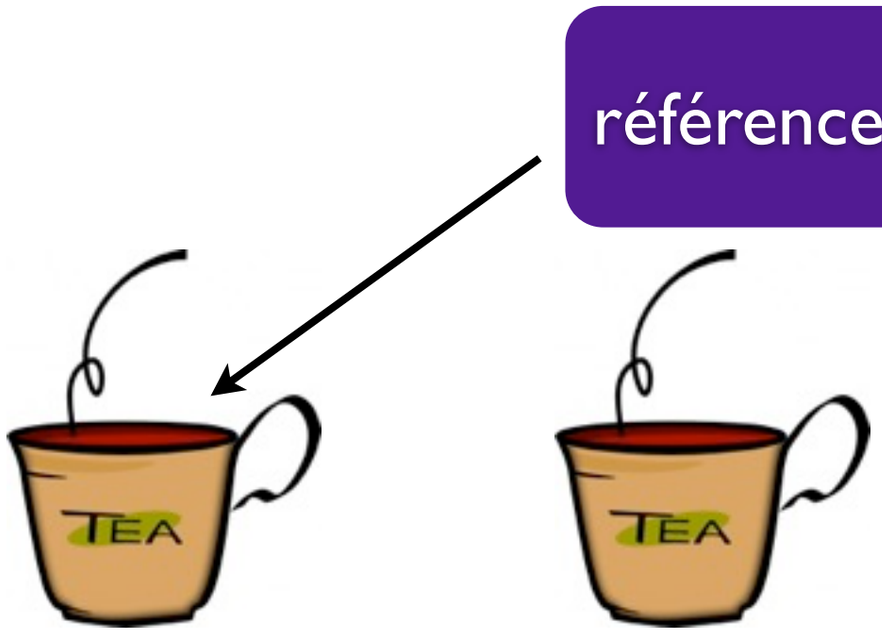
nom = Antoine

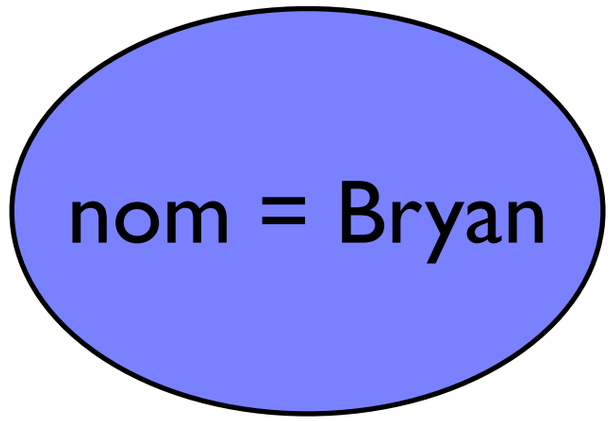
objet b2

Références

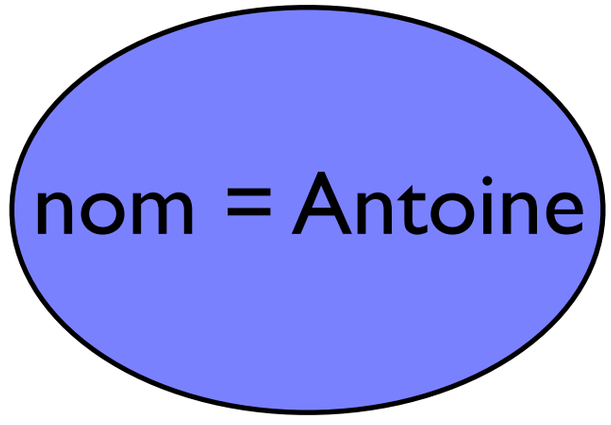
Utiliser "=" pour mettre à jour les références

```
b1 = b2;
```





objet b1

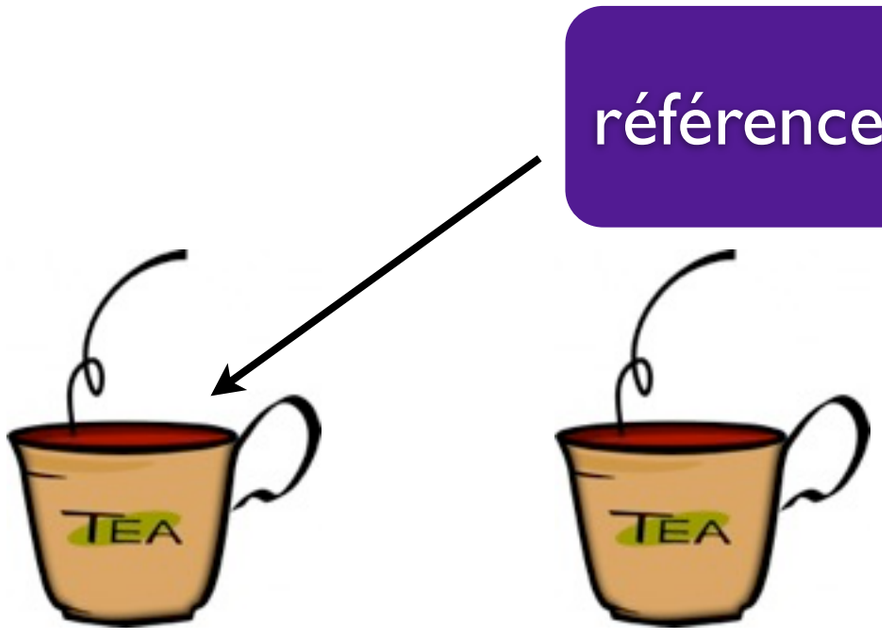


objet b2

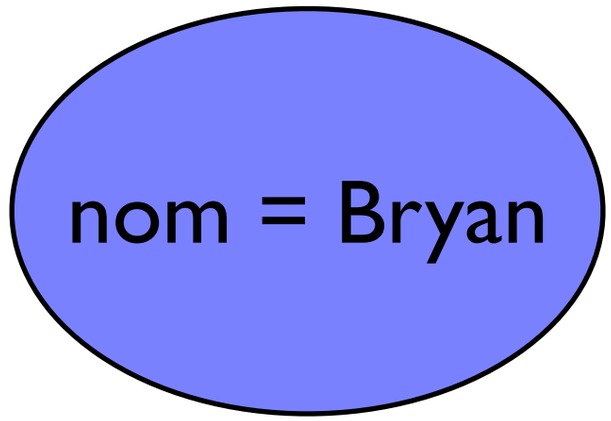
Références

Utiliser "=" pour mettre à jour les références

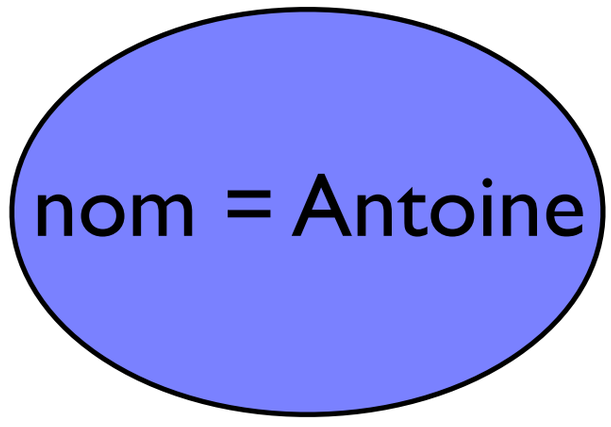
```
b1 = b2;
```



b1 b2



objet b1



objet b2

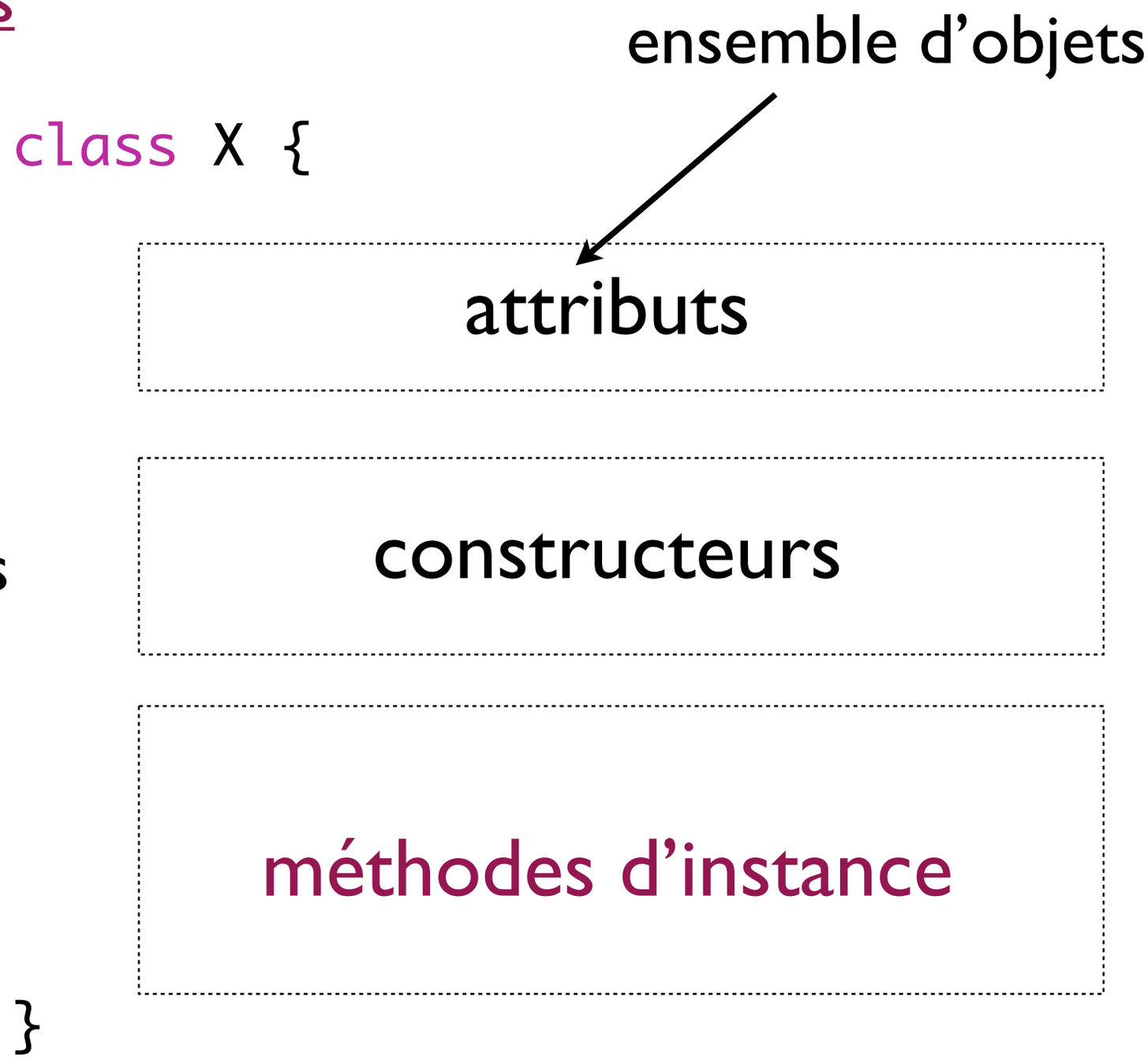
```
b1.nom = ?
```

Ce qu'il faut retenir

- Classes ensemblistes

  - utilisation des tableaux

  - utilisation des classes génériques de Java (avec ses méthodes)



```java
import java.util.ArrayList;

public class Library {

    static int openingHours = 8;
    static int closingHours = 17;
    ArrayList<Book> biblio ;
    String address;

    public Library(String addr){
     this.biblio = new ArrayList<Book>();
     this.address = addr;
    }

    public boolean addBook(Book b){
     return this.biblio.add(b);
    }


}
```

ArrayList<E> & méthodes utiles

- par rapport à l'ensemble:

"E" : Livre, Domino, etc

```
int size()              retourne la taille de l'ensemble
boolean isEmpty()       teste si l'ensemble est vide
Object clone()          retourne une copie de l'objet ArrayList<E>
```

- par rapport à la lecture dans l'ensemble:

```
E get(int index)        retourne l'élément d'indice index
boolean contains (E o)          teste si l'ensemble contient o
int indexOf (E o)               retourne l'indice de l'élément o ou -1 s'il n'existe pas
```

- par rapport à l'écriture dans l'ensemble

```
boolean add(E e)        ajouter un élément e à la fin du tableau (retourner true)
void add(int index, E e)    insérer l'élément e à l'indice index
E set(int index, E e)   remplace l'élément d'indice index par e et retourner l'ancien
E remove(int index)     supprimer l'élément d'indice index et le retourne
E remove(E e)           supprimer e et retourn true s'il existait, false sinon
```

Ce qu'il faut retenir

- Paquetages
  - Chaque classe appartient à un paquetage.
  - Les classes dans un même paquetage servent aux mêmes buts
  - Les classes dans un autre paquetage doivent être importées (s'il y a besoin d'utilisation)
  - Les classes dans un même paquetages se trouve automatiquement (pas besoin d'importer)
  - Toutes les classes, par défaut, importent (implicitement) les classes dans java.lang, exemple: java.lang.String, java.lang.System

Ce qu'il faut retenir

- Utilisation de Java API

  - Java API: Java Application Programming Interface
  - Java incluit beaucoup de paquetages/classes (par exemple, ArrayList, ...)
  - Utiliser les classes dans Java API pour économiser/ optimiser le travail
  - http://docs.oracle.com/javase/6/docs/api (le lien se trouve sur le site du cours)

Sets

- Ressemble à ArrayList, mais
  - seulement un copie pour chaque objet
  - pas d'indice de tableau
- Propriétés:
  - ajouter un objet dans un ensemble (méthode add)
  - supprimer un objet dans un ensemble (méthode remove)
  - vérifier si un objet donné appartient à un ensemble

TreeSet: ordonnés (petit à grand, alphabet)
HashSet: non-ordonnés (pseudo-random)

TreeSet<E> & Démo

```
import java.util.TreeSet;

class TreeSetExemple{
  public static void main   (String [] args){
     TreeSet<String> strings = new
TreeSet<String>();

     strings.add("Even");
     strings.add("Eugene");
     strings.add("Adam");

     System.out.println(strings.size());
     System.out.println(strings.first);
     System.out.println(strings.last);

     strings.remove("Eugene");

     for (String s: strings) {
        System.out.println(s);
     }
  }
}
```

Maps

- Stocker un couple (clé, valeur) d'objets

- Donné un clé, chercher la valeur.

  Exemple: (nom, mail); (étudiant, numéro d'étudiant)

TreeMap: ordonnés (petit à grand, alphabet)
HashMap: non-ordonnés (pseudo-random)

HashMap<E> & Démo

```
import java.util.*; // import some packages for
                    // HashMap

class HashMapExemple{
  public static void main  (String [] args){
     HashMap<String,String> strings = new
          HashMap<String,String>();

     strings.put("Even", "even@bio.evry");
     strings.add("Eugene", "eugene@math.evry");
     strings.add("Adam", "adam@info.evry");

     System.out.println
          (strings.size());
     strings.remove("Even");
     System.out.println
       (strings.get("Eugene"));
}
```

```
for (String s: strings.keySet()) {
   System.out.println(s);
}

for (String s: strings.values()) {
   System.out.println(s);
}

}
```

Examen

- Même types de questions comme le DS
  - Attributs, constructeurs, méthodes d'instance
  - Classes ensemblistes
- Plus
  - questions de cours (sur les références, static ...)
  - utilisation des paquetages
  - style d'écriture

Examen

- Même types de questions comme le DS
  - Attributs, constructeurs, méthodes d'instance
  - Classes ensemblistes
- Plus
  - questions de cours (sur les références, static ...)
  - utilisation des paquetages
  - style d'écriture

Bon courage!