Amphi 2

Méthodes d'instance simples

Remarques

- Nom de chaque classe est en Majuscule

- 1 classe = 1 fichier

- Une classe a une méthode main: la class peut-être exécutée

Attributs + Constructeurs

```
class Domino {

  int  m1; // marque 1
  int  m2; // marque 2

  public Domino(int vm1, int vm2) {

     this.m1 = vm1; // marque 1
     this.m2 = vm2; // marque 2
  }
}
```

attributs

constructeur

new <Constructeur> (Paramètres)

```
Domino d = new Domino (2,3);
```

Attributs + Constructeurs

```
class Bebe {
    String name;
    double poids;
    boolean estGarçon;

    public Bebe (String s, double p, boolean b){
        this.name = n;
        this.poids = p;
        this.estGarçon = b;
    }
}
```

attributs

constructeur

```
Bebe b = new Bebe (Antoine, 5.0, true);
```

Orienté-Objet

```
class X {
```

attributs

constructeurs

méthodes d'instance

```
}
```

Orienté-Objet

```
class X {
```

attributs

constructeurs

méthodes d'instance

```
}
```

permet d'inclure dans les objets les méthodes "responsables" d'agir sur leurs attributs

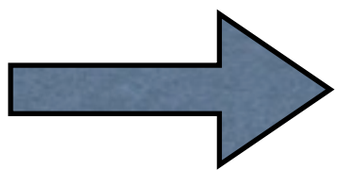
Example de TD I

```
class MainDominos {
            ...
  public static boolean estDouble(Domino d){
     return d.m1 == d.m2;
  }
}
```

Example de TD I

```
class MainDominos {
            ...
  public static boolean estDouble(Domino d){
     return d.m1 == d.m2;
  }
}
```

- Cette méthode n'est pas à sa place en Orienté-Objet
- Dans ce paradigm, on préfèrera penser que le domino est lui-même le mieux placé pour savoir s'il est ou non un double



transférer la méthode vers la class Domino

Méthodes d'instance

```
class MainDominos {
            ...
  public static boolean
     estDouble(Domino d){
      return d.m1 == d.m2;
  }
            ...
}
```

```
class Domino {
  int m1; // marque 1
  int m2; // marque 2

  public Domino(int vm1, int vm2){
      this.m1 = vm1; // marque 1
      this.m2 = vm2; // marque 2
  }

  public boolean estDouble(){
      return this.m1 == this.m2;
  }
}
```

Méthodes d'instance

```
class MainDominos {
            ...
  public static boolean
     estDouble(Domino d){
      return d.m1 == d.m2;
  }
            ...
}
```

```
class Domino {
  int m1; // marque 1
  int m2; // marque 2

  public Domino(int vm1, int vm2){
      this.m1 = vm1; // marque 1
      this.m2 = vm2; // marque 2
  }

  public boolean estDouble(){
      return this.m1 == this.m2;
  }
}
```

- Le paramètre d a disparu: la méthode réfère maintenant au domino en cours d'exécution
- Le mot static signifiant "réfère à la classe", il ne doit pas être utilisé ici.

Méthodes d'instance

Comment appeler une méthode d'instance?

objet.nomMéthode (Paramètres)

```
class Domino {
  int m1; // marque 1
  int m2; // marque 2

  public Domino(int vm1, int vm2){
      this.m1 = vm1; // marque 1
      this.m2 = vm2; // marque 2
  }

  public static boolean estDouble(){
       return this.m1 == this.m2;
  }
}
```

Méthodes d'instance

Comment appeler une méthode d'instance?

objet.nomMéthode (Paramètres)

```
class MainDominos {
            ...
public static void test1()
{
   if (d1.estDouble()) {
            ...
}
}
            ...
}
```

```
class Domino {
  int m1; // marque 1
  int m2; // marque 2

  public Domino(int vm1, int vm2){
     this.m1 = vm1; // marque 1
     this.m2 = vm2; // marque 2
  }

  public static boolean estDouble(){
      return this.m1 == this.m2;
  }
}
```

Méthodes d'instance

Comment appeler une méthode d'instance?

objet.nomMéthode (Paramètres)

```
class MainDominos {
            ...
public static void test1()
{
    if (d1.estDouble()) {
            ...
}
...
}
```

invocation de méthode

mis à jour de "this"

```
class Domino {
  int m1; // marque 1
  int m2; // marque 2

  public Domino(int vm1, int vm2){
      this.m1 = vm1; // marque 1
      this.m2 = vm2; // marque 2
  }

  public static boolean estDouble(){
       return this.m1 == this.m2;
  }
}
```

exécution de méthode

Méthodes d'instance

```
class MainDominos {
            ...
public static void test1(){
   if (d1.estDouble()) {
            ...
}

System.out.println
            (d1.domino2String());

System.out.println
            (d1.valeur());
}
            ...
}
```

invocations de méthodes d'instance

```
class Domino {
  int m1; // marque 1
  int m2; // marque 2

  public boolean estDouble(){
      return this.m1 == this.m2;
  }
  public String domino2String(){
      return " (" + this.m1 + ","
                  + this.m2 + ")";
  }
  public boolean valeur(){
      return this.m1 + this.m2;
  }
}
```

méthodes d'instance de la classe

Classes & Objets

```
class X {
```

attributs (variables d'instance)

constructeurs

méthodes d'instance

```
}
```

- La manière que les objets doivent être conçus
- Un objet est vu comme une entité qui encapsule un état interne (attributs) et qui fournit des services (méthodes d'instance)

Example Bebe

```
public class Bebe {
  String nom;
  boolean estGarcon;
  double poids;

  Bebe (String n, boolean b, double p){
      nom = n; estGarcon = b; poids = p;
  }

  void ditBonjour(){
      System.out.println("Bonjour, mon nom est " + name);
  }

  void grandir(double poids_plus){
      if (poids_plus > 0){
         this.poids = this.poids + poids_plus;
      }
  }
}
```

Example Bebe

```
public class X {
   Bebe b1 = new Bebe
      ("Bryan", true, 5.0);

   System.out.println(b1.nom);

   System.out.println(b1.nom);
}
```

accéder aux attributs

OBJET.NOM_ATTRIBUT

```
public class X {
      Bebe b1 = new Bebe
         ("Bryan", true, 5.0);

   b1.ditBonjour();

   b1.grandir(1.0);
}
```

appeler une méthode d'instance

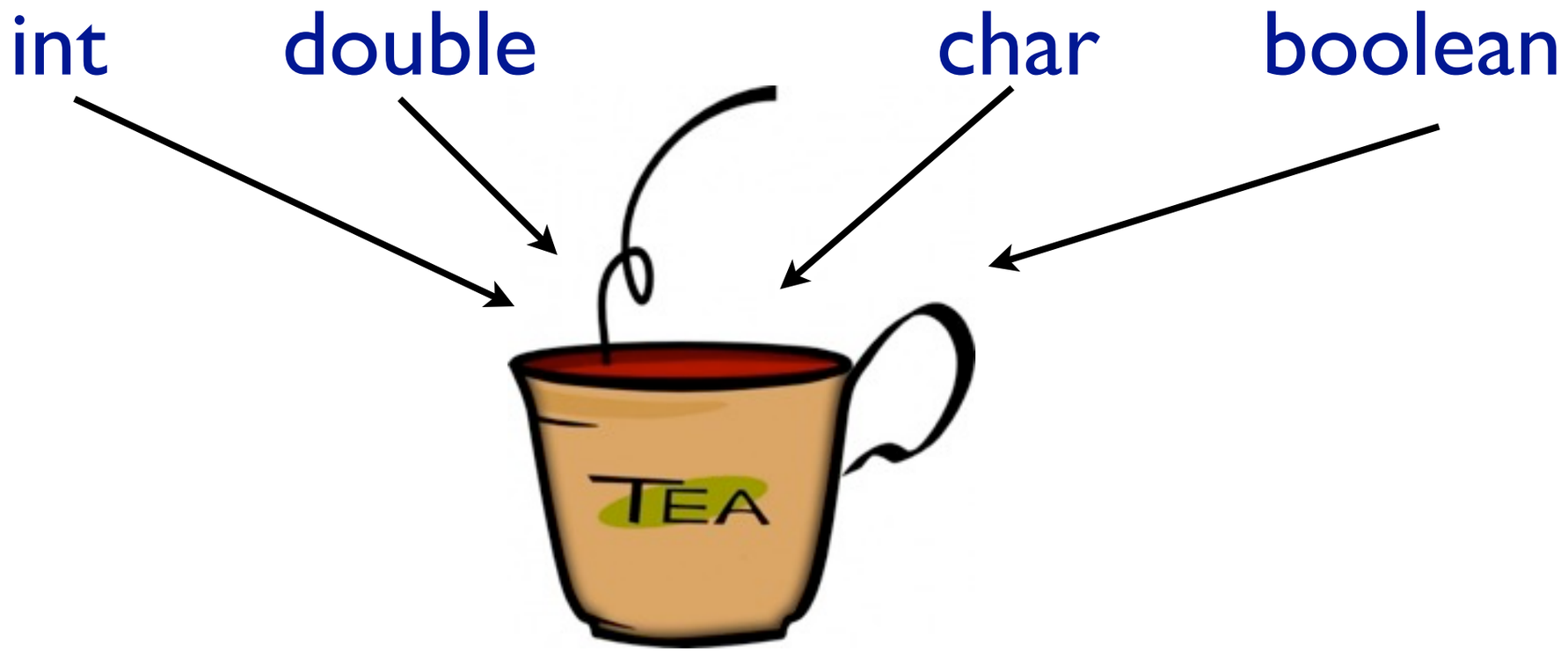
OBJET.NOM_METHODE
(ARGUMENT)

Références vs valeurs

- Types primitives sont les types basiques
  - int, long, double, boolean, char, short, byte, float
  - Les valeurs sont stockées dans les variables
- Types de références sont les tableaux ou les objets
  - String, int[], Bebe ...

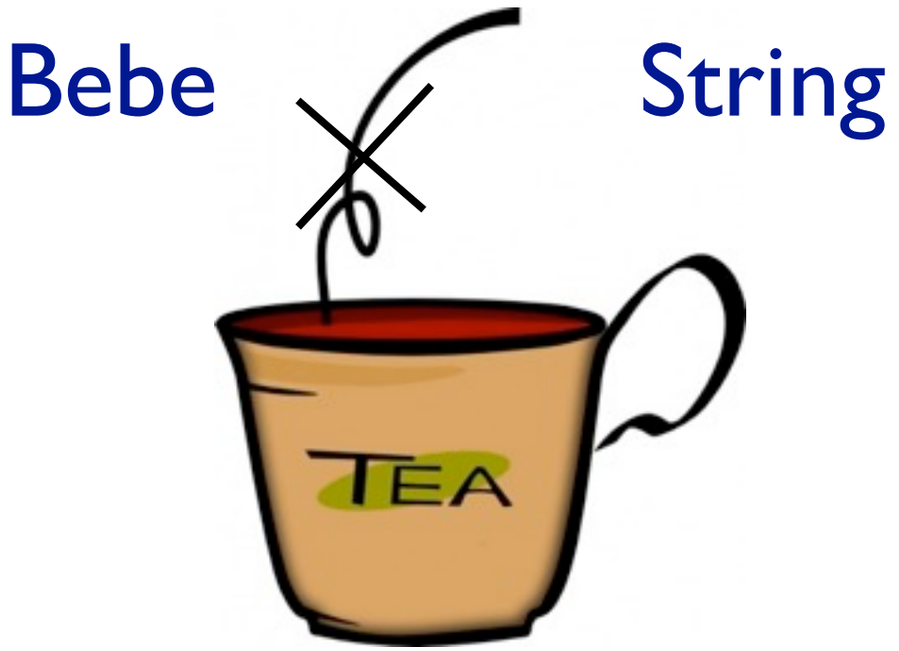
Stockage de Java

- Variables ressemblent aux tasses.
- Les primitifs sont suffisament petits



Stockage de Java

- Les objets sont “trop” grands
  - stocker ailleurs
  - Variable stocke un numéro (adresse) pour localiser l’objet



Stockage de Java

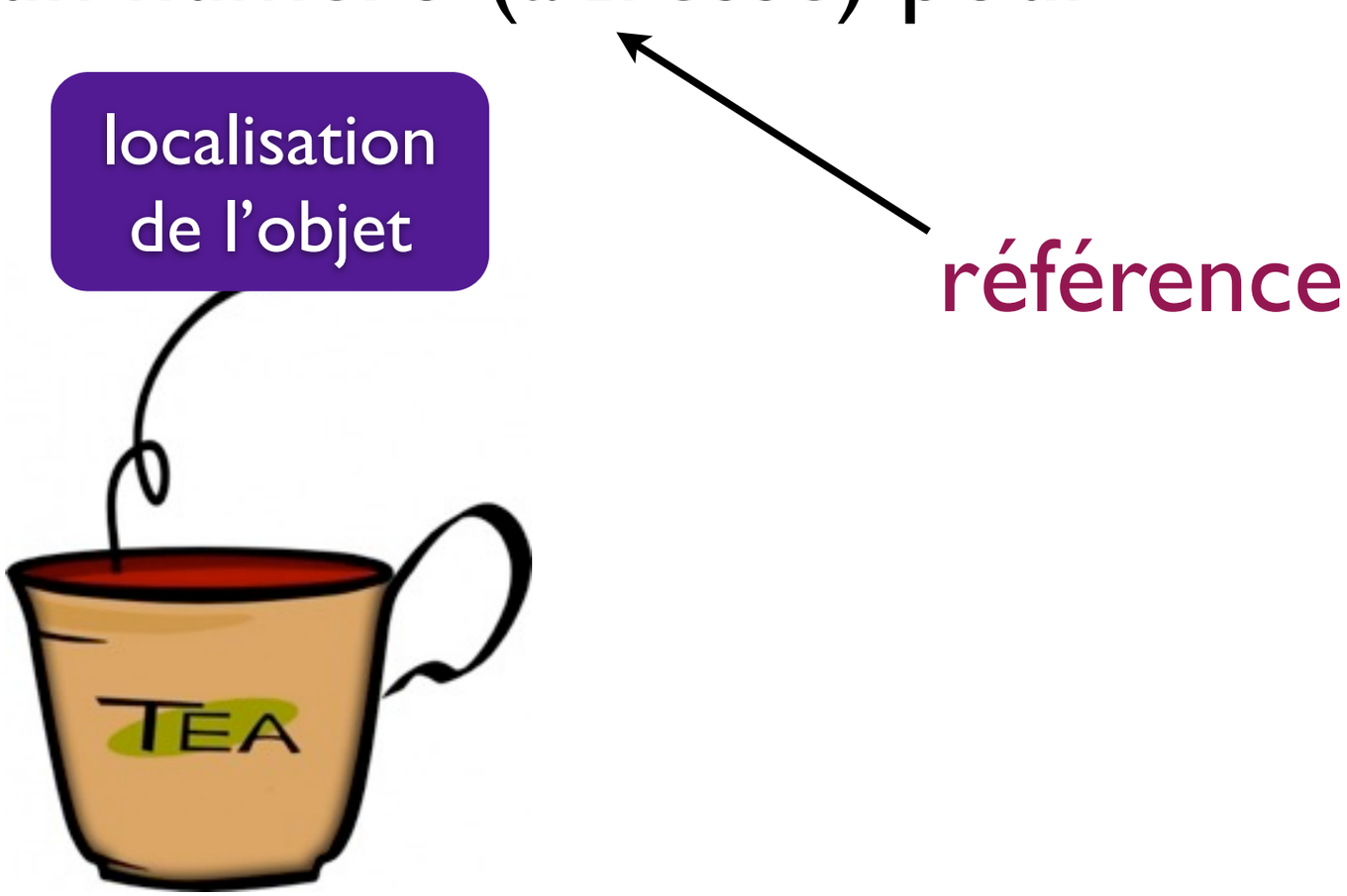
- Les objets sont “trop” grands
  - stocker ailleurs
  - Variable stocke un numéro (adresse) pour localiser l’objet

Stockage de Java

- Les objets sont “trop” grands
  - stocker ailleurs
  - Variable stocke un numéro (adresse) pour localiser l’objet

Stockage de 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?

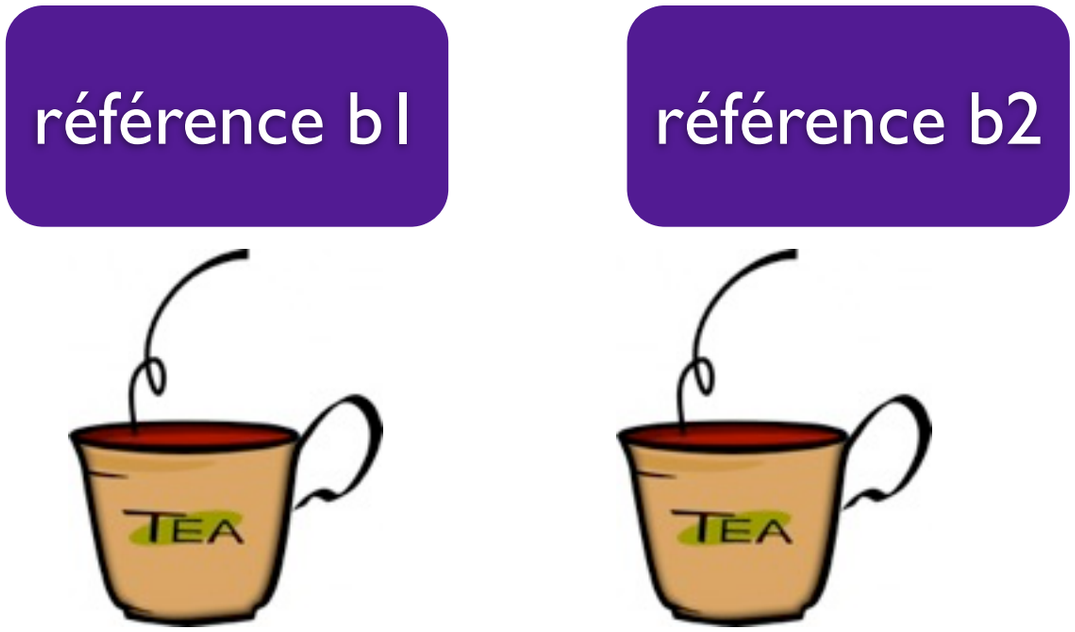
Références

- “==” compare les références

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

b1 == b2?

Non



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;
```

référence b1

b1

~~nom = Bryan~~

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

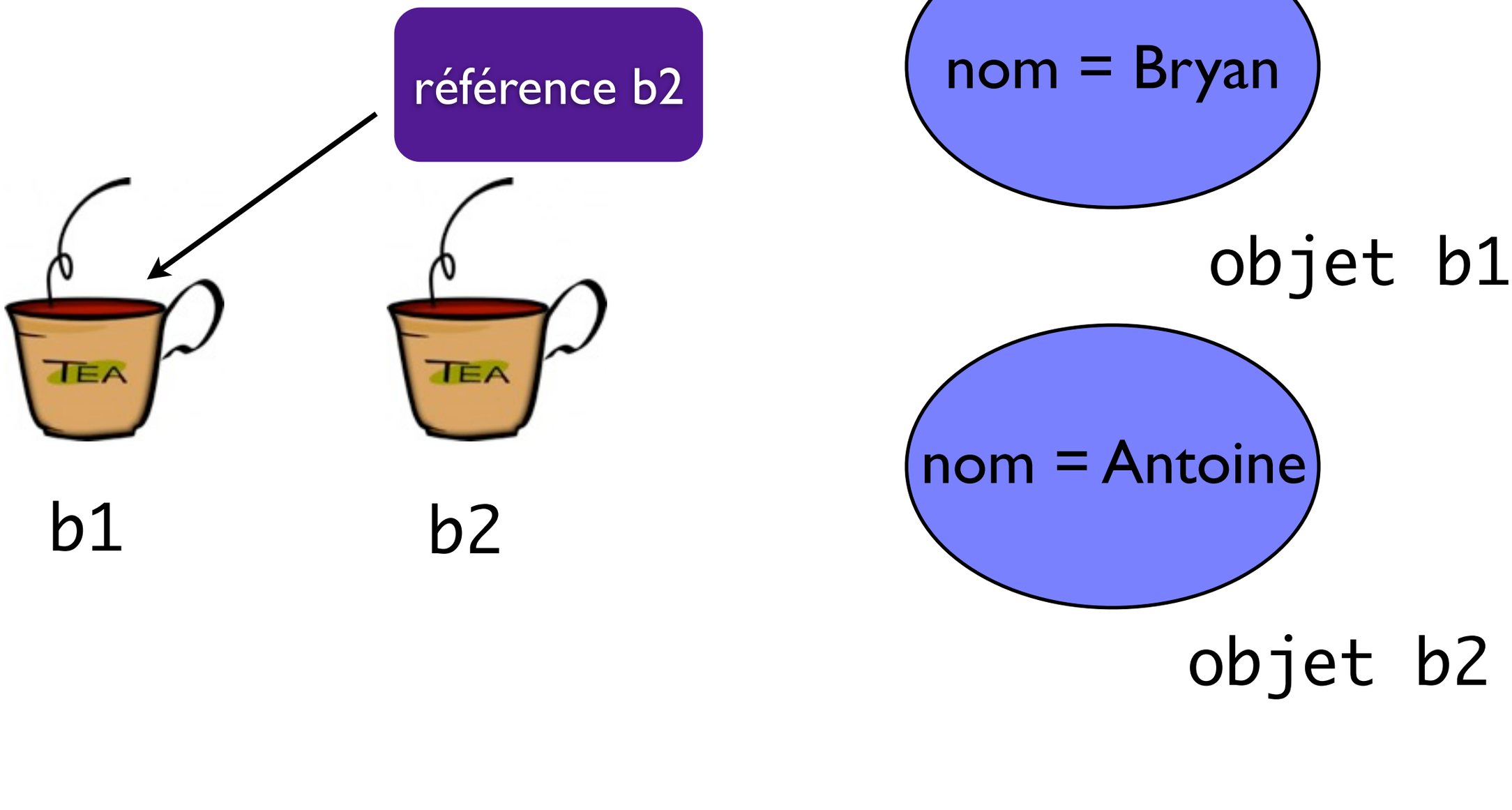
nom = Antoine

objet b2

Références

Utiliser "=" pour mettre à jour les références

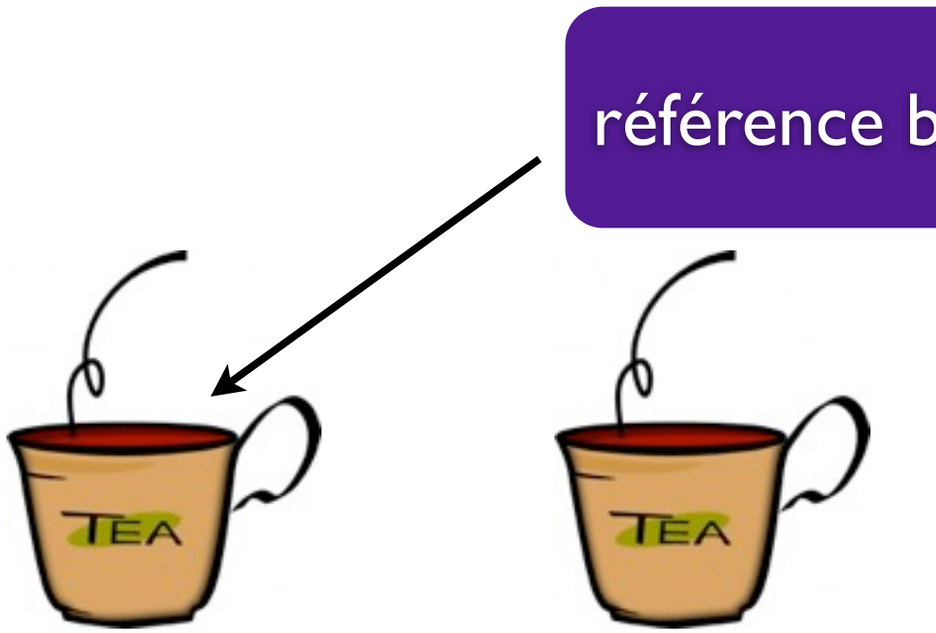
```
b1 = b2;
```



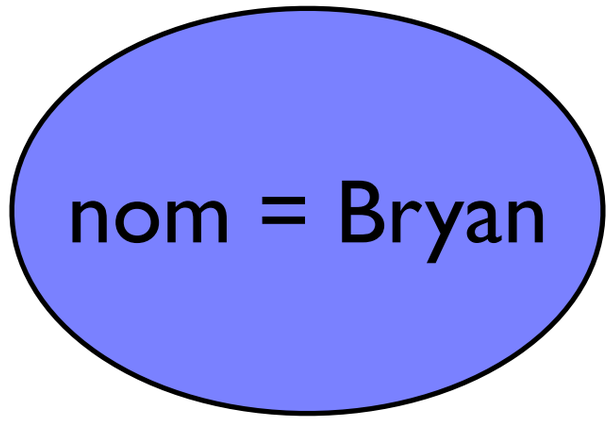
Références

Utiliser "=" pour mettre à jour les références

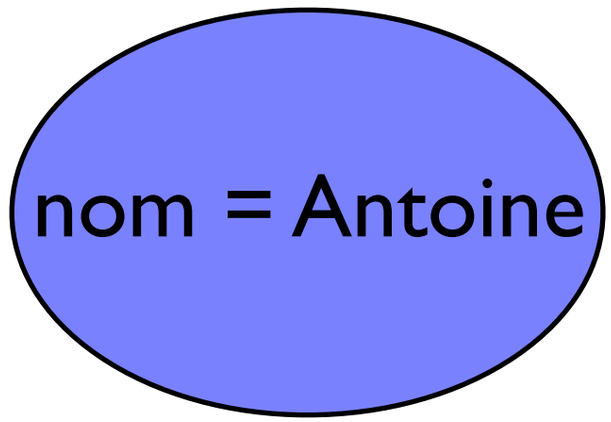
```
b1 = b2;
```



b1 b2



objet b1



objet b2

```
b1.nom = ?
```

Remarques

- TD: groupe Eco1 + Bio: mardi12 Fev 13h15

  Autres groupes: Jeudi 14 Fev, 8h00

- Dominos
- Livres
- Notes:

  max{1/3 DS + 2/3 Examen, Examen}