



Modèles de mémoire en Java CardTM Introduction du concept de pré-persistance

Serge Chaumette, **Damien Sauveron**
chaumett@labri.fr, sauveron@labri.fr



Plan

La carte à puce

La technologie Java Card

Les objets persistants et transients

Introduction du concept de pré-persistance

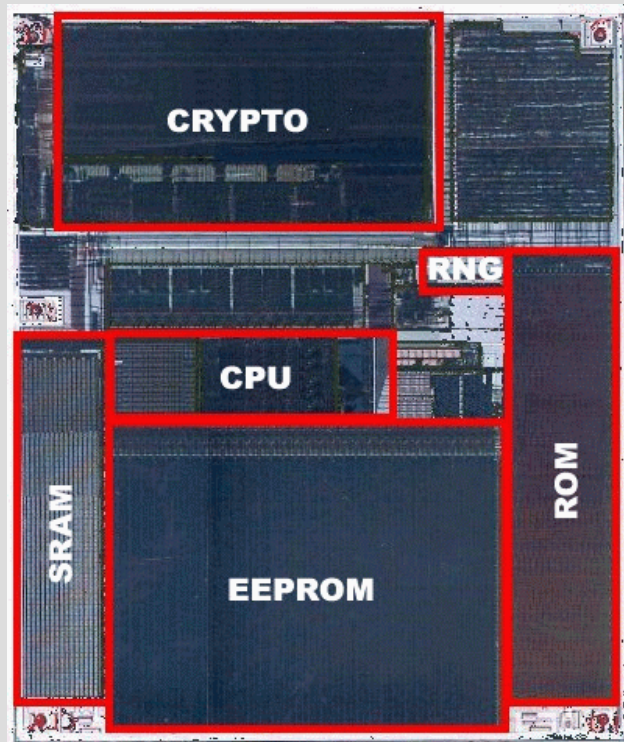
Exemple de code en Java

Exemple de code en Java Card **sans** pré-persistance

Exemple de code en Java Card **avec** pré-persistance

Conclusion et perspectives

La carte à puce : un périphérique de nature persistante



Les mémoires :

Permanente : ROM

- 32 à 64 Ko
- Card Operating System + données permanentes

Volatile : RAM

- 1 à 2 Ko
- très rapide
- *mémoire de travail* (pile d'appels, etc.)

Persistante : EEPROM (ou FeRAM ou Flash)

- 24 à 64 Ko
- lente (200 fois plus que la RAM)
- *mémoire de stockage*
- durée de vie limitée à 100 000 cycles d'écriture

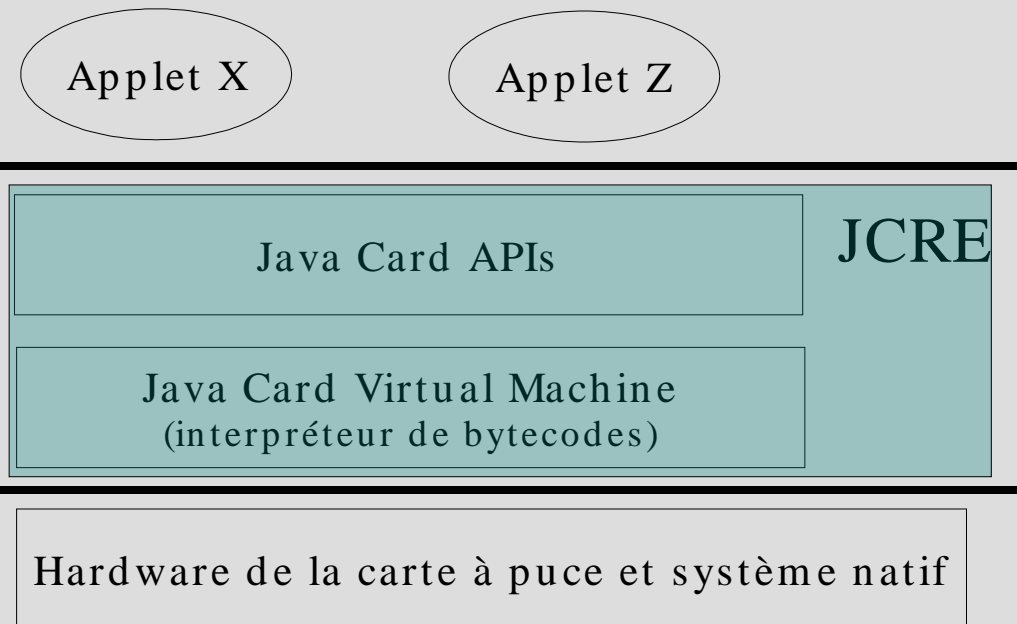
La technologie Java Card

Difficultés pour adapter Java aux cartes :

- faible capacité mémoire
- périphérique de nature persistante

Solutions :

- limitation au strict nécessaire sur la carte (types simples, tableaux de types simples, ...)
- introduction des concepts de persistance et de transience



Objets persistants

Un objet persistant est stocké en mémoire persistante : entête + champs.

Spécifications :

Un objet persistant **devra** être créé lorsque :

- la méthode `Applet.register` est appelée (enregistrement d'une application) ;
- une référence à l'objet est stockée dans un champ d'un objet persistant ou dans celui statique d'une classe.

Quelques caractéristiques :

- il a été créé par le bytecode `new` avant de devenir persistant ;
- il conserve ses valeurs lors des sessions avec le lecteur.

Objets transients

Un objet transient est un objet dont le contenu est temporaire :

- l'entête persiste et l'espace alloué au contenu également.



Un objet transient est associé à un événement dont l'occurrence déclenche la réinitialisation de ses champs :

- CLEAR_ON_DESELECT ;
- CLEAR_ON_RESET.

Les seuls types d'objets transients jusque dans les spécifications JC 2.2.1 sont les tableaux de types simples.

Quelques caractéristiques :

- il a été créé par makeTransientBooleanArray (et consœurs de la classe JCSysyem) ;
- il est réinitialisé lors de l'occurrence d'un événement.

Introduction du concept de pré-persistance

Spécifications :

- un objet est créé lors de :
 - l'exécution d'un bytecode `new`, `newarray` et `newarray` ;
 - l'invocation d'une méthode de l'API `makeTransientXXXArray`.
- le développeur **rendra** les objets persistants lorsqu'une référence à l'objet est stockée dans un champ d'un objet persistant ou dans celui d'une classe.

Quelle est la nature des objets entre le moment de leur création et celui de leur affectation à un champ d'un objet persistant ?

Nous désignerons par espace **pré-persistant** l'espace où sont placés les objets lors de leur création par l'exécution d'un bytecode `new`, `newarray` et `newarray`.

Sa localisation reste à définir mais **dans la suite** nous le considérerons **en RAM**

Attention : **pré-persistant** n'implique pas que l'objet deviendra nécessairement persistant.

Exemple de code en Java

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() { ----- ICI  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}  
  
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

Instance de A
s1 =
o1 =

RAM

Pile de frames

var. locales

pile d'opérandes

Exemple de code en Java

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0      ←----- ICI  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}  
  
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

Instance de A
s1 =
o1 =

RAM

Pile de frames

var. locales

pile d'opérandes

this
0x0000

Exemple de code en Java

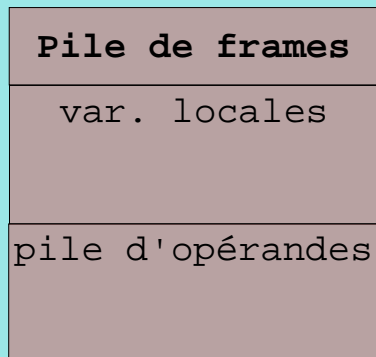
```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1 <----- ICI  
    aload_0  
    new Object  
    putfield_a o1  
}  
  
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

Instance de A
s1 = 0x0000
o1 =

RAM



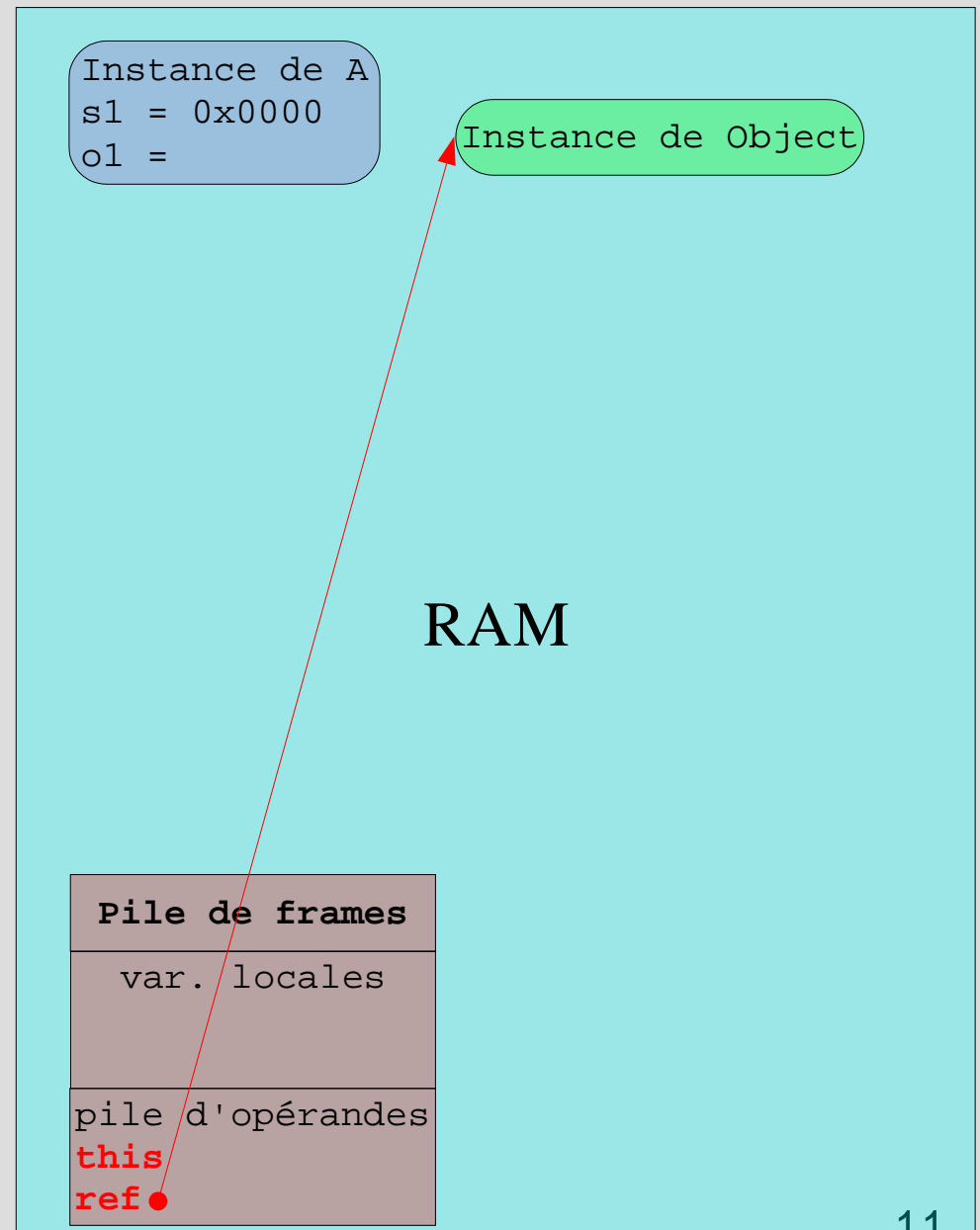
Exemple de code en Java

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object ←----- ICI  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

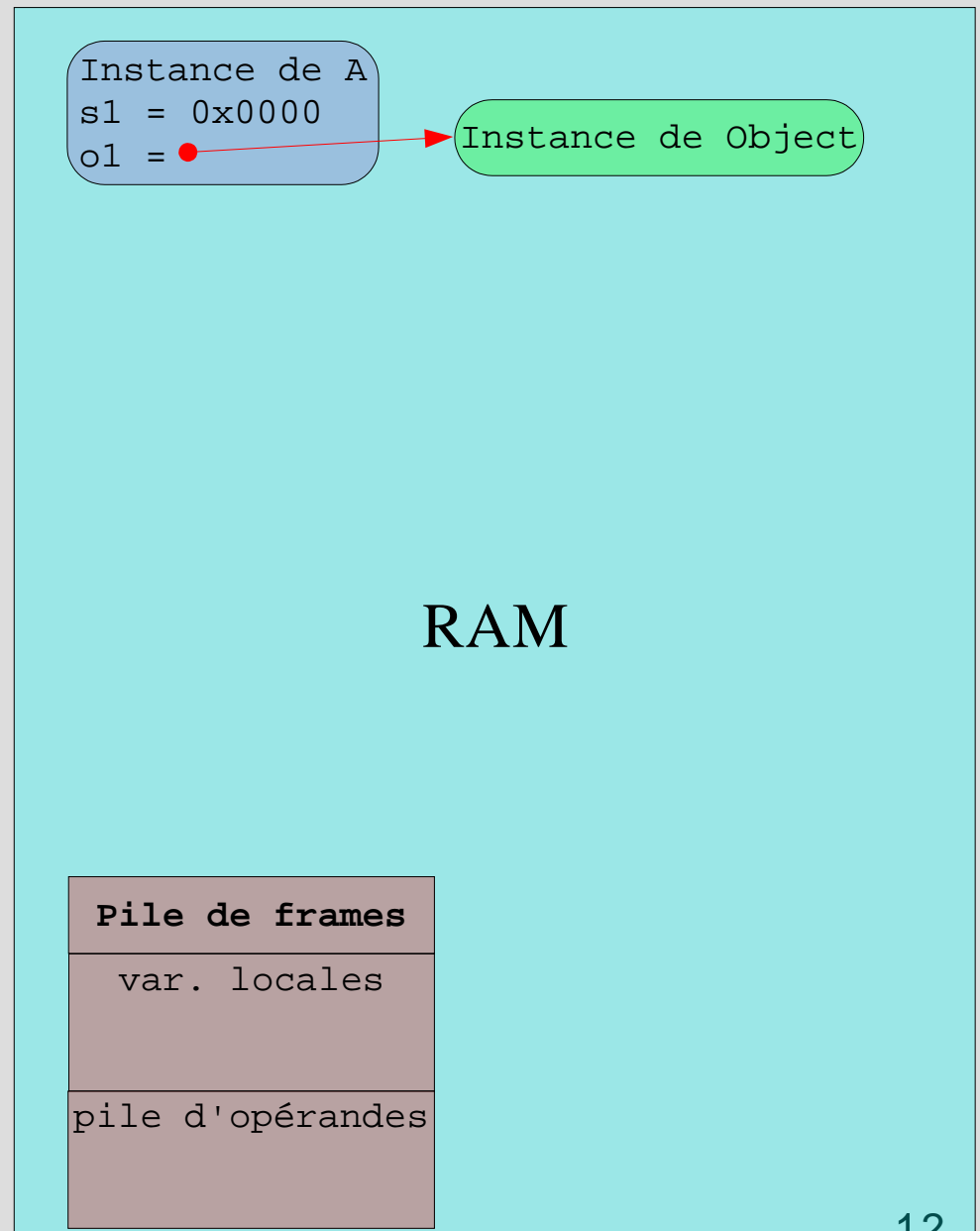


Exemple de code en Java

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1 ←----- ICI  
}  
  
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```



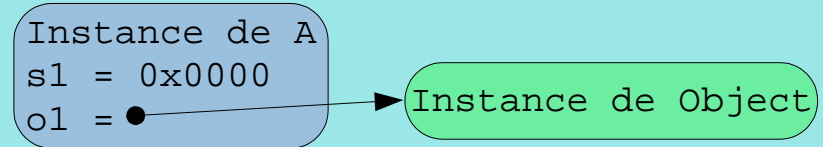
Exemple de code en Java

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

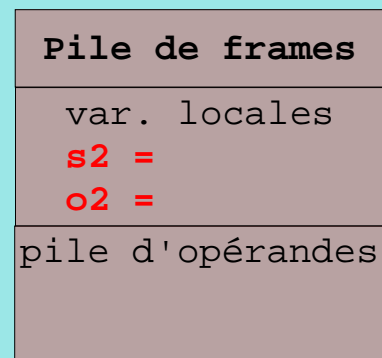
Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() { ←----- ICI  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```



RAM



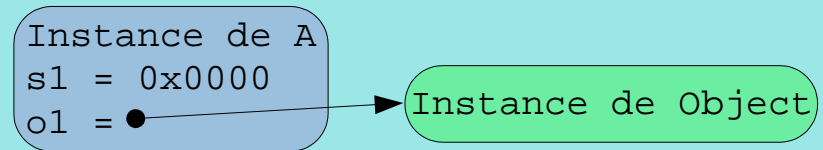
Exemple de code en Java

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

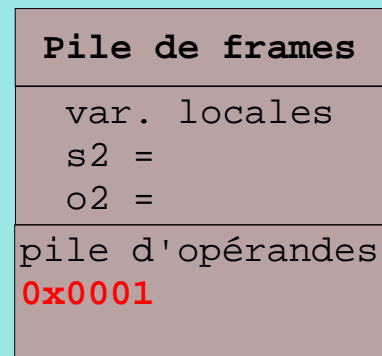
Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1 ←----- ICI  
    sstore s2  
    new Object  
    astore o2  
}
```



RAM



Exemple de code en Java

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

←----- ICI

Instance de A
s1 = 0x0000
o1 = ●

Instance de Object

RAM

Pile de frames

var. locales
s2 = 0x0001
o2 =

pile d'opérandes

Exemple de code en Java

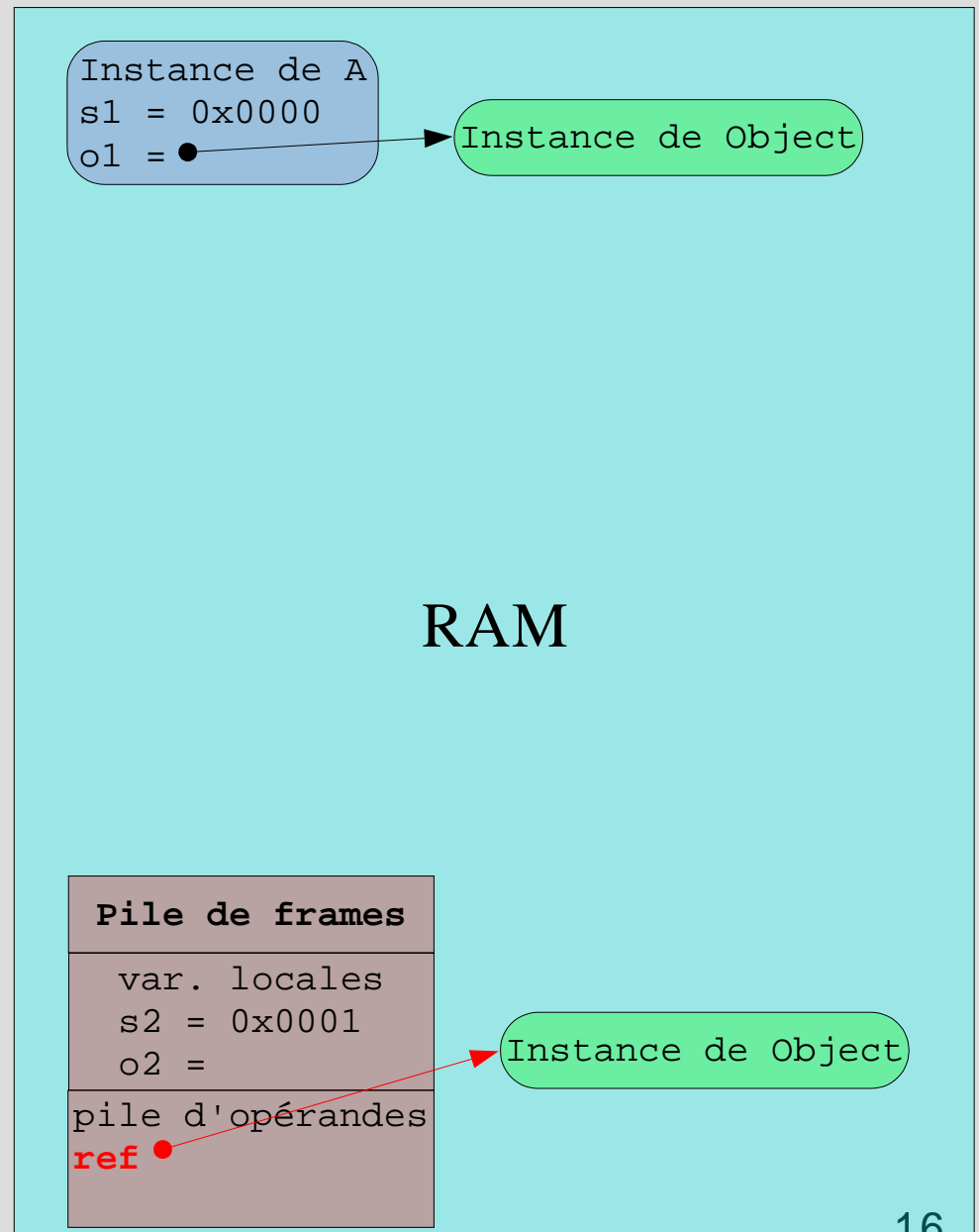
```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

←----- ICI



Exemple de code en Java

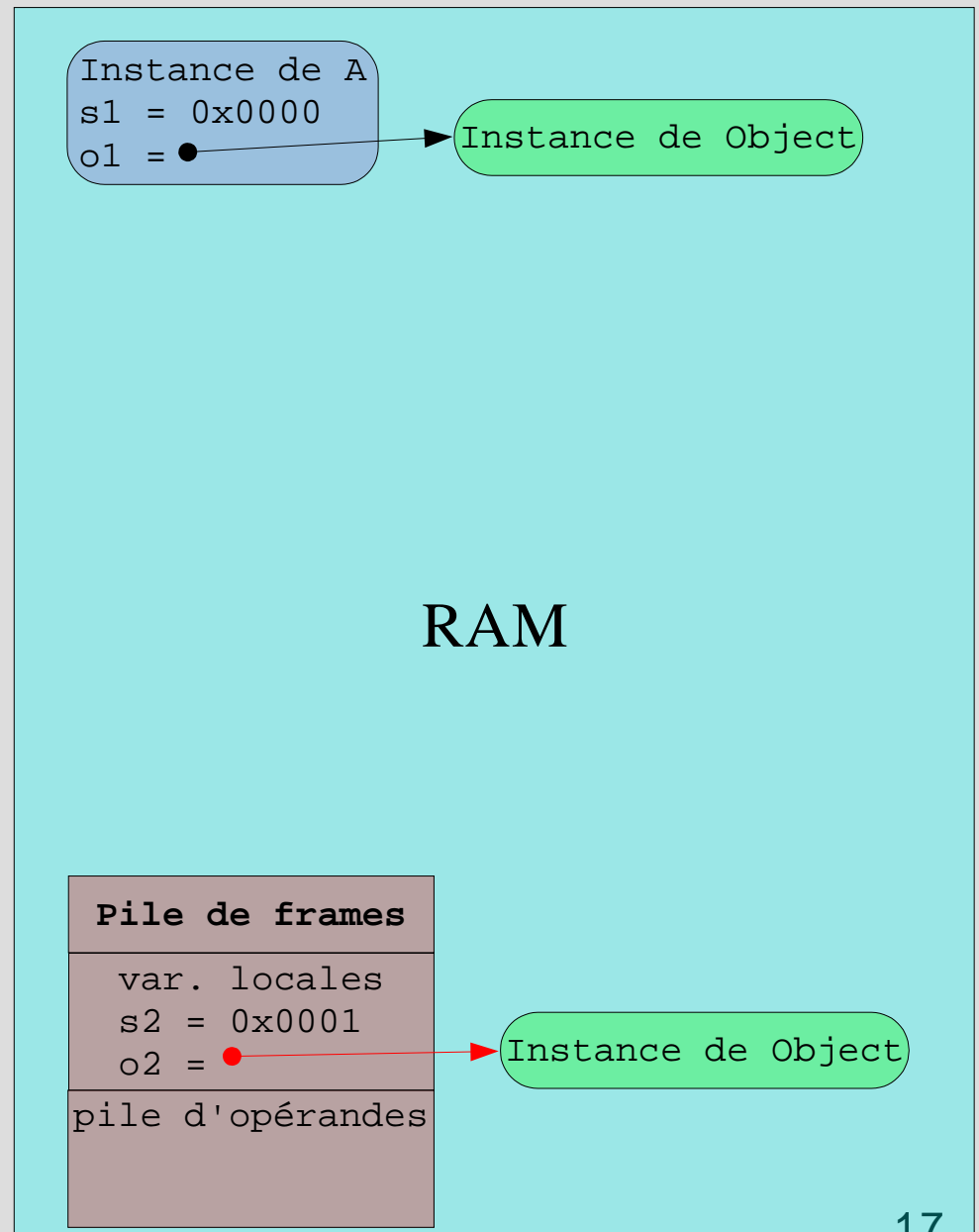
```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

←----- ICI



Exemple de code en Java

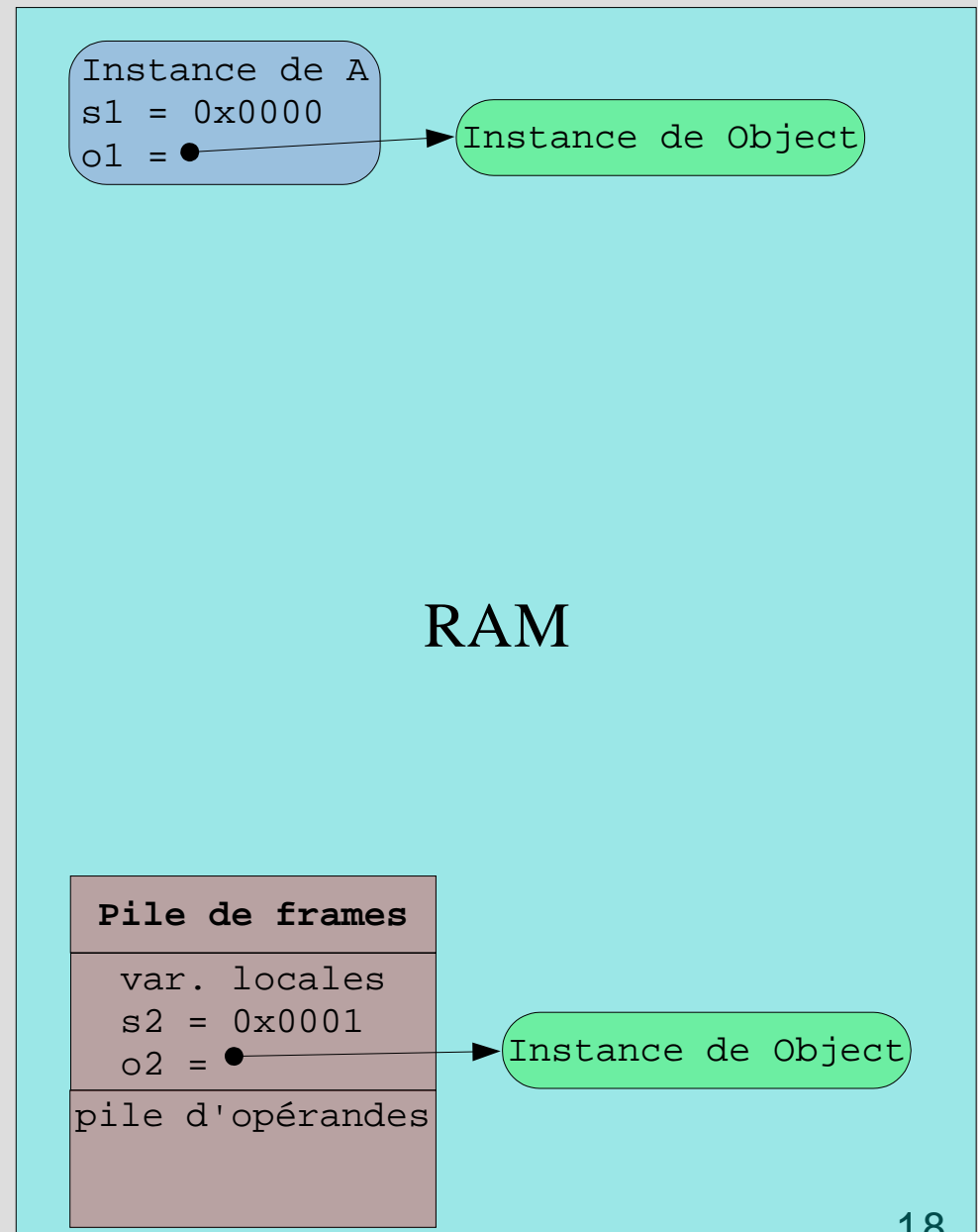
Remarques :

- sémantiques **identiques et homogènes** pour les variables locales et les variables d'instances (contenu et référence)

- Tous les objets sont dans une même mémoire **RAM**.

⇒ **Tout est temporaire !**

Normal, Java est un environnement de programmation temporaire.



Exemple de code en Java Card **sans** pré-persistence

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() { ----- ICI  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}  
  
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

Instance de A
s1 =
o1 =

EEPROM

RAM

Pile de frames

var. locales

pile d'opérandes

Exemple de code en Java Card **sans** pré-persistence

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

←----- ICI

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

Instance de A
s1 =
o1 =

EEPROM

RAM

Pile de frames

var. locales

pile d'opérandes

this

0x0000

Exemple de code en Java Card **sans** pré-persistance

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1 <----- ICI  
    aload_0  
    new Object  
    putfield_a o1  
}  
  
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

Instance de A
s1 = 0x0000
o1 =

EEPROM

RAM

| |
|------------------|
| Pile de frames |
| var. locales |
| pile d'opérandes |

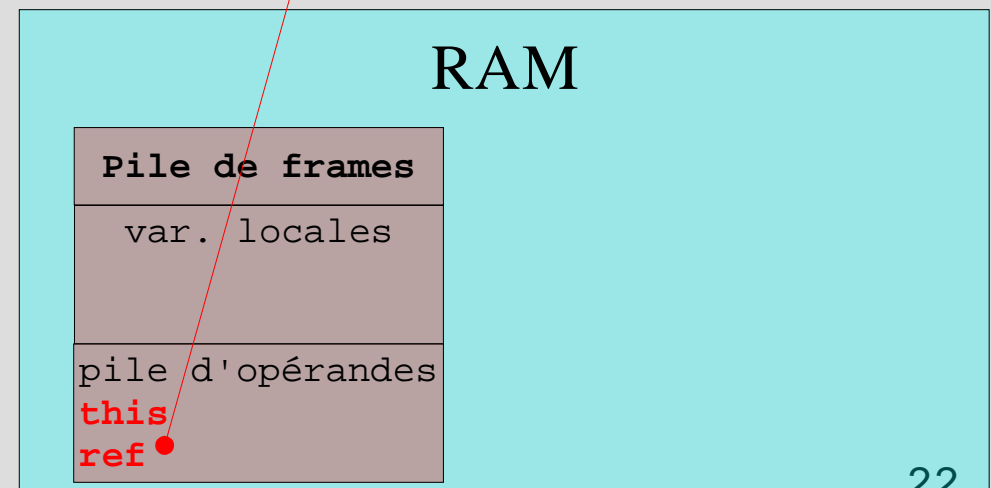
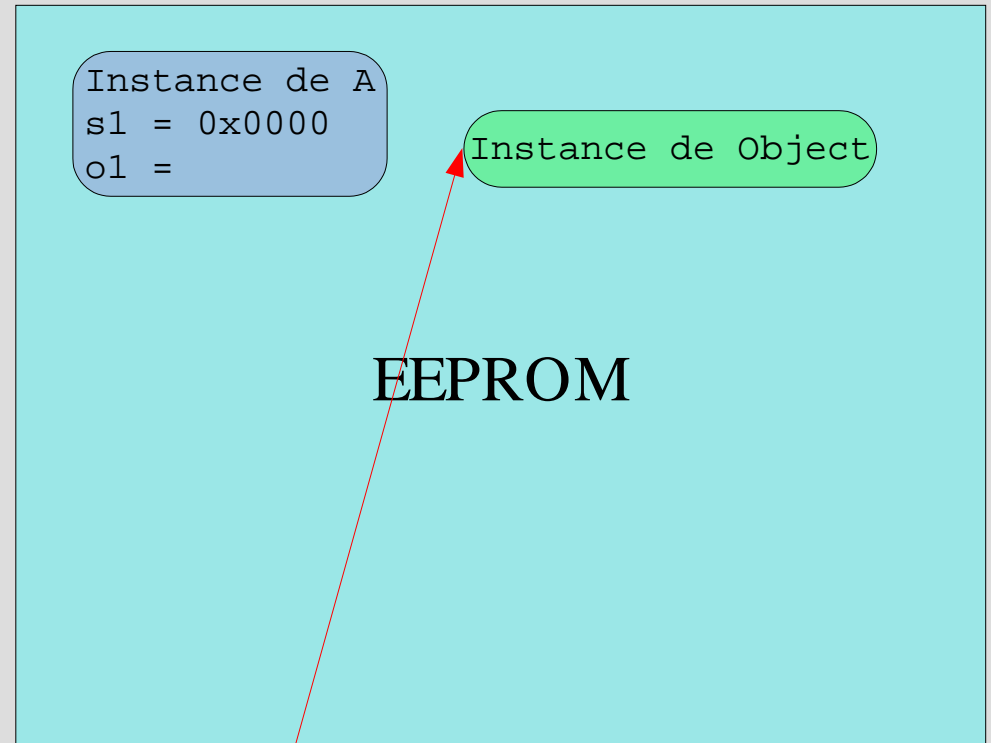
Exemple de code en Java Card **sans** pré-persistance

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object ←----- ICI  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

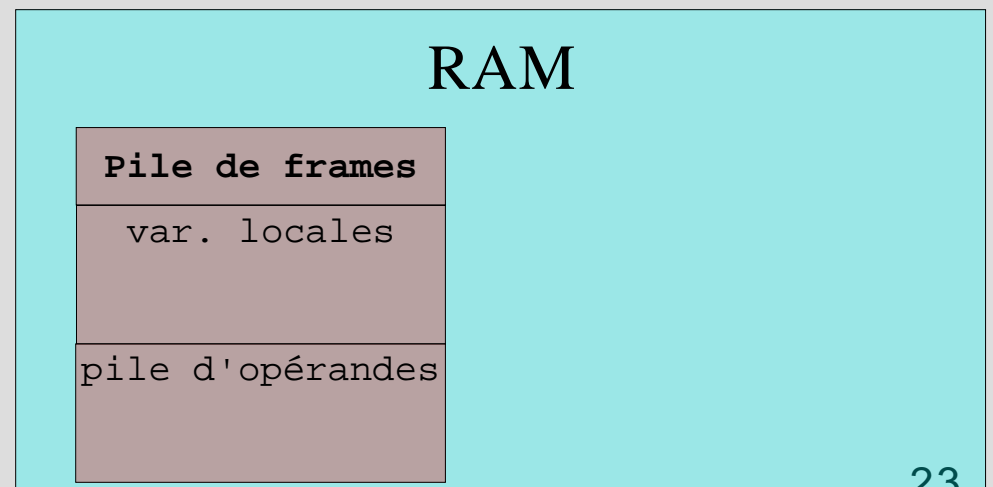
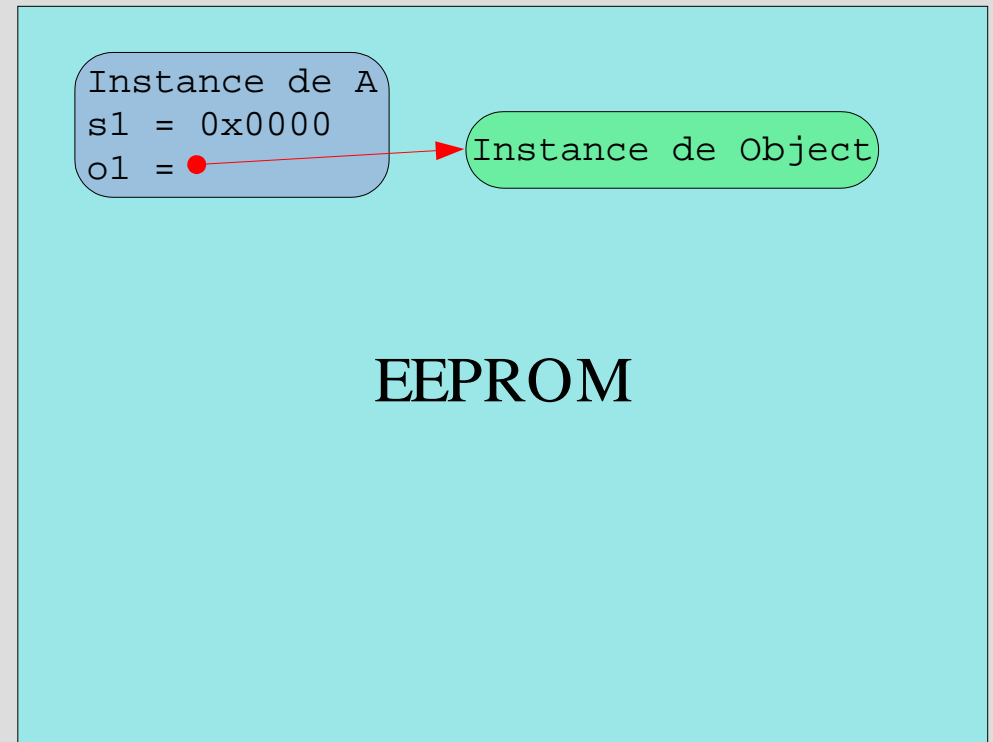


Exemple de code en Java Card **sans** pré-persistance

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1 ←----- ICI  
}  
  
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```



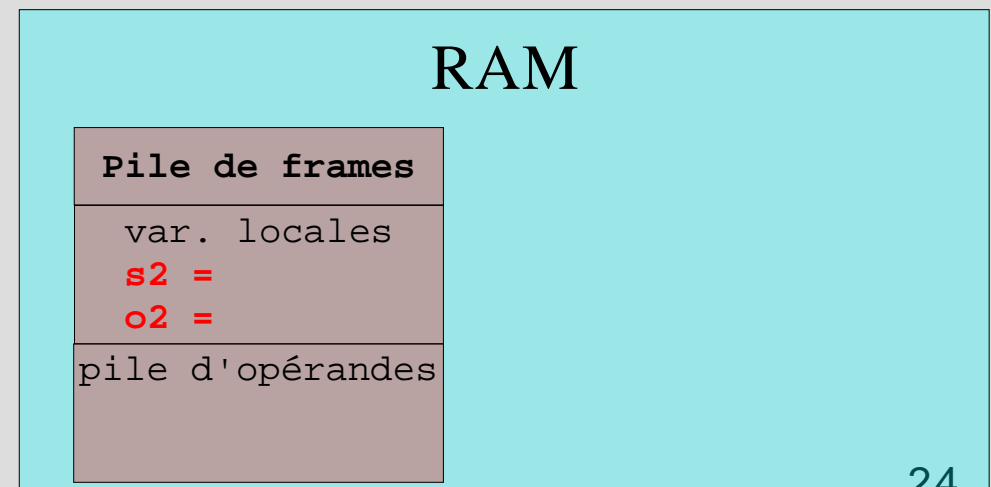
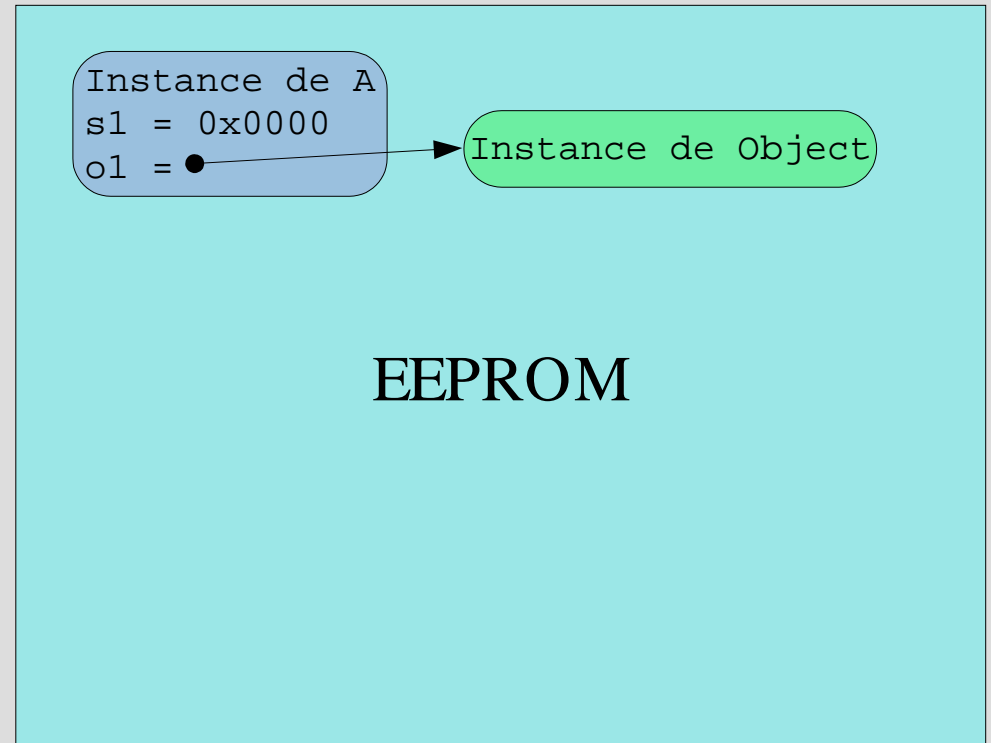
Exemple de code en Java Card **sans** pré-persistance

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() { ←----- ICI  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```



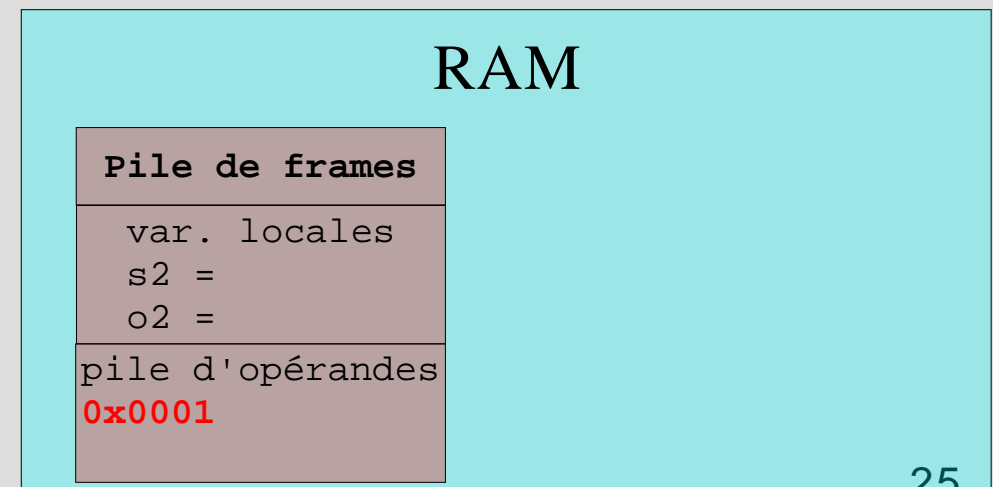
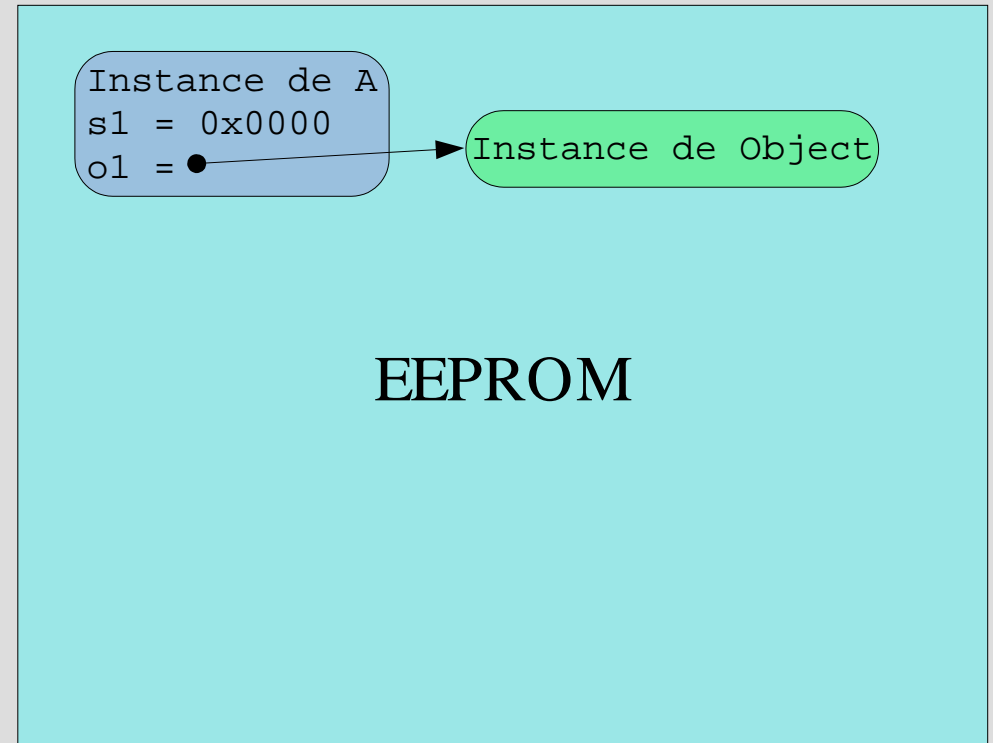
Exemple de code en Java Card **sans** pré-persistance

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1 ←----- ICI  
    sstore s2  
    new Object  
    astore o2  
}
```



Exemple de code en Java Card **sans** pré-persistance

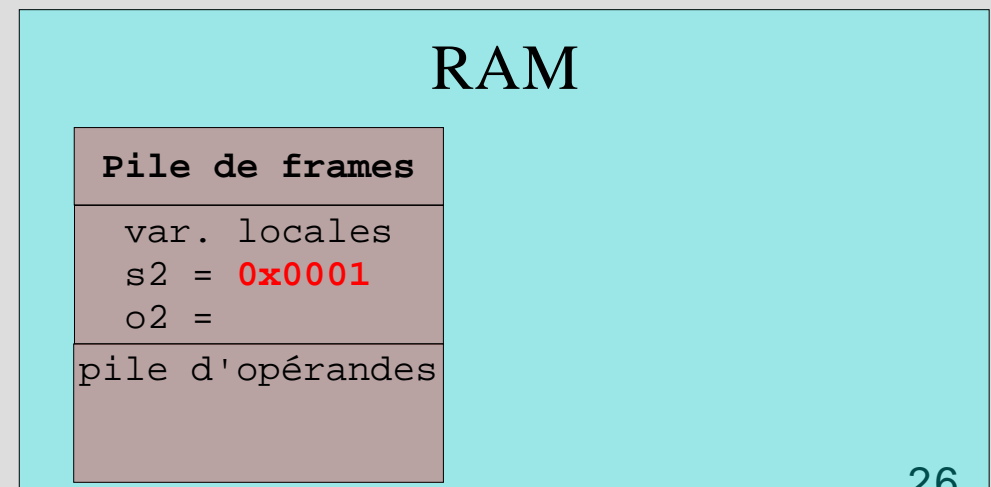
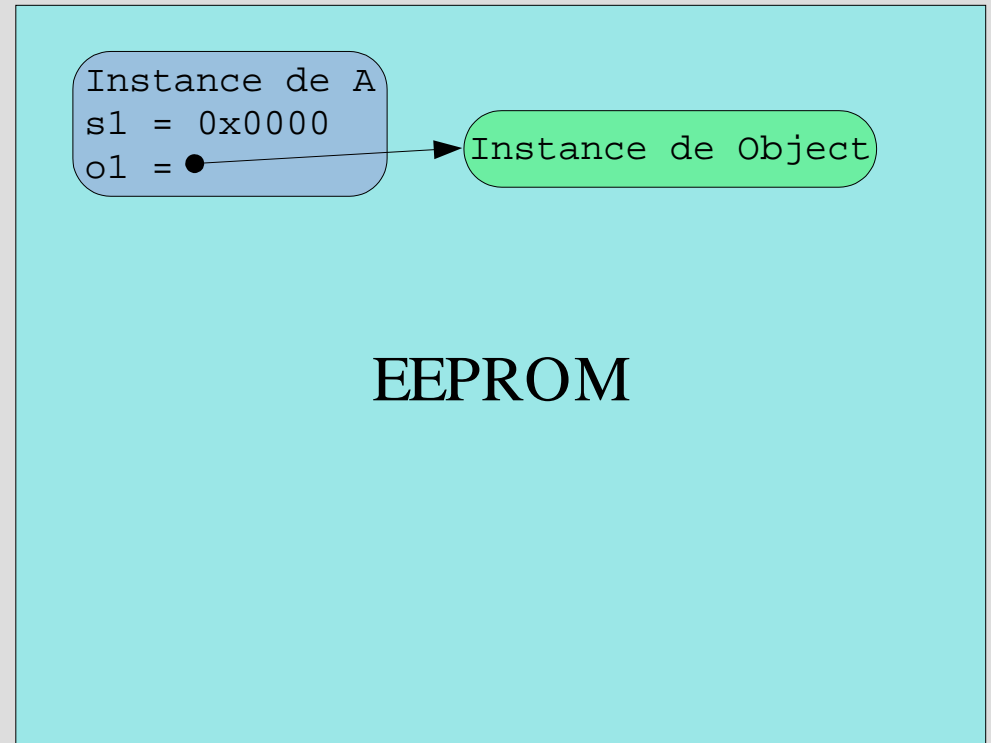
```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

←----- ICI



Exemple de code en Java Card **sans** pré-persistance

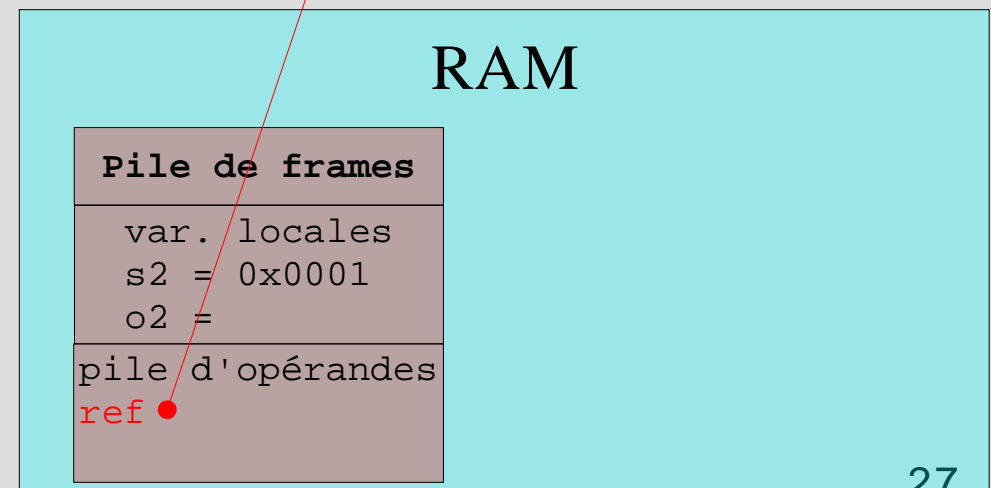
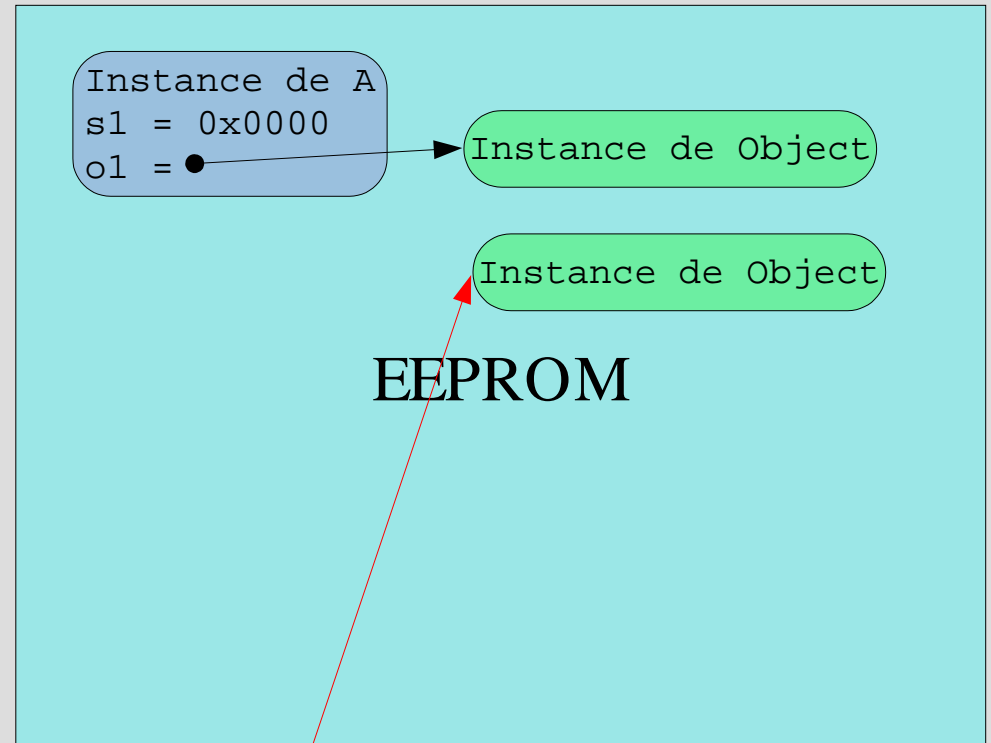
```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

←----- ICI



Exemple de code en Java Card **sans** pré-persistance

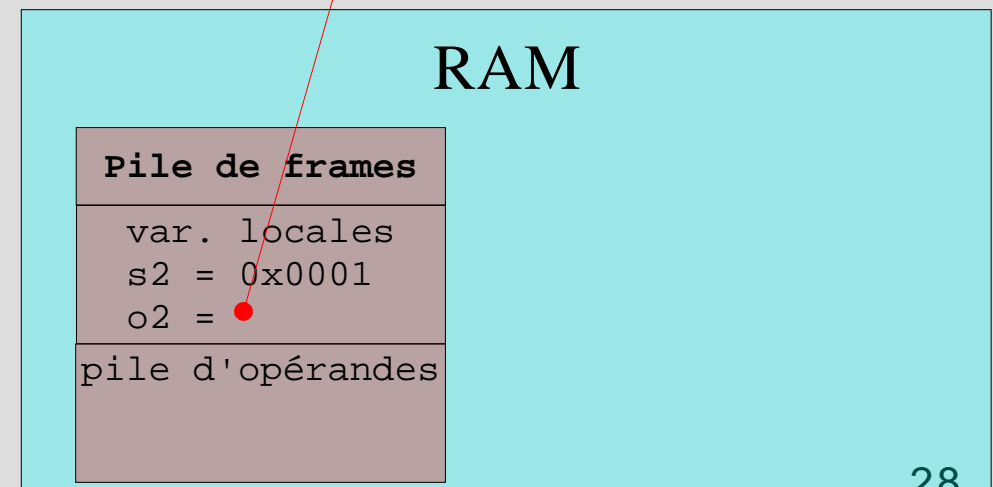
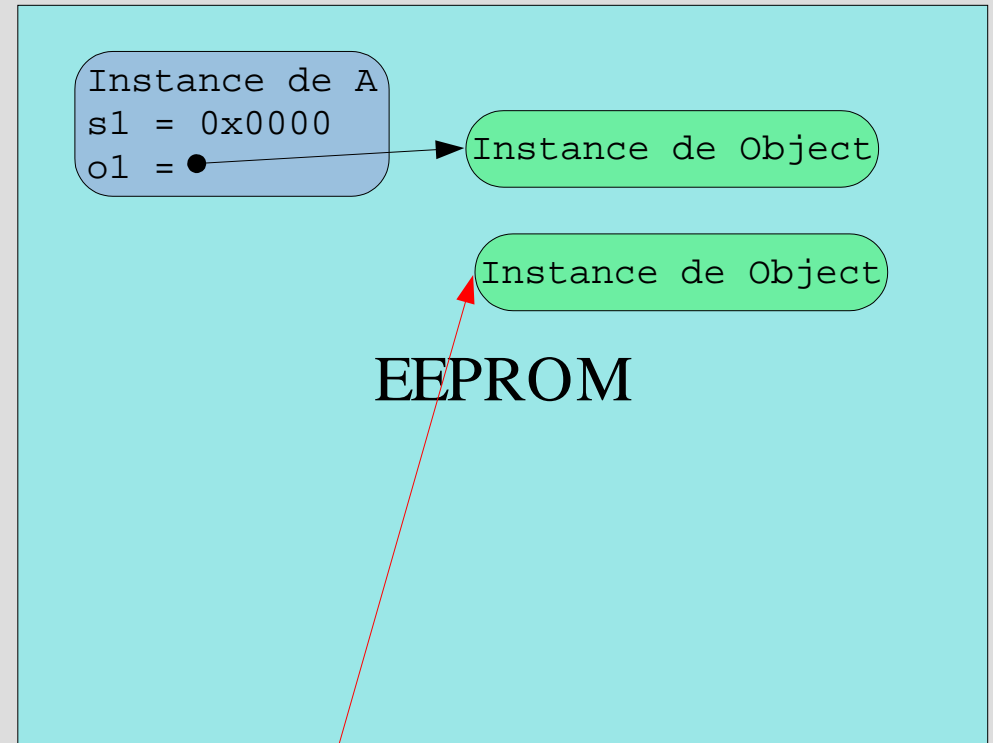
```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

←----- ICI



Exemple de code en Java Card **sans** pré-persistence

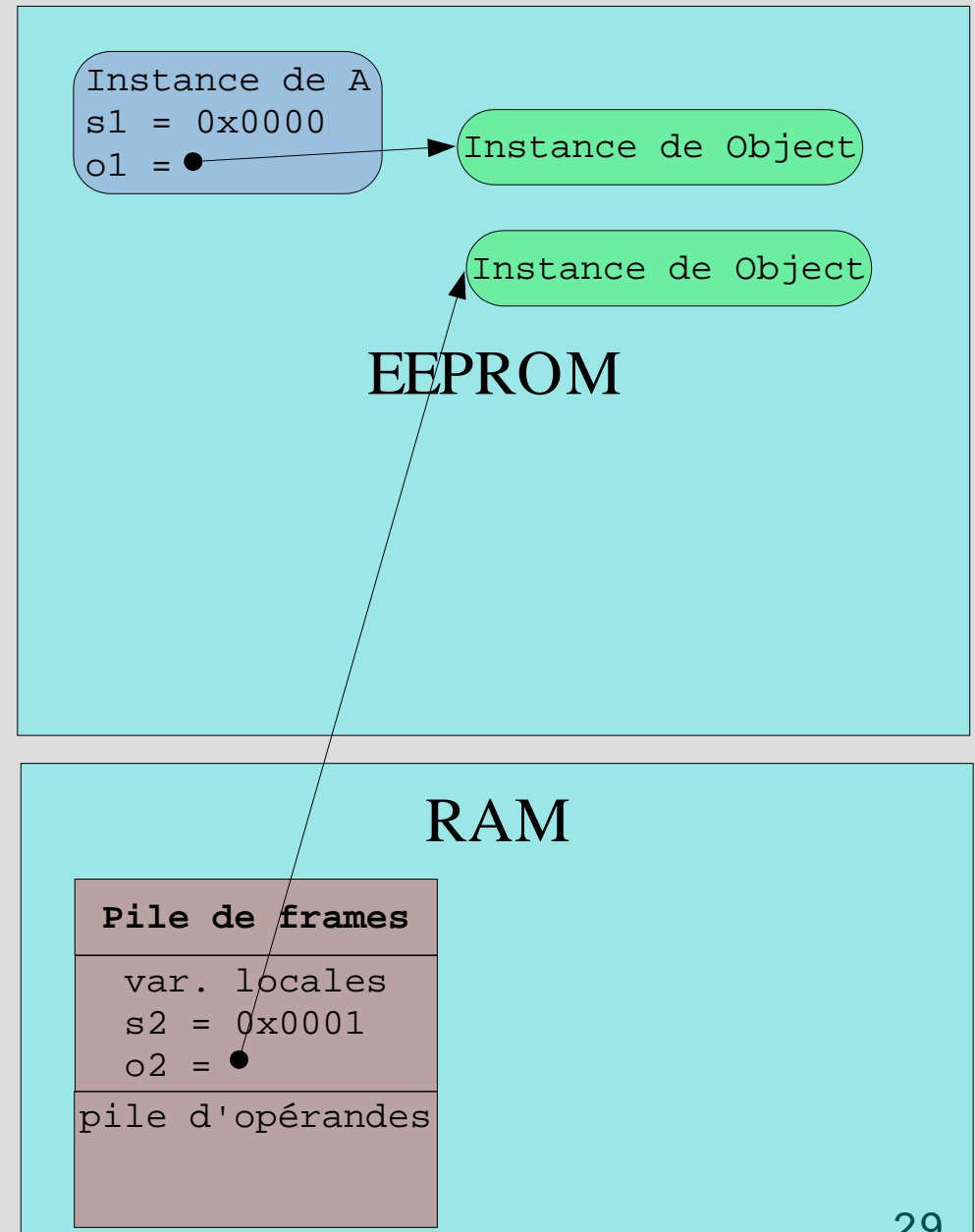
Remarques :

- sémantiques **identiques** les variables d'instances (contenu et référence)
- sémantiques **différentes** pour les variables locales (contenu et référence)
- Tous les objets sont dans l'EEPROM.

Problèmes :

Perte d'alimentation

⇒ **objet inaccessible et non ramassé**



Exemple de code en Java Card **avec** pré-persistence

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() { ←----- ICI  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}  
  
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

Instance de A
s1 =
o1 =

EEPROM

RAM

Pile de frames

var. locales

pile d'opérandes

Exemple de code en Java Card **avec** pré-persistence

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

←----- ICI

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

Instance de A
s1 =
o1 =

EEPROM

RAM

Pile de frames

var. locales

pile d'opérandes

this
0x0000

Exemple de code en Java Card **avec** pré-persistence

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1 <----- ICI  
    aload_0  
    new Object  
    putfield_a o1  
}  
  
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

Instance de A
s1 = 0x0000
o1 =

EEPROM

RAM

| |
|------------------|
| Pile de frames |
| var. locales |
| pile d'opérandes |

Exemple de code en Java Card avec pré-persistance

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object ←----- ICI  
    putfield_a o1  
}
```

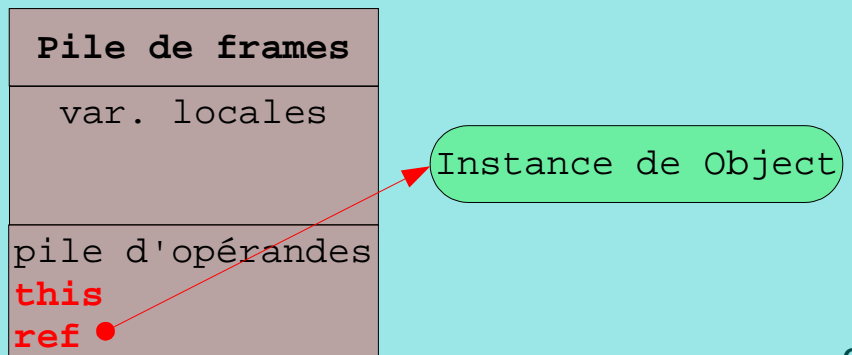


```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

Instance de A
s1 = 0x0000
o1 =

EEPROM

RAM

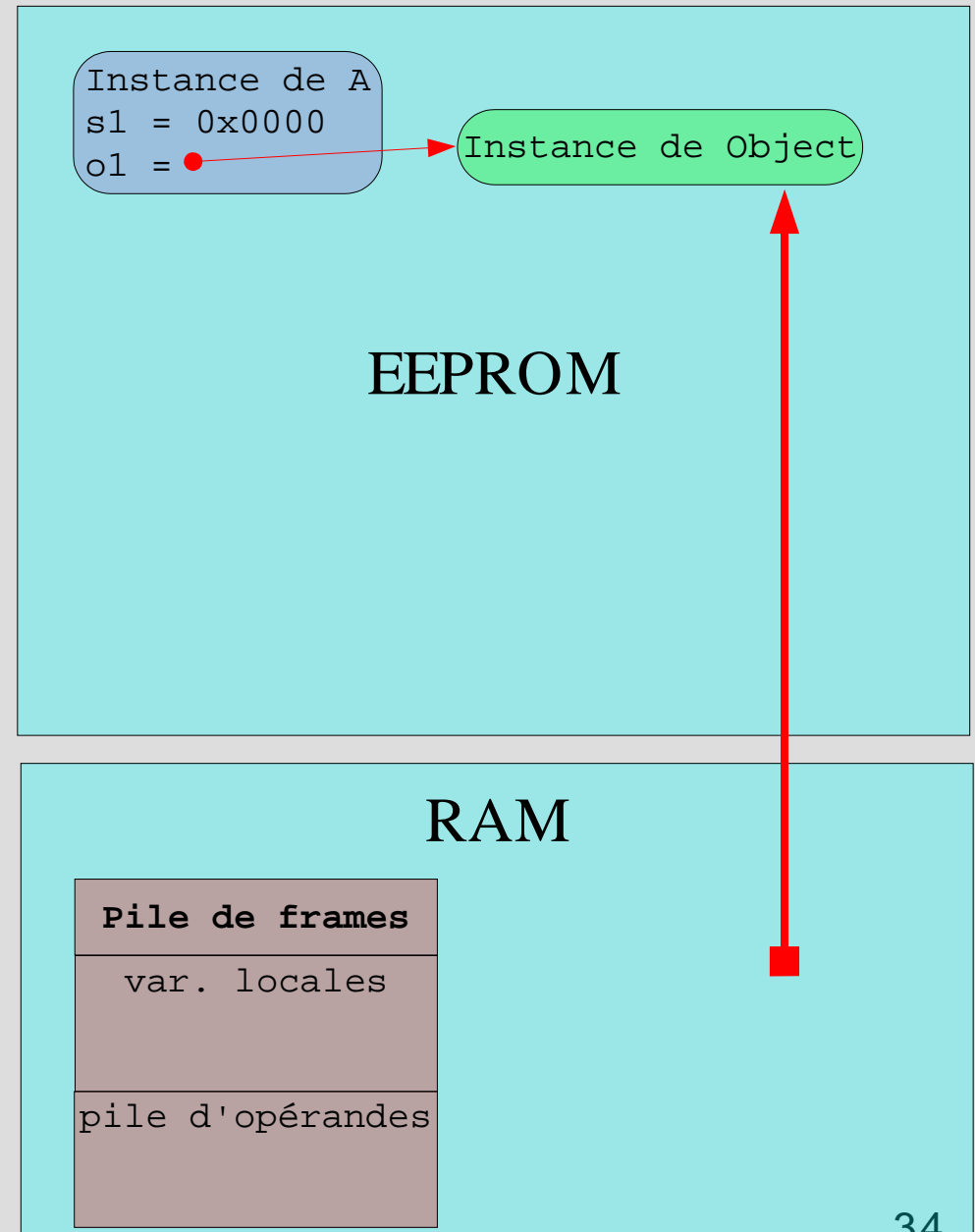


Exemple de code en Java Card avec pré-persistance

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1 ←----- ICI  
}  
  
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```



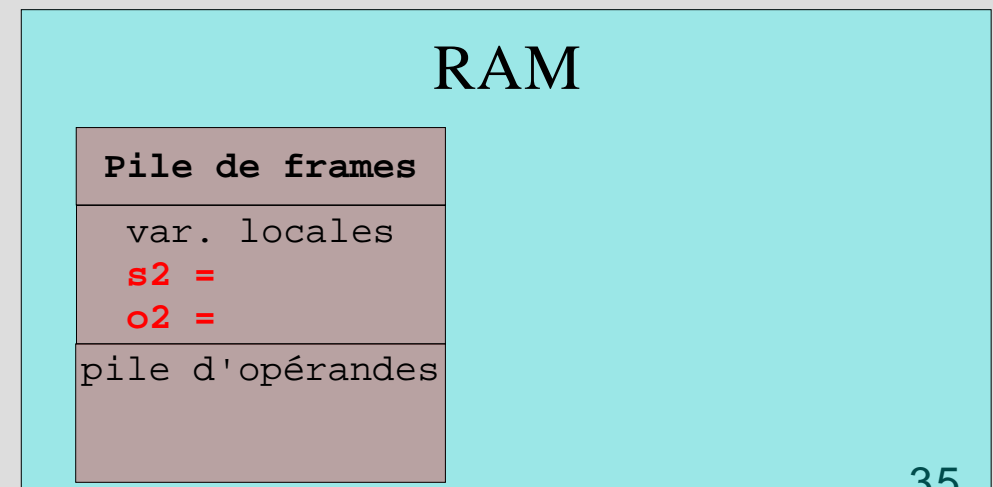
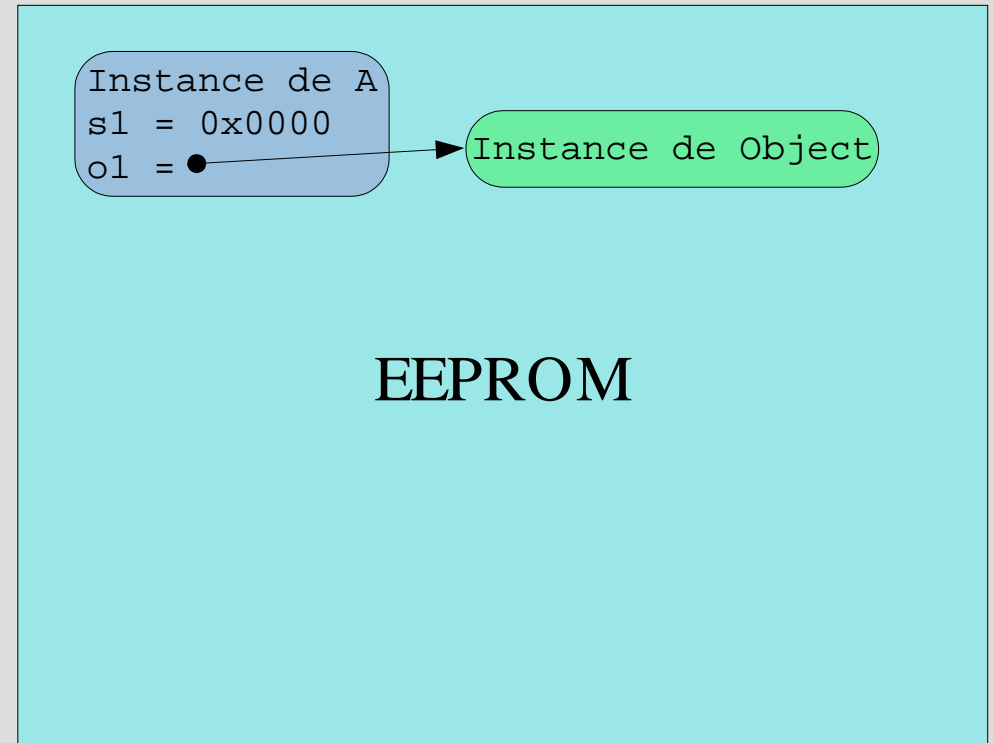
Exemple de code en Java Card avec pré-persistance

```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() { ←----- ICI  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```



Exemple de code en Java Card avec pré-persistance

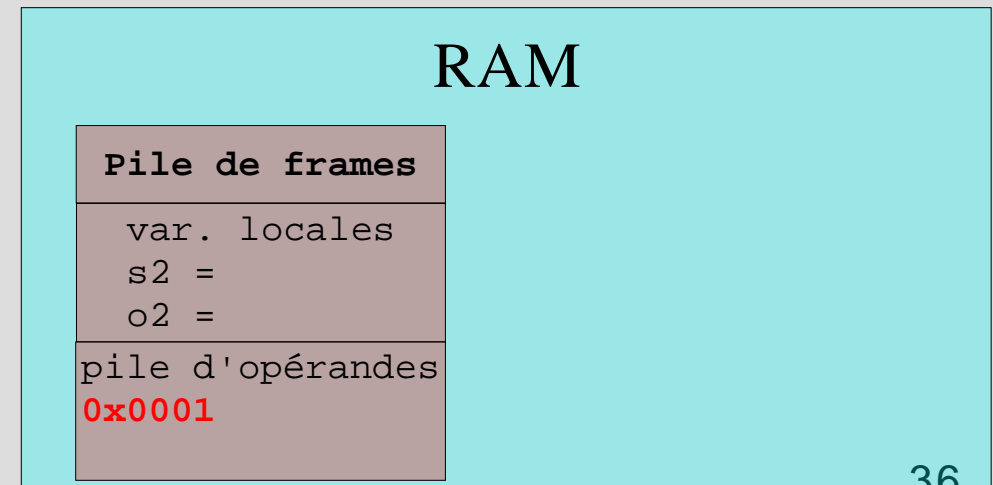
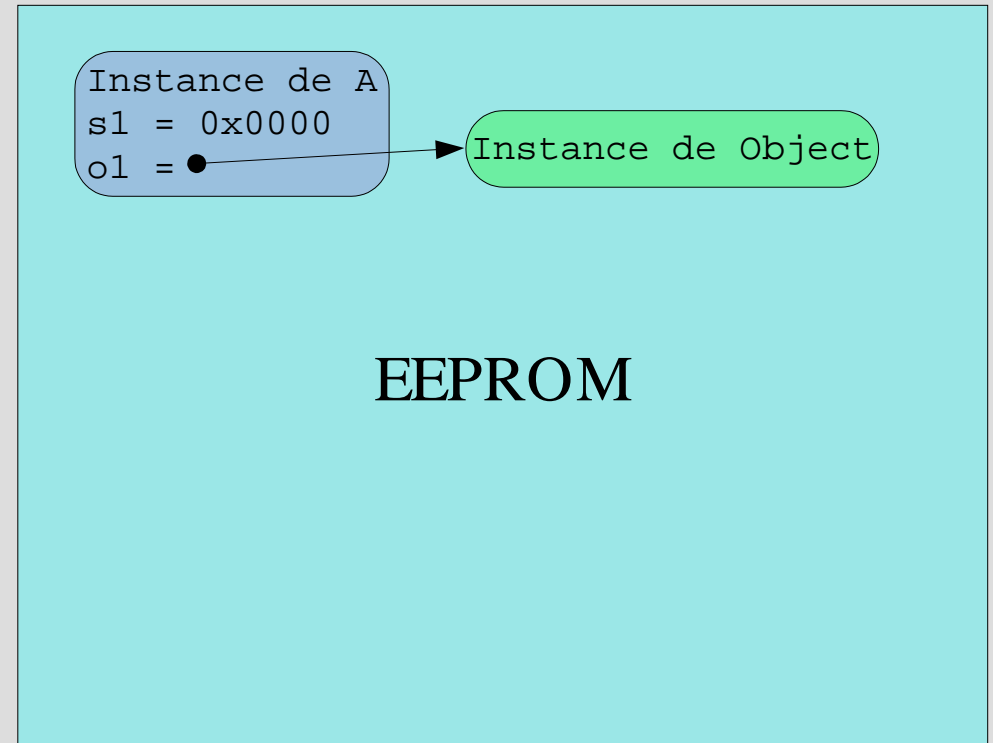
```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

<----- ICI



Exemple de code en Java Card avec pré-persistance

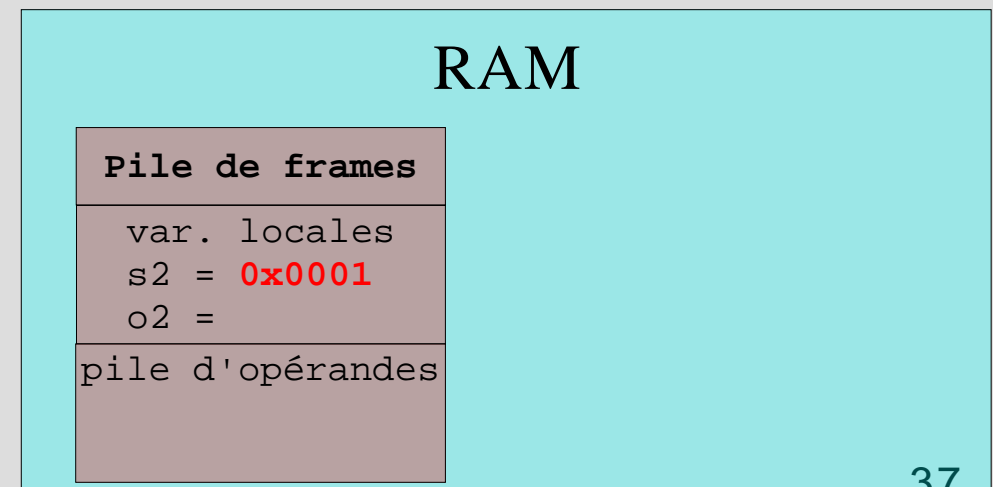
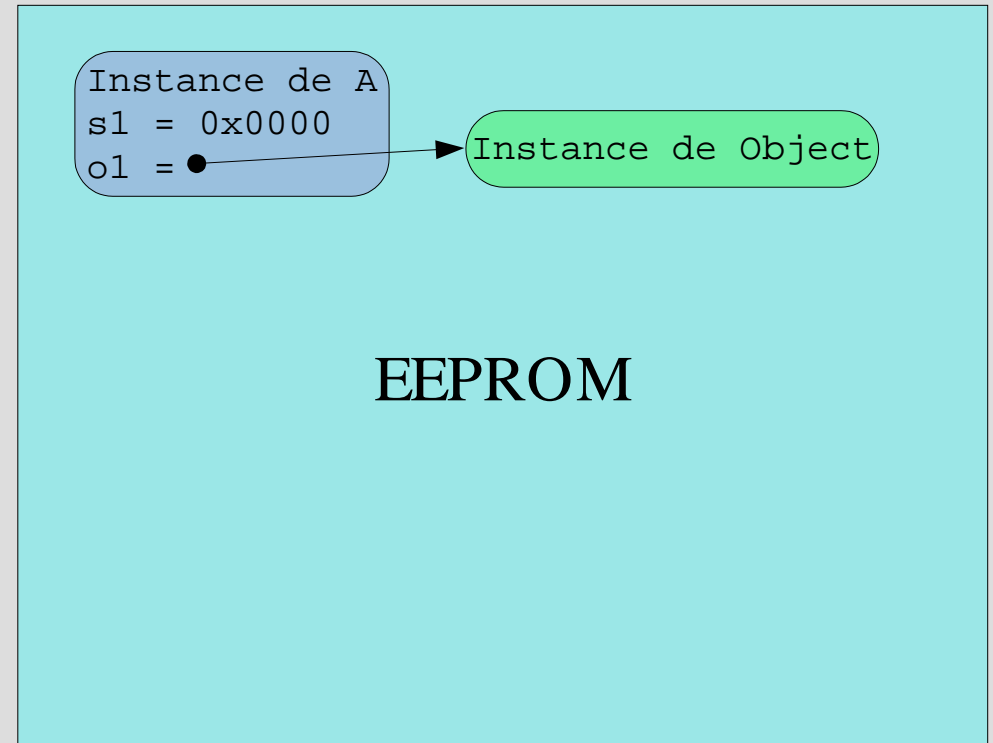
```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

<----- ICI



Exemple de code en Java Card avec pré-persistance

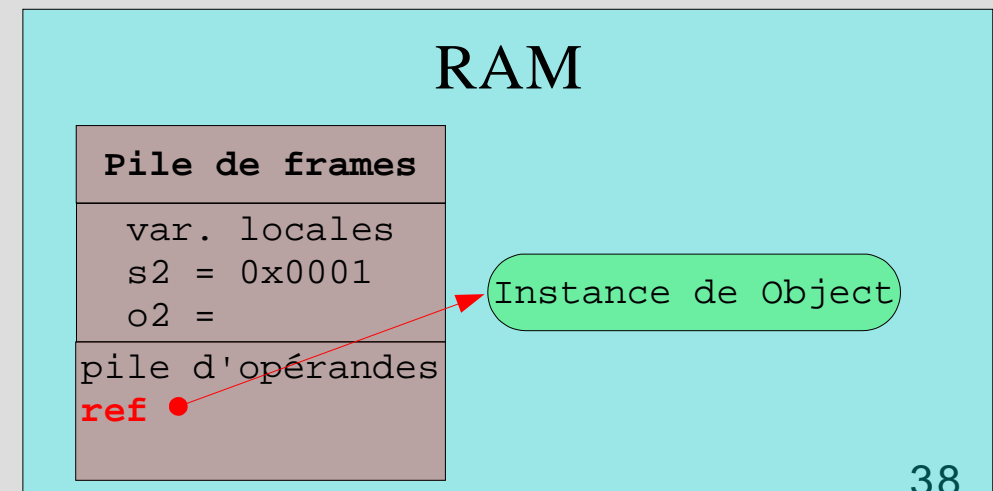
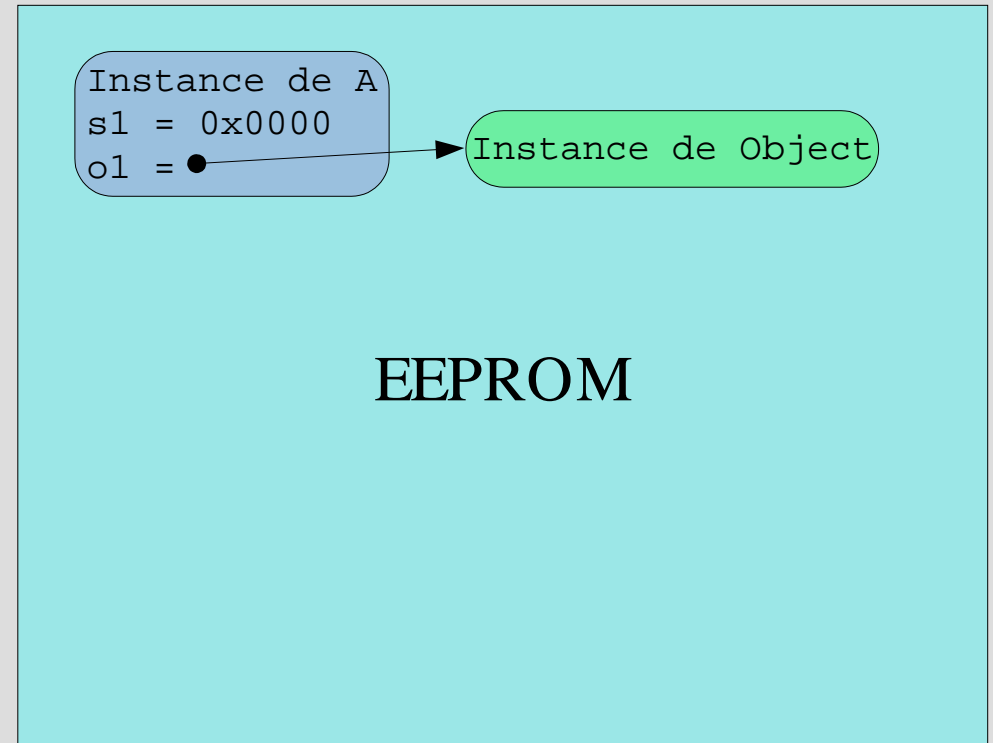
```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

←----- ICI



Exemple de code en Java Card avec pré-persistance

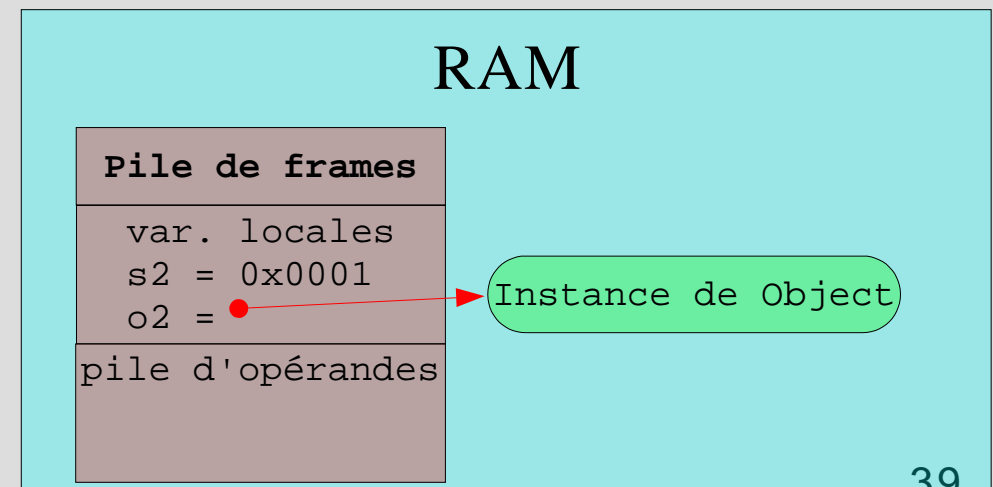
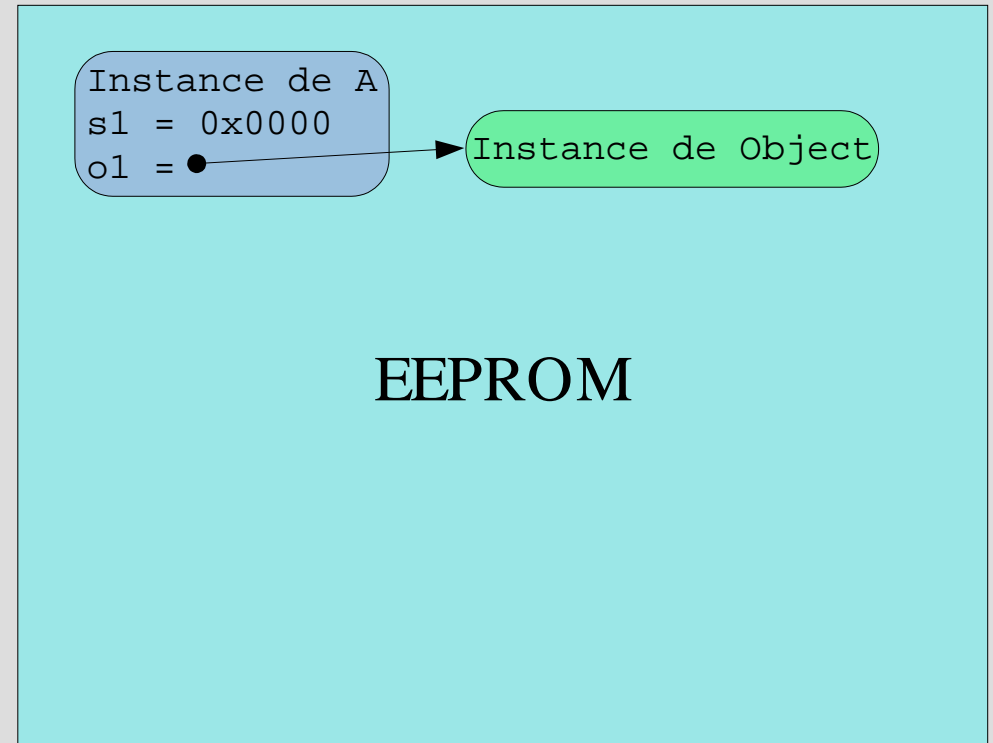
```
public class A {  
  
    private short s1;  
    private Object o1;  
  
    public void foo() {  
        s1 = (short) 0;  
        o1 = new Object();  
    }  
  
    public void bar() {  
        short s2 = (short) 1;  
        Object o2 = new Object();  
    }  
}
```

Traduction en bytecodes des méthodes foo et bar

```
void foo() {  
    aload_0  
    sconst_0  
    putfield_s s1  
    aload_0  
    new Object  
    putfield_a o1  
}
```

```
void bar() {  
    sconst_1  
    sstore s2  
    new Object  
    astore o2  
}
```

←----- ICI



Exemple de code en Java Card **avec** pré-persistance

Remarques :

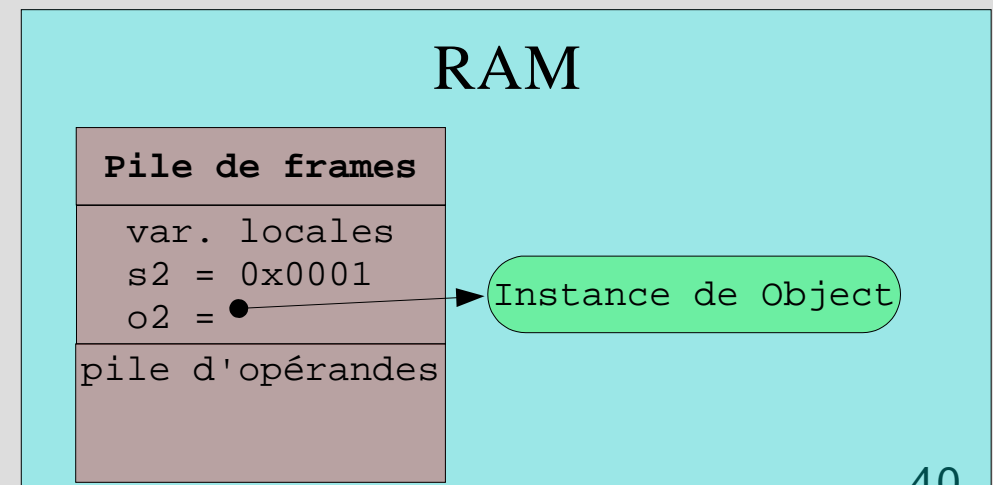
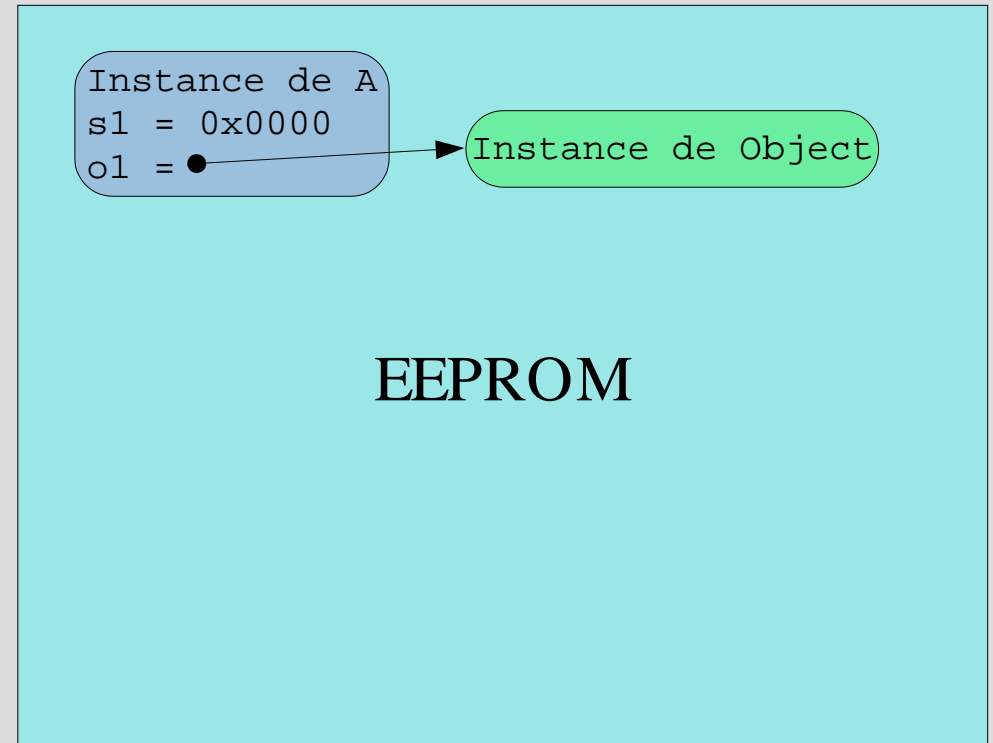
- sémantiques **identiques** les variables d'instances (contenu et référence)
- sémantiques **identiques** pour les variables locales (contenu et référence)
- Les objets sont dans deux mémoires

Avantages :

- sémantiques **homogènes** comme en Java
- sur la perte d'alimentation : **ramasse-miette naturelle de la RAM**
- ⇒ pas d'objet inaccessible non ramassé

Inconvénients :

- Mécanisme de recopie des objets de la RAM vers l'EEPROM



Conclusion et perspectives

Notre contribution :

- L'introduction du concept de pré-persistence :
 - obtention de la sémantique originelle de Java

Une clarification des spécifications concernant le TAS (définition, contenu, localisation).

⇒ 4 modèles mémoires aux propriétés très différentes : fonctionnalités, sémantique et **sécurité** !

Perspectives :

Formalisation à l'aide d'une sémantique opérationnelle des interactions entre objets :

- persistants
- transients
- pré-persistants

⇒ visualisation de la contamination lors de l'affectation d'un objet transient ou pré-persistant à un champ d'un objet persistant !

But final : Proposer au Java Card Forum une modification des spécifications.

Comparatif des 2 solutions par rapport à Java

| Propriété | Java | Java Card sans PP | Java Card avec PP |
|--|-------------------------------|--------------------------|---|
| Sémantique pour les variables d'instance | cohérent | cohérent mais persistant | cohérent mais persistant |
| Sémantique pour les variables locales | cohérent | incohérent avec Java | cohérent avec Java |
| Inaccessibilité des objets | aucun grâce au ramasse-miette | oui | non sauf si le programmeur le désire ! Il y a un mécanisme de ramasse-miette naturelle de la RAM |