Cours JAVA : Gestion de la mémoire en Java.

Version 3.01

Julien Sopena[1]

[1]julien.sopena@lip6.fr
Équipe REGAL - INRIA Rocquencourt
LIP6 - Université Pierre et Marie Curie

Licence professionnelle DANT - 2015/2016

Grandes lignes du cours

- **Manipulation des objets en mémoire**
  - Allocation
  - Affectation
  - Comparaison
  - Duplication
- **Les méthodes**
  - Appels de méthodes
  - La récursion
- **Les tableaux**
- **Les objets immuables**
  - Les classes enveloppes
  - La classe String
- **Les classes**
  - Les membres de classe
  - L'héritage

Outline

Outline

Premier exemple d'allocation.

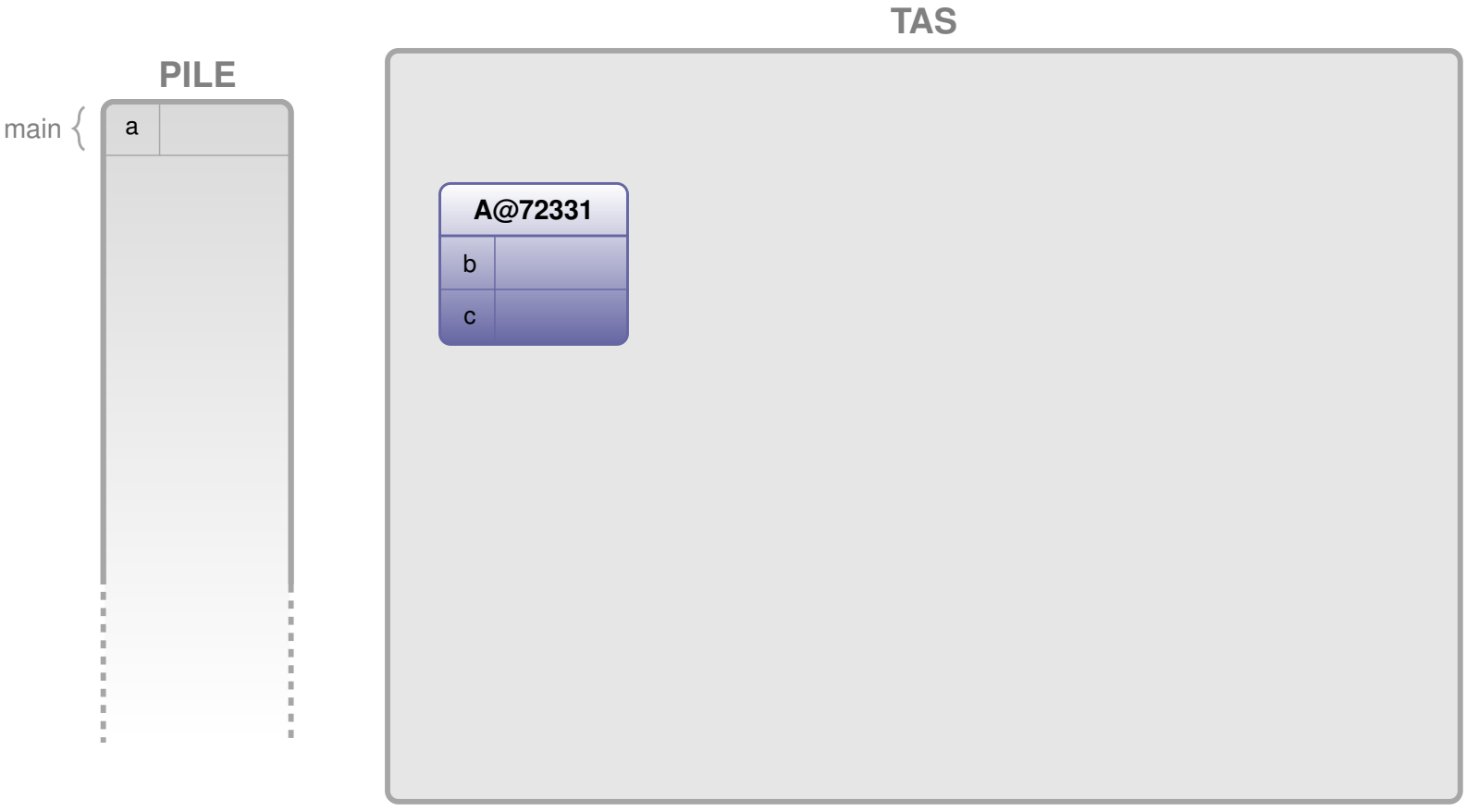
```
public class A {
  B b ;
  C c ;
}
```

```
public class C {
  int i;
}
```

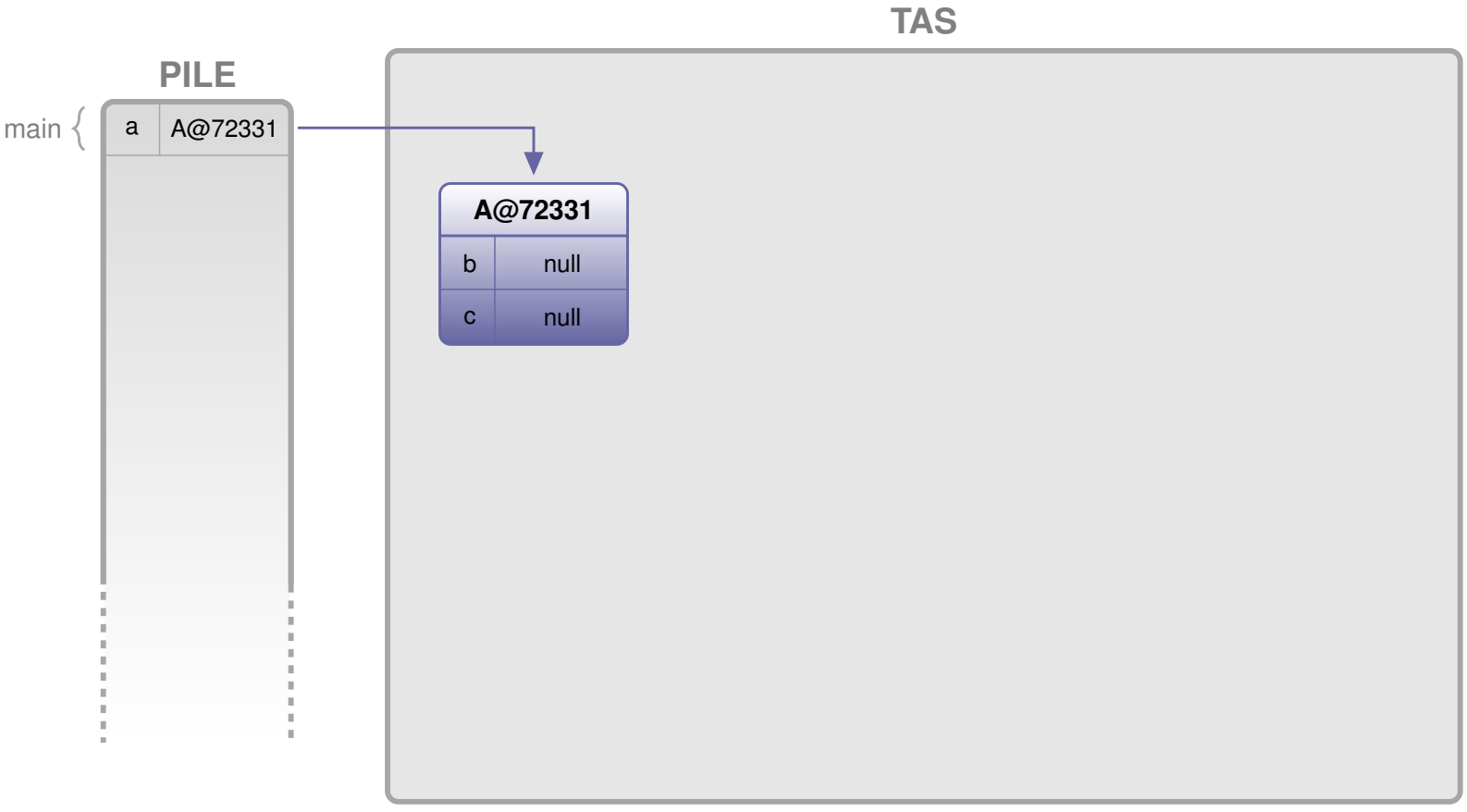
```
public class B {
  int i;
  C c;
}
```

```
public class Main {
  public static void main(String[] args) {
    A a = A();
  }
}
```

Premier exemple d'allocation.



Premier exemple d'allocation.



Allocations avec des constructeurs.

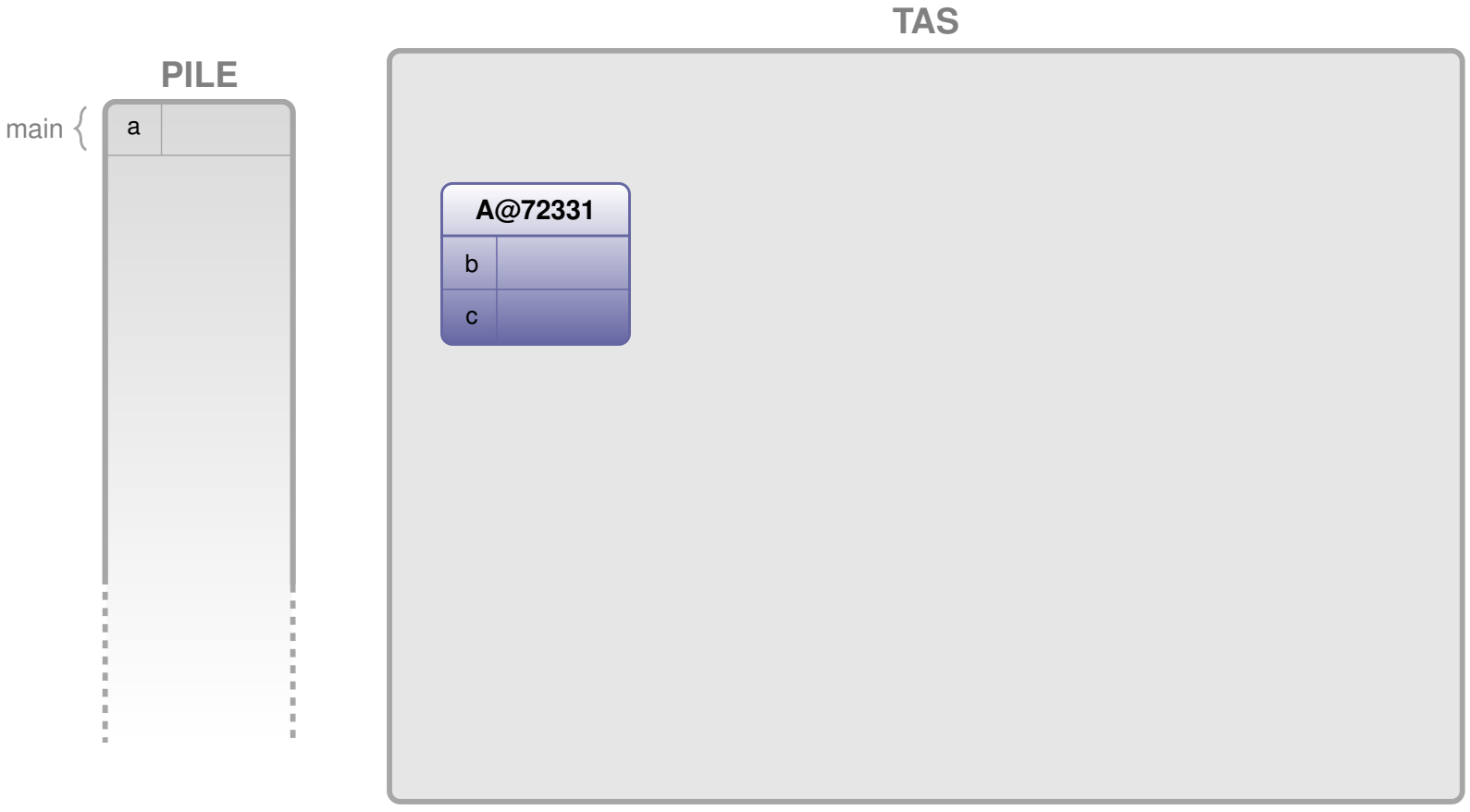
```
public class A {
  B b ;
  C c ;
  public A() {
    b = new B(1,2);
    c = new C(3);
  }
}
```

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
}
```

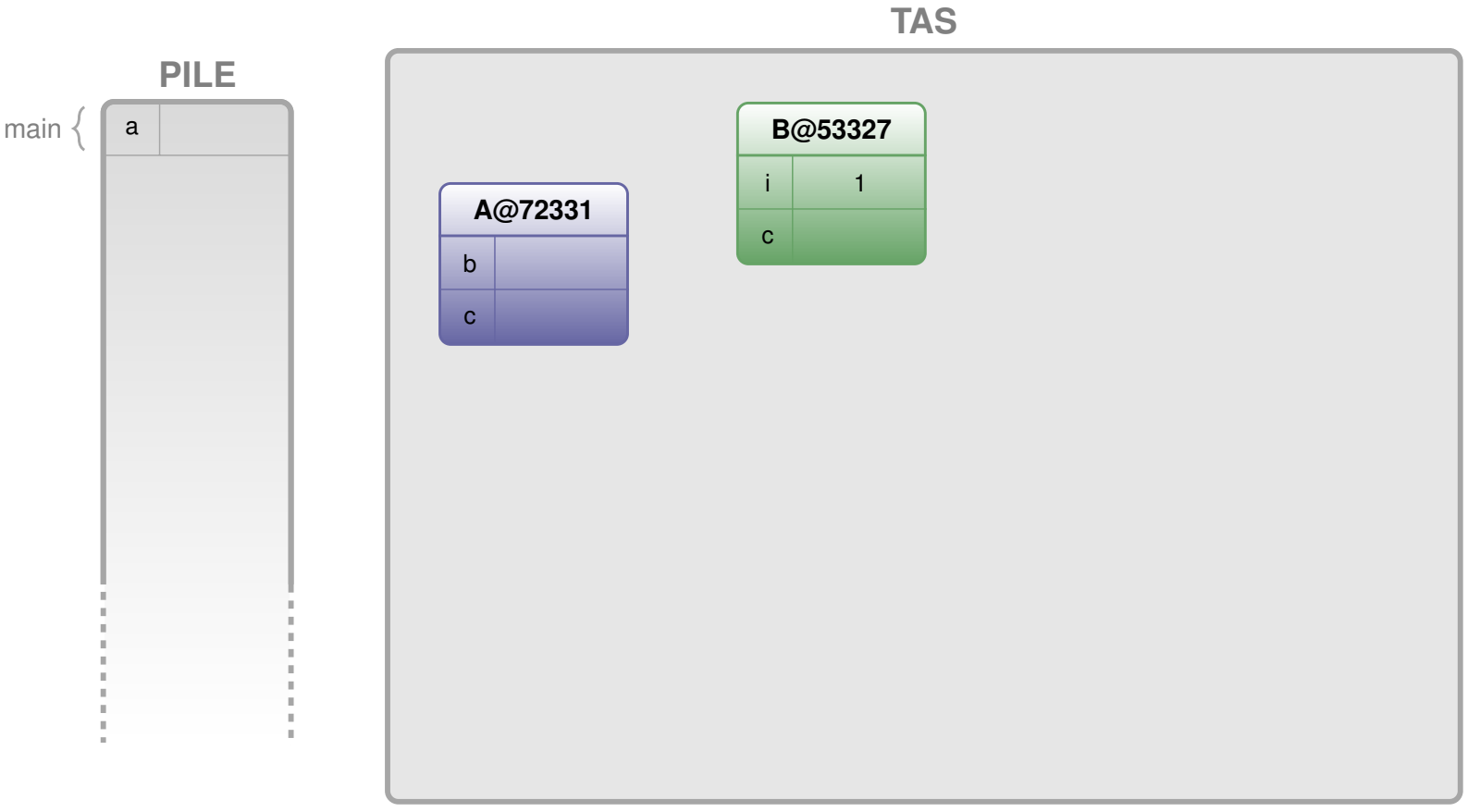
```
public class C {
  int i;
  public C(int y) {
    i = y;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    A a = new A();
  }
}
```

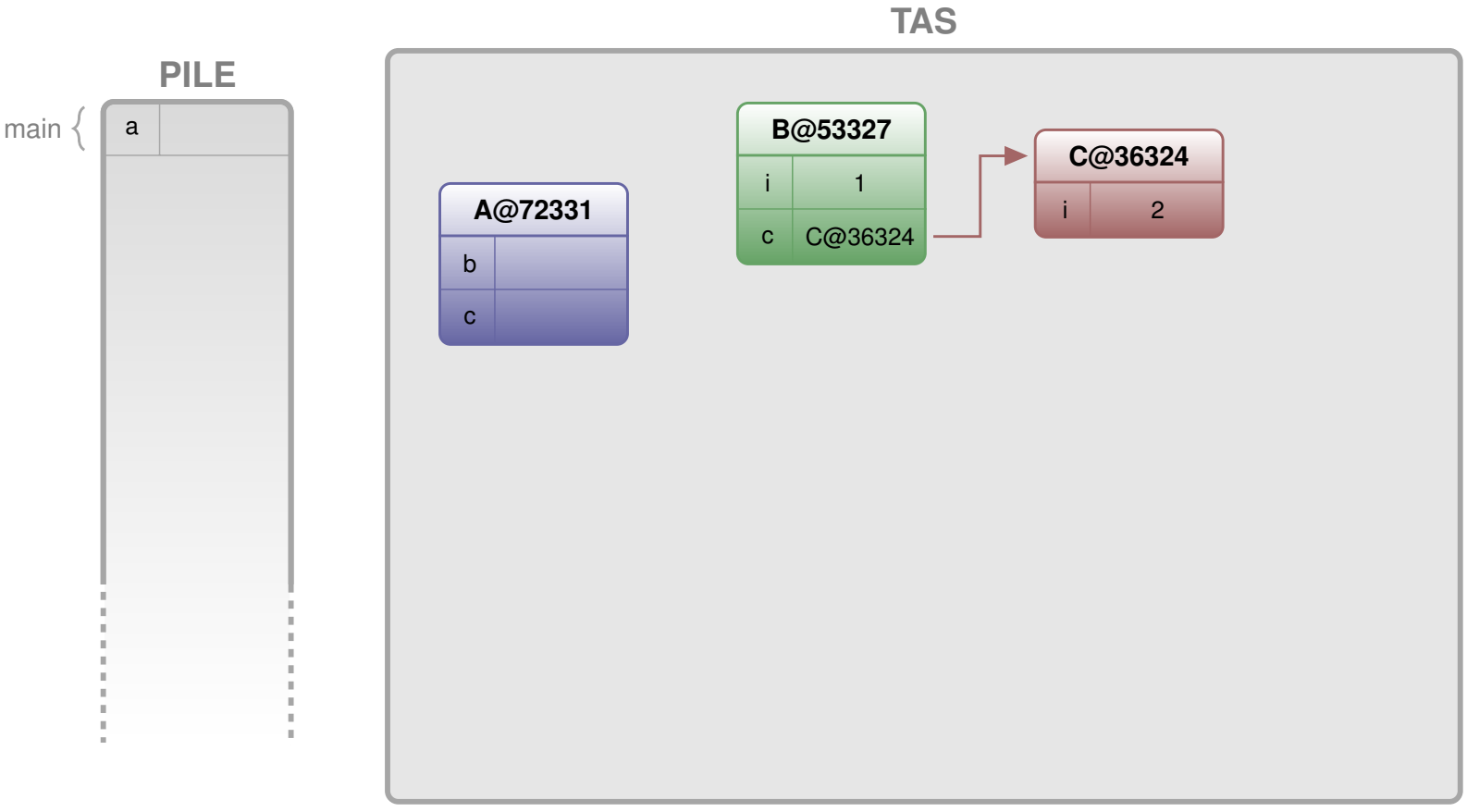
Premier exemple d'allocation.



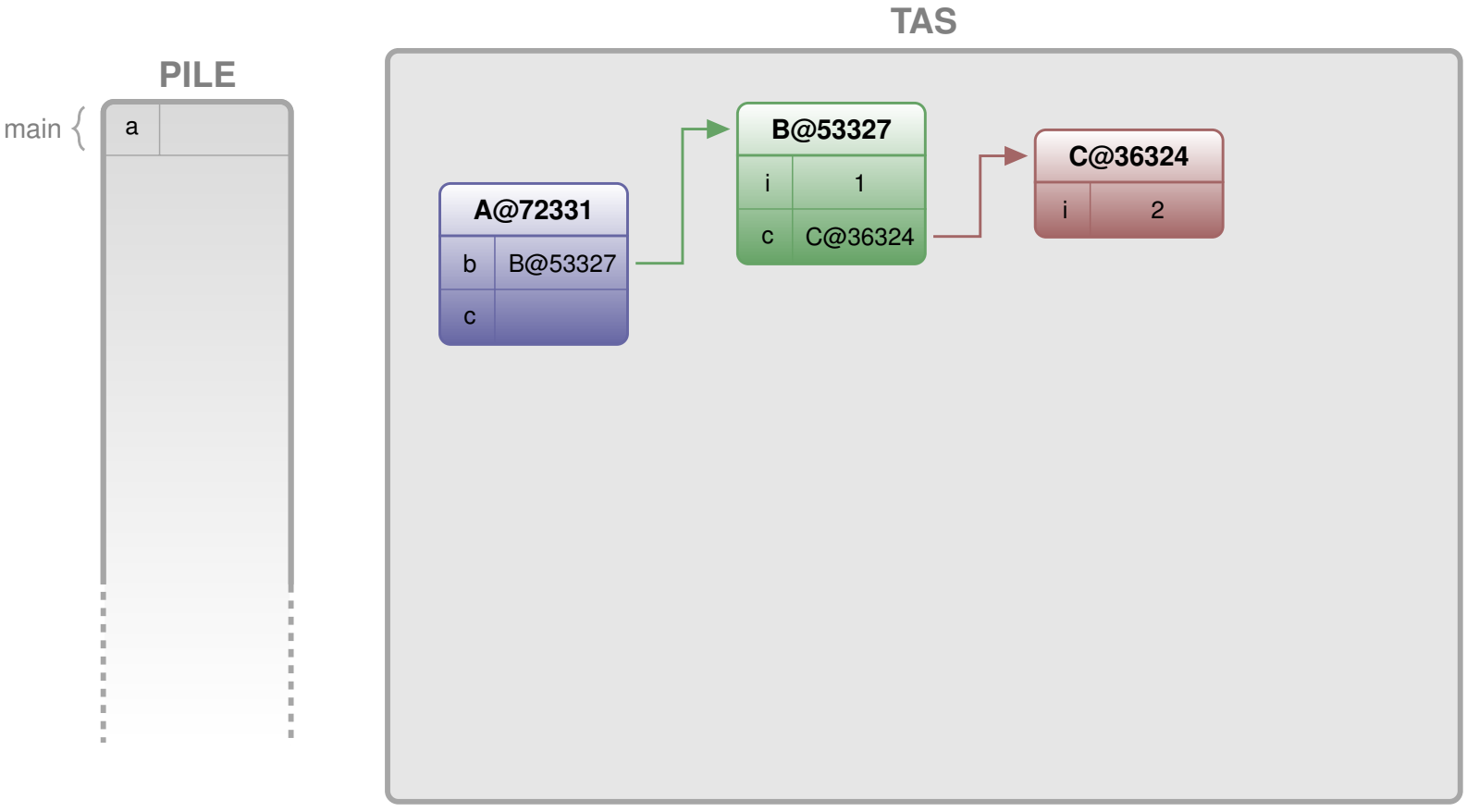
Premier exemple d'allocation.



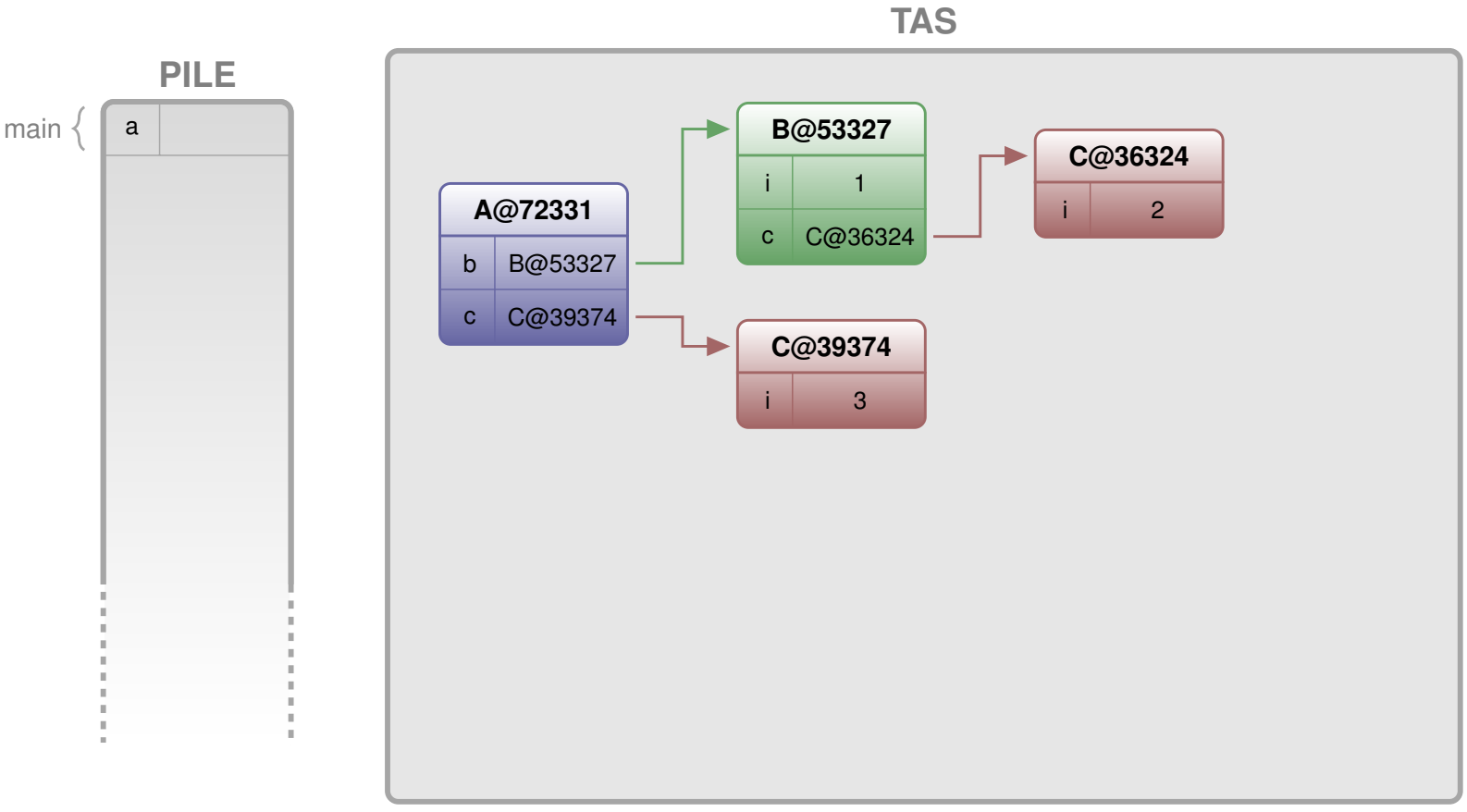
Premier exemple d'allocation.



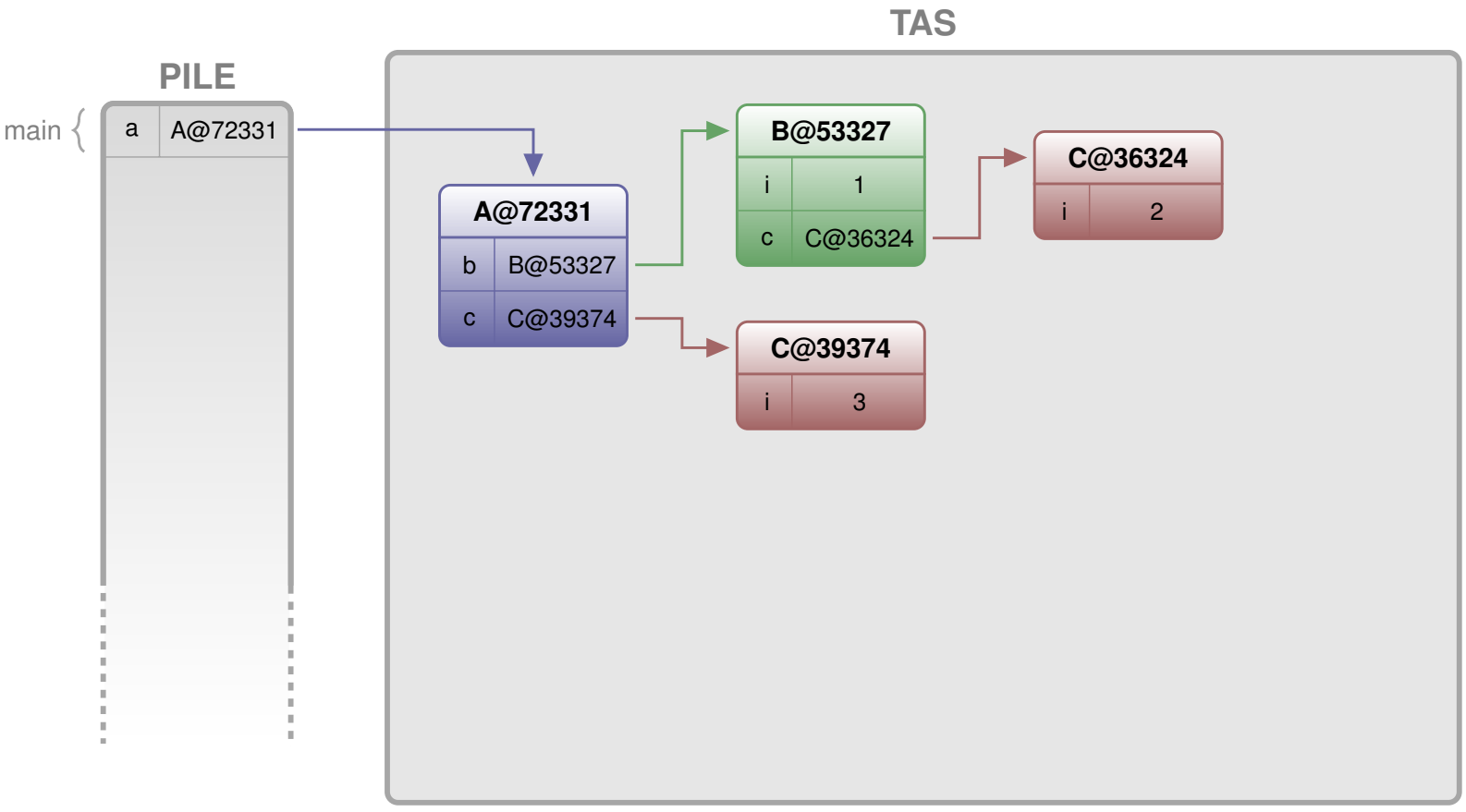
Premier exemple d'allocation.



Premier exemple d'allocation.



Premier exemple d'allocation.



Outline

Affectations et références.

```
public class A {
  B b ;
  C c ;
  public A() {
    b = new B(1,2);
    c = new C(3);
  }
}
```

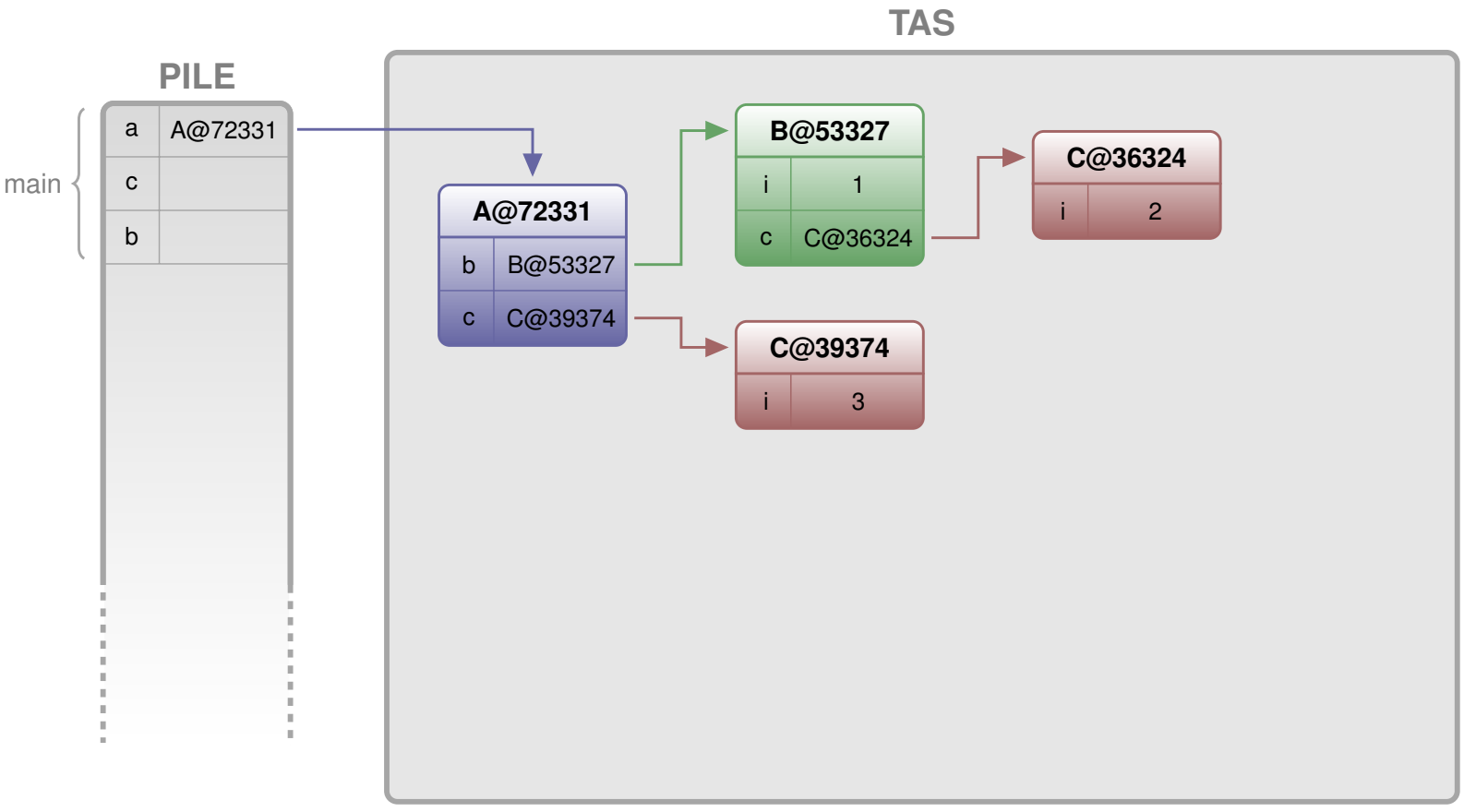
```
public class C {
  int i;
  public C(int y) {
    i = y;
  }
}
```

```
public class B {
  int i;
  C c;
  public B(C c){
    i = 0;
    this.c = c;
  }
  public B(int x,int y){
    i = x;
    c = new C(y);
  }
}
```

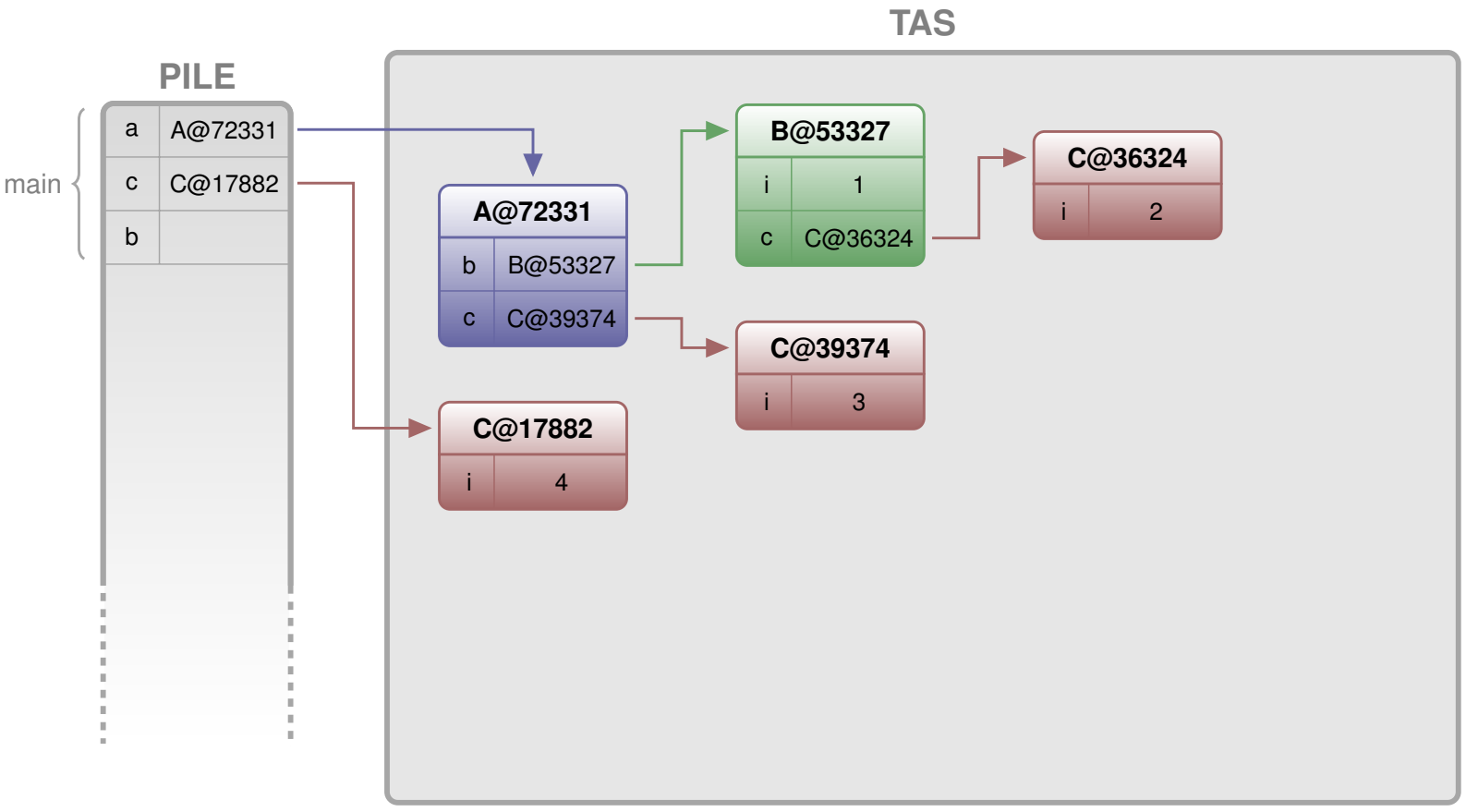
```
public class Main {
  public static void main(String[] args) {
    A a = new A();
    C c = new C(4);
    B b = new B(a.b.c);
    a.b.c = a.c ;
    a.c = c ;
  }
}
```

Affectations et références.

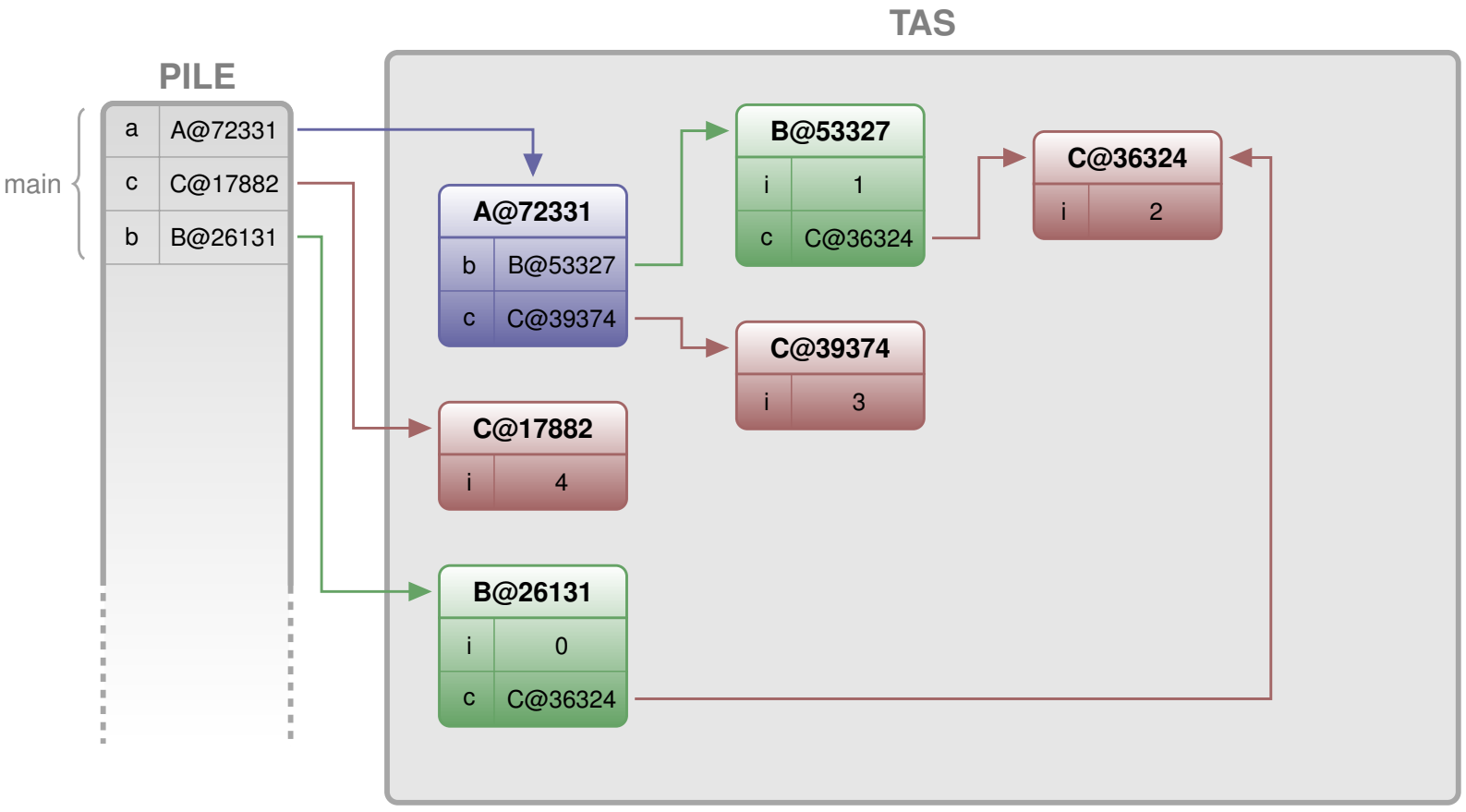
Affectations et références.



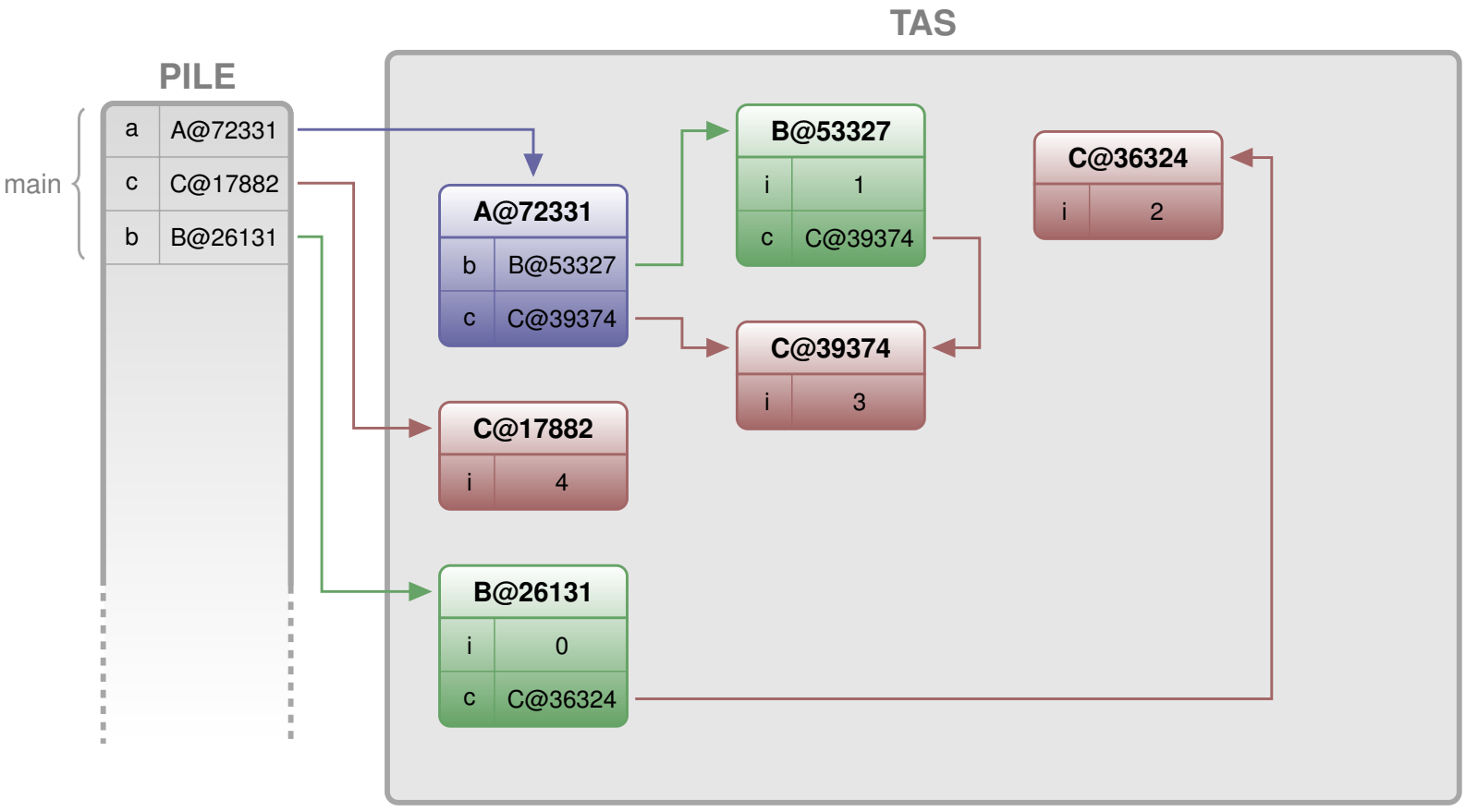
Affectations et références.



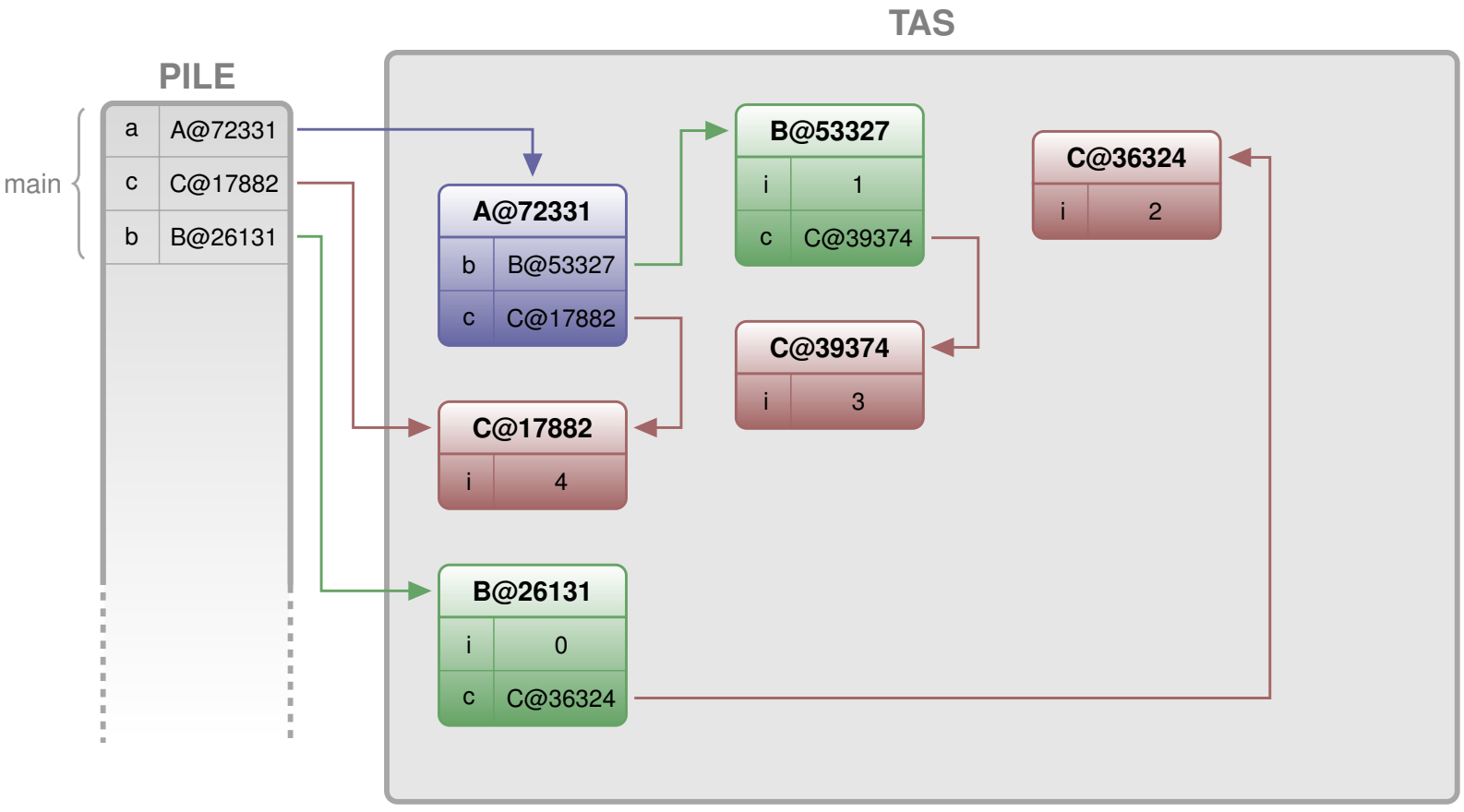
Affectations et références.



Affectations et références.



Affectations et références.



Outline

Comparaison avec ==.

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
}
```

```
public class C {
  int i;
  public C(int y) {
    i = y;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b1 = new B(1,2);
    B b2 = new B(1,2);
    if ( b1 == b2 ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
```

Comparaison avec ==.

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
}
```

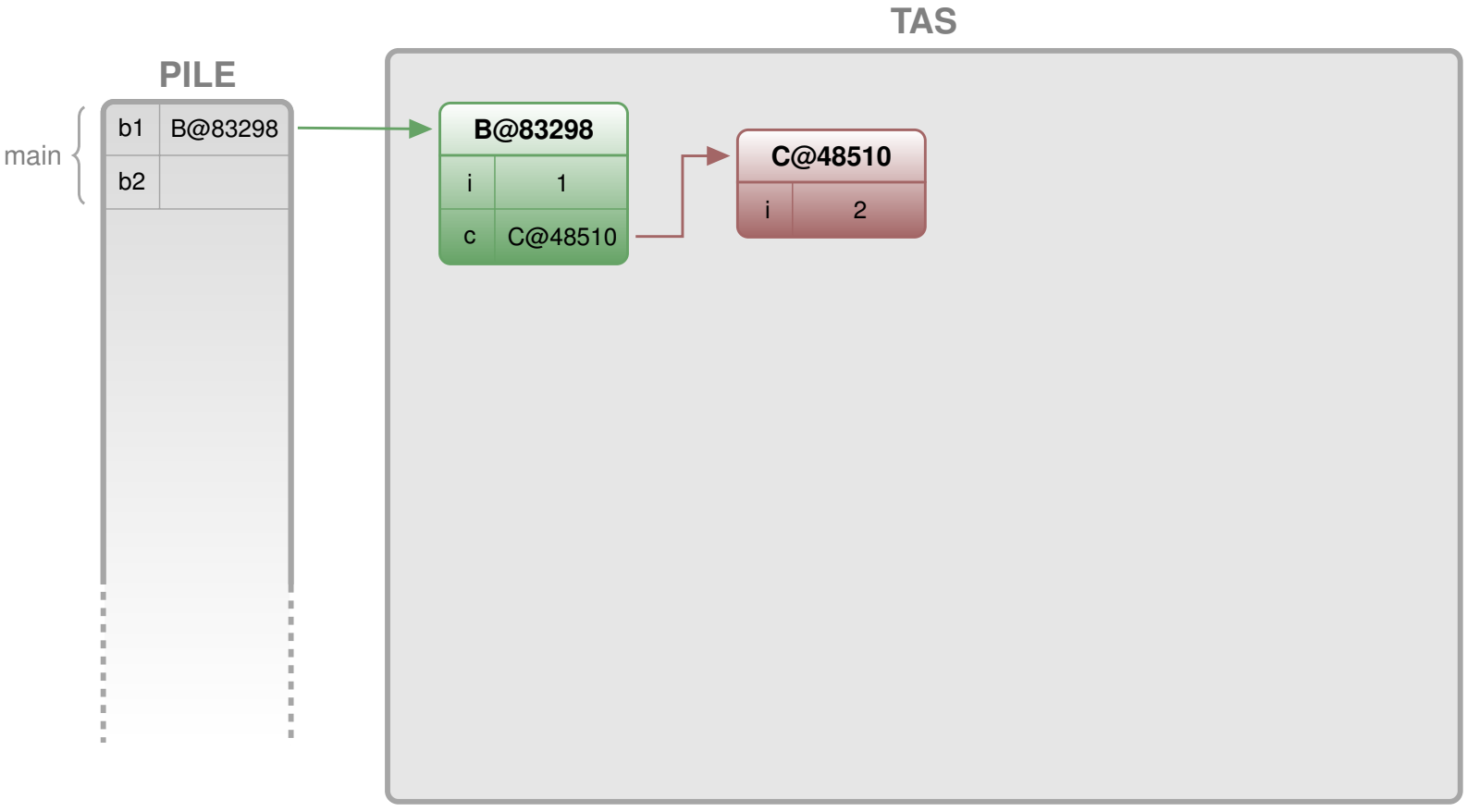
```
public class C {
  int i;
  public C(int y) {
    i = y;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b1 = new B(1,2);
    B b2 = new B(1,2);
    if ( b1 == b2 ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

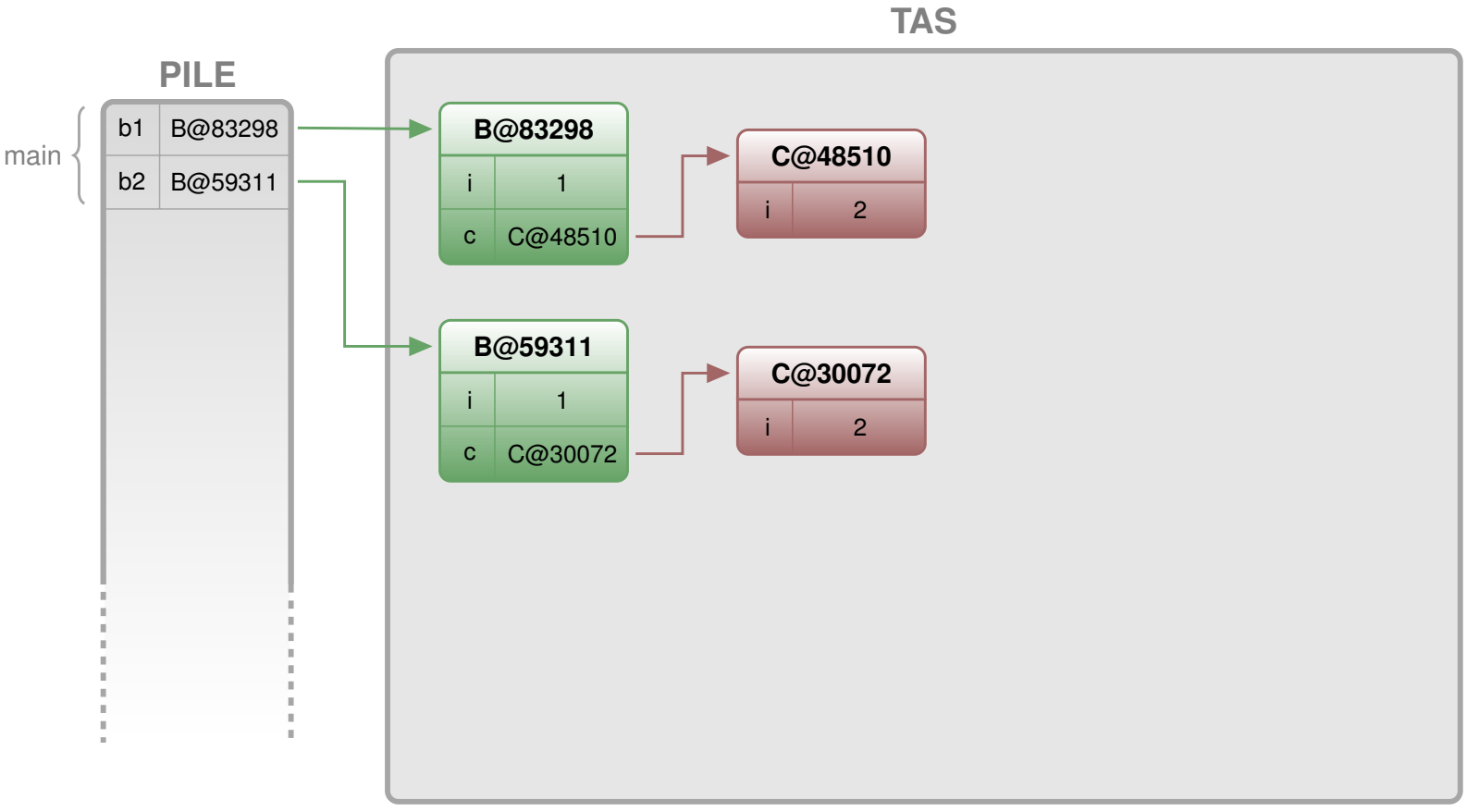
```
java Main
     Différents !
```

Comparaison avec ==.

Comparaison avec ==.



Comparaison avec ==.



Comparaison avec equals.

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
}
```

```
public class C {
  int i;
  public C(int y) {
    i = y;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b1 = new B(1,2);
    B b2 = new B(1,2);
    if ( b1.equals(b2) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
```

Comparaison avec equals.

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
}
```

```
public class C {
  int i;
  public C(int y) {
    i = y;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b1 = new B(1,2);
    B b2 = new B(1,2);
    if ( b1.equals(b2) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
     Différents !
```

Comparaison avec redéfinition des equals.

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return ((B)obj).c.equals(this.c)
           && ((B)obj).i == this.i ;
  }
}
```

```
public class C {
  int i;
  public C(int y) {
    i = y;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b1 = new B(1,2);
    B b2 = new B(1,2);
    if ( b1.equals(b2) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
```

Comparaison avec redéfinition des equals.

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return ((B)obj).c.equals(this.c)
           && ((B)obj).i == this.i ;
  }
}
```

```
public class C {
  int i;
  public C(int y) {
    i = y;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b1 = new B(1,2);
    B b2 = new B(1,2);
    if ( b1.equals(b2) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
     Différents !
```

Comparaison avec redéfinition des equals.

```java
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return ((B)obj).c.equals(this.c)
           && ((B)obj).i == this.i ;
  }
}
```

```java
public class C {
  int i;
  public C(int y) {
    i = y;
  }
  @Override
  public boolean equals(Object obj) {
    return ((C)obj).i == this.i ;
  }
}
```

```java
public class Main {
  public static void main(String[] args) {
    B b1 = new B(1,2);
    B b2 = new B(1,2);
    if ( b1.equals(b2) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
```

Comparaison avec redéfinition des equals.

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return ((B)obj).c.equals(this.c)
           && ((B)obj).i == this.i ;
  }
}
```

```
public class C {
  int i;
  public C(int y) {
    i = y;
  }
  @Override
  public boolean equals(Object obj) {
    return ((C)obj).i == this.i ;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b1 = new B(1,2);
    B b2 = new B(1,2);
    if ( b1.equals(b2) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
     Identiques !
```

Attention à la redéfinition des equals

```java
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return ((B)obj).c.equals(this.c) &&
           ((B)obj).i == this.i ;
  }
}
```

```java
public class Main {
  public static void main(String[] args) {
    B b1 = new B(1,2);
    B b2 = new B(1,2);
    if ( b1.equals(b2) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
```

Attention à la redéfinition des equals

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return ((B)obj).c.equals(this.c) &&
           ((B)obj).i == this.i ;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b = new B(1,2);
    C c =  C(3);
    if ( b.equals(c) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
```

Attention à la redéfinition des equals

```java
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return ((B)obj).c.equals(this.c) &&
           ((B)obj).i == this.i ;
  }
}
```

```java
public class Main {
  public static void main(String[] args) {
    B b = new B(1,2);
    C c =  C(3);
    if ( b.equals(c) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
     Exception in thread "main" java.lang.ClassCastException: C cannot be cast to B
            at comparaison.B.equals(B.java:10)
```

Attention à la redéfinition des equals

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return obj instanceof B &&
           ((B)obj).c.equals(this.c) &&
           ((B)obj).i == this.i ;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b = new B(1,2);
    C c =  C(3);
    if ( b.equals(c) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
```

Attention à la redéfinition des equals

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return obj instanceof B &&
           ((B)obj).c.equals(this.c) &&
           ((B)obj).i == this.i ;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b = new B(1,2);
    C c =  C(3);
    if ( b.equals(c) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
     Différents !
```

Comparaison plus efficace : attention à l'ordre

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return obj instanceof B &&
           ((B)obj).c.equals(this.c) &&
           ((B)obj).i == this.i ;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b1 = new B(1,2);
    B b2 = new B(2,2);
    if ( b1.equals(b2) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

Comparaison plus efficace : attention à l'ordre

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return obj instanceof B &&
           ((B)obj).i == this.i &&
           ((B)obj).c.equals(this.c) ;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b1 = new B(1,2);
    B b2 = new B(2,2);
    if ( b1.equals(b2) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

Comparaison encore plus efficace : auto-comparaison

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return  obj instanceof B &&
           ((B)obj).i == this.i &&
           ((B)obj).c.equals(this.c) ;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b = new B(1,2);
    if ( b.equals(b) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
```

Comparaison encore plus efficace : auto-comparaison

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return  obj instanceof B &&
           ((B)obj).i == this.i &&
           ((B)obj).c.equals(this.c) ;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b = new B(1,2);
    if ( b.equals(b) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

```
java Main
     Identique !
```

Comparaison encore plus efficace : auto-comparaison

```
public class B {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public boolean equals(Object obj) {
    return this == obj ? true :
           obj instanceof B &&
           ((B)obj).i == this.i &&
           ((B)obj).c.equals(this.c) ;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    B b = new B(1,2);
    if ( b.equals(b) ) {
      System.out.println("Identiques !");
    } else {
      System.out.println("Différents !");
    }
  }
}
```

Outline

Duplication d'objet avec constructeur par copie.

```
public class Main {
  public static void main(String[] args) {
    A a1 = new A(1,2);
    A a2 = new A(a1);
  }
}
```

```
public class A {
  B b ;
  C c ;
  public A() {
    b = new B(1,2);
    c = new C(3);
  }
  public A(A a) {
    this.b = new B(a.b);
    this.c = new C(a.c);
  }
}
```

```
  public class B {
  int i;
  C c;
  public B(int x,C y) {
    i = x;
    c = y;
  }
  public B(int x,int y){
    this(x,new C(y));
  }
  public B (B b) {
    this(b.i,new C(b.c);
  }
}
```

```
public class C {
  int i;
  public C(int y) {
    i = y;
  }
  public C(C c) {
    this(c.i);
  }
}
```

Duplication d'objet par appel à la méthode clone.

```
public class Main {
  public static void main(String[] args) {
    A a1 = new A(1,2);
    A a2 = (A) a1.clone();
  }
}
```

```
public class A implements Cloneable {
  B b ;
  C c ;
  public A() {}
  public A(int x,int y) {
    b = new B(x,y);
    c = new C(x);
  }
  @Override
  public Object clone() {
    A a = new A();
    a.b = (B) this.b.clone();
    a.c = (C) this.c.clone();
    return (Object) a;
  }
}
```

```
public class B implements Cloneable {
  int i;
  C c;
  public B(int x,int y){
    i=x;
    c = new C(y);
  }
  @Override
  public Object clone() {
    return (Object) new B(i,c.i);
  }
}
```

```
public class C implements Cloneable {
  int i;
  public C(int y) {
    i = y;
  }
  @Override
  public Object clone() {
    C c = new C(this.i) ;
    return (Object) c ;
  }
}
```

Problème de la duplication d'un graphe d'objets.

```
public class X {
  Y y1,y2 ;
  public X() {
    Z z = new Z();
    this.y1 = new Y(z);
    this.y2 = new Y(z);
  }
  public X(X x) {
    this.y1 = new Y(x.y1);
    this.y2 = new Y(x.y2);
  }
}
```

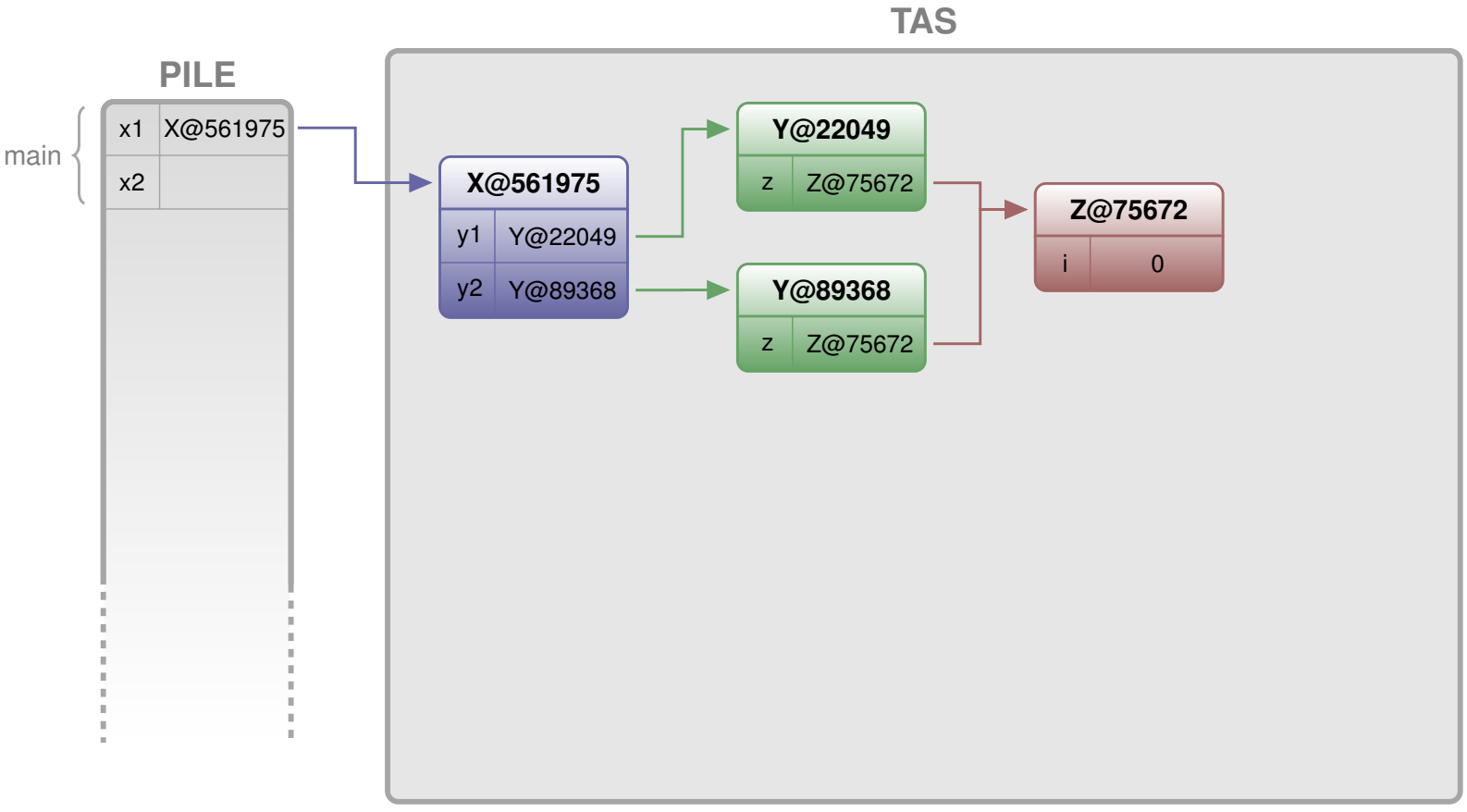
```
public class Y {
  Z z ;
  public Y(Z z) {
    this.z = z;
  }
  public Y(Y y) {
    this.z = new Z(y.z);
  }
}
```

```
public class Z {
      int i = 0 ;
      public Z () {}
      public Z (Z z) {}
    }
```

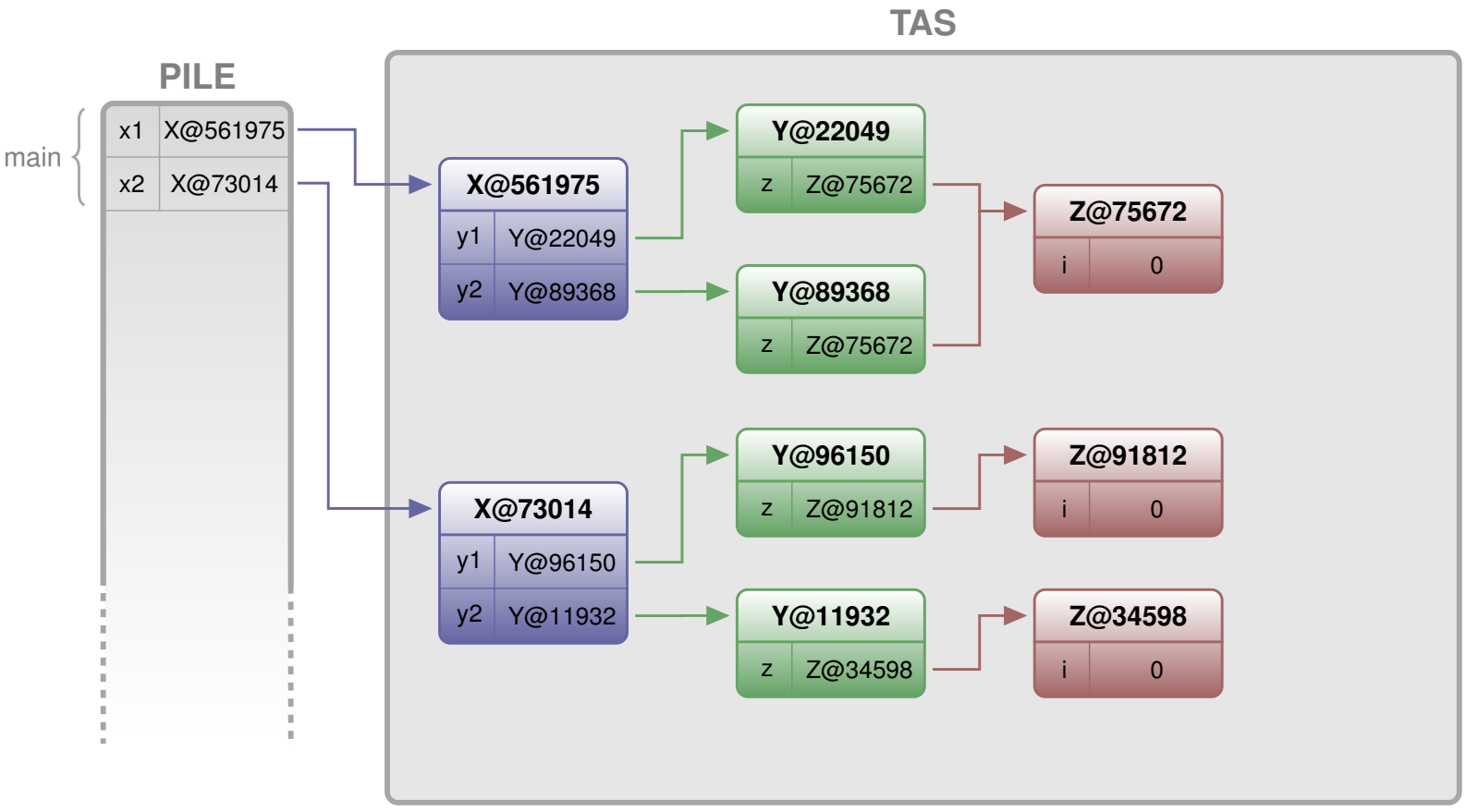
```
public class Main {
  public static void main(String[] args) {
    X x1 = new X();
    X x2 = new X(x1);
  }
}
```

Problème de la duplication d'un graphe d'objets.

Problème de la duplication d'un graphe d'objets.

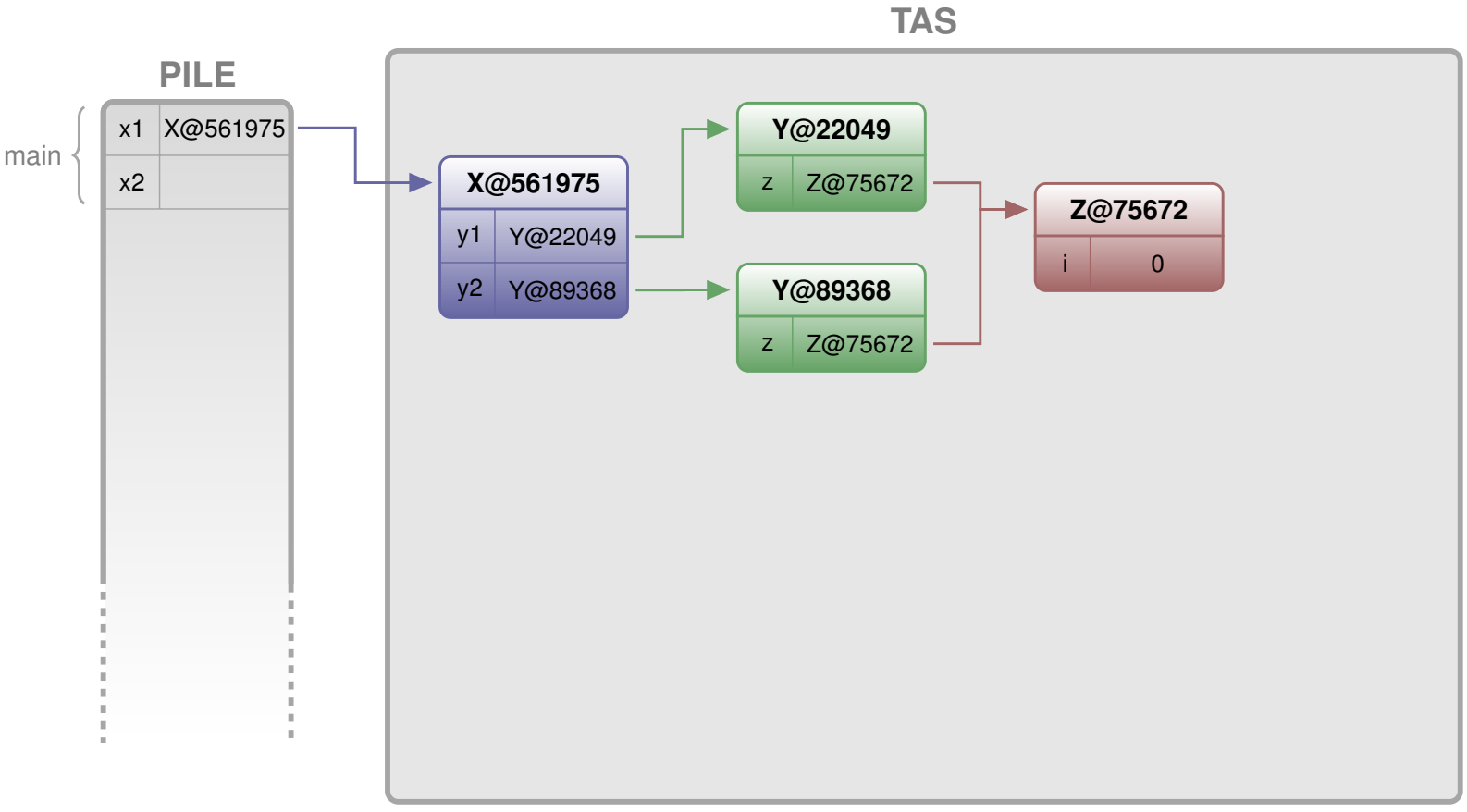


Problème de la duplication d'un graphe d'objets.

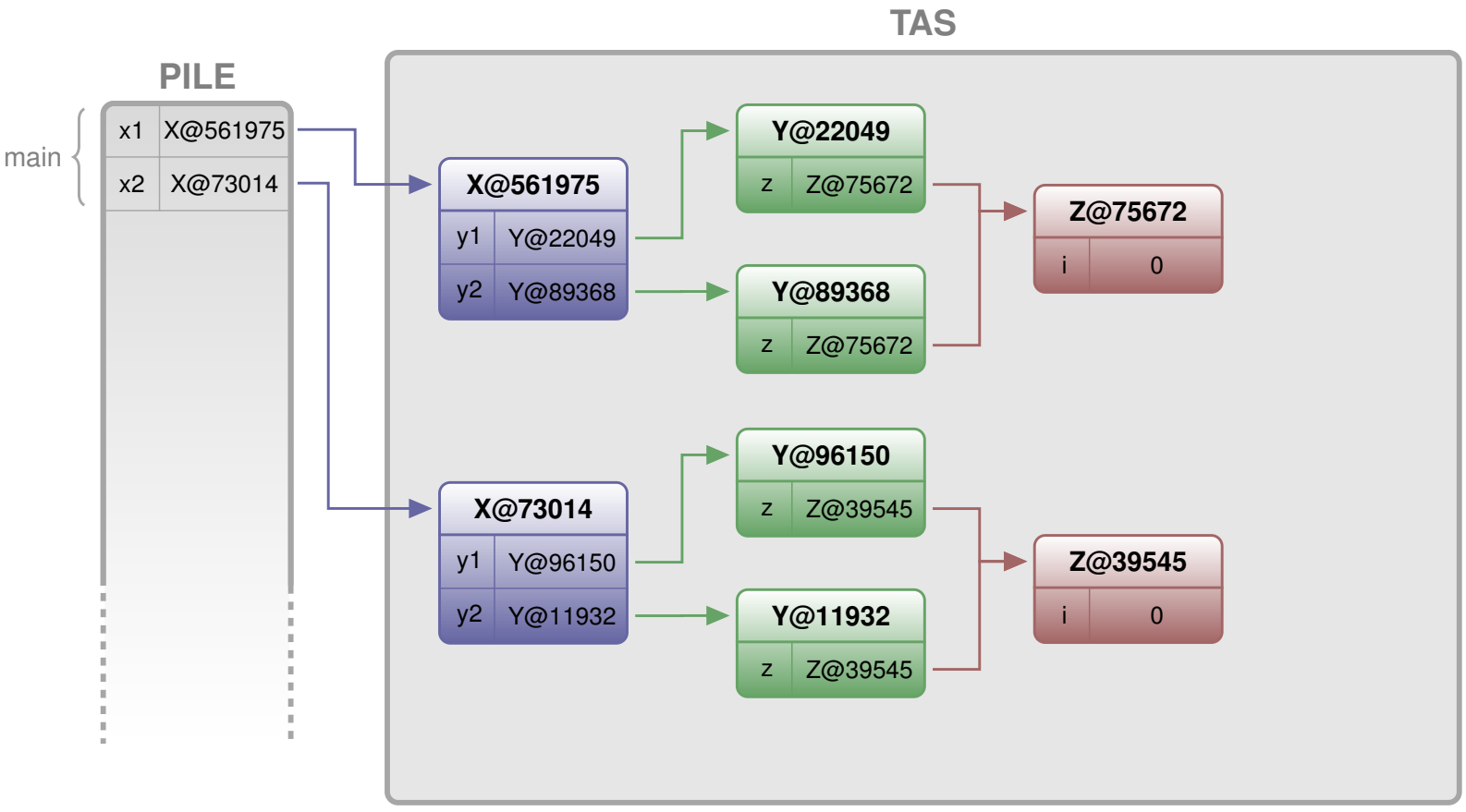


Duplication par sérialisation.

Duplication par sérialisation.



Duplication par sérialisation.



Outline

Outline

Appels successifs de méthodes

```
public class A {
  int x;
  B b ;
  public void f (int x) {
    this.x = x ;
    this.b = new B(x);
  }
  public B g (B b) {
    B ret = this.b ;
    this.b = b;
    return ret ;
  }
  public void h (B b) {
    this.b.f(b,this.x);
    this.b.f(b,this.x);
  }
}
```

```
public class B {
  int i;
  public B(int i) {
    this.i = i;
   }
  public void f (B b, int x) {
    this.i = b.i ;
    x = x * 2 ;
    b.i += x ;
  }
}
```

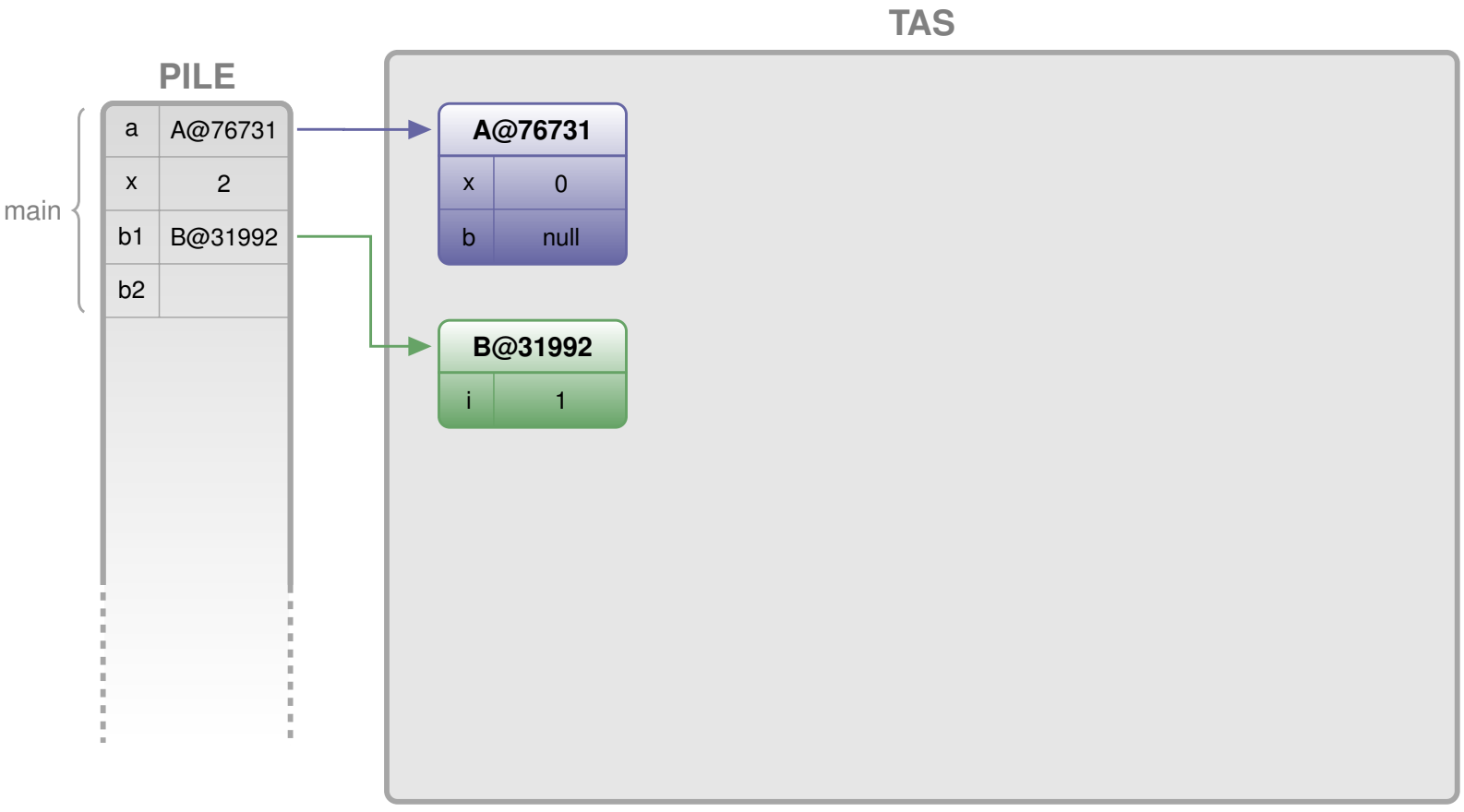
```
public class Main {
  public static void main(String[] args) {
    A a = new A();
    int x = 2;
    B b1 = new B(1);
    B b2 ;
    a.f(3);
    b2 = a.g(b1);
    a.h(x);
  }
}
```

Appels successifs de méthodes

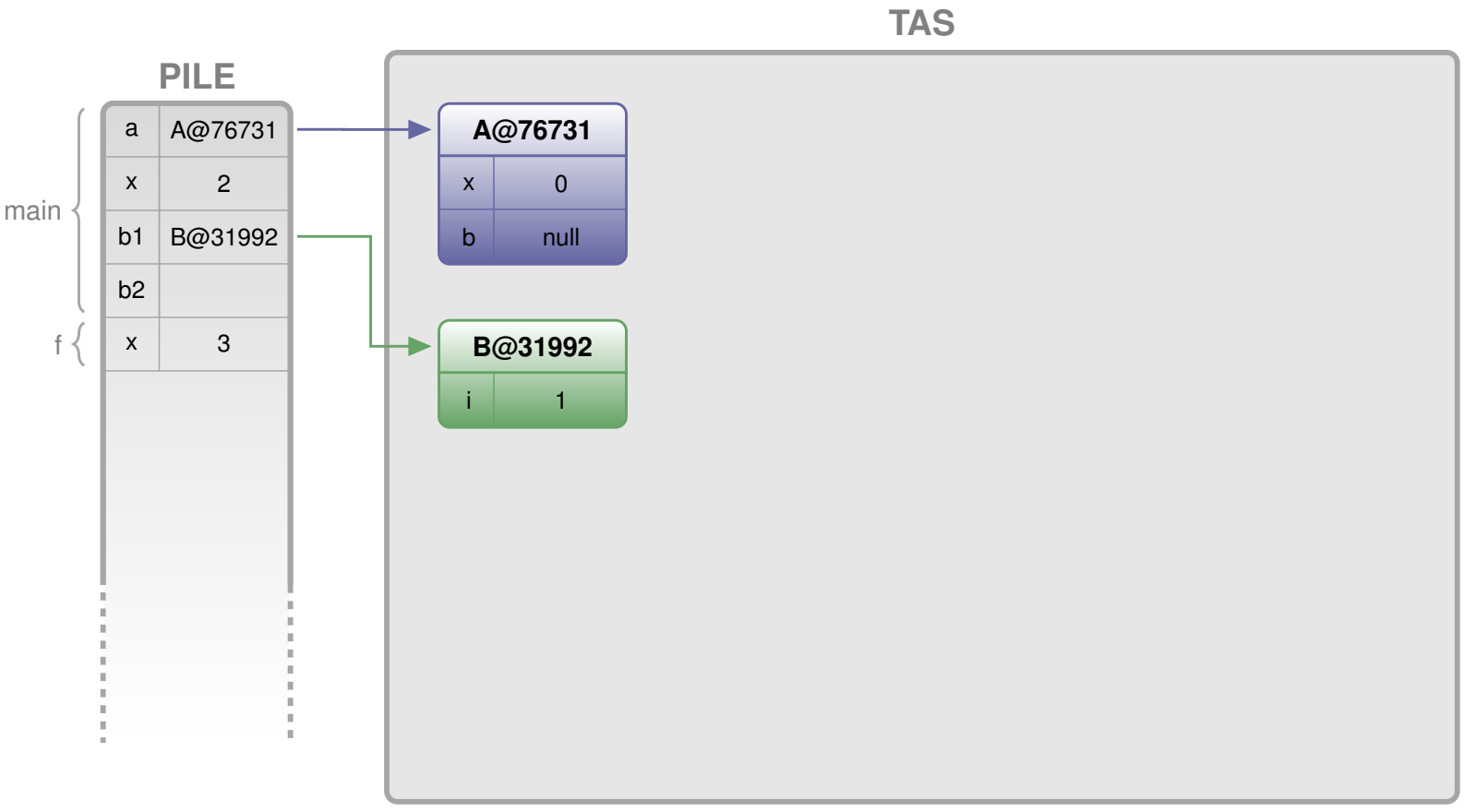
Appels successifs de méthodes

Appels successifs de méthodes

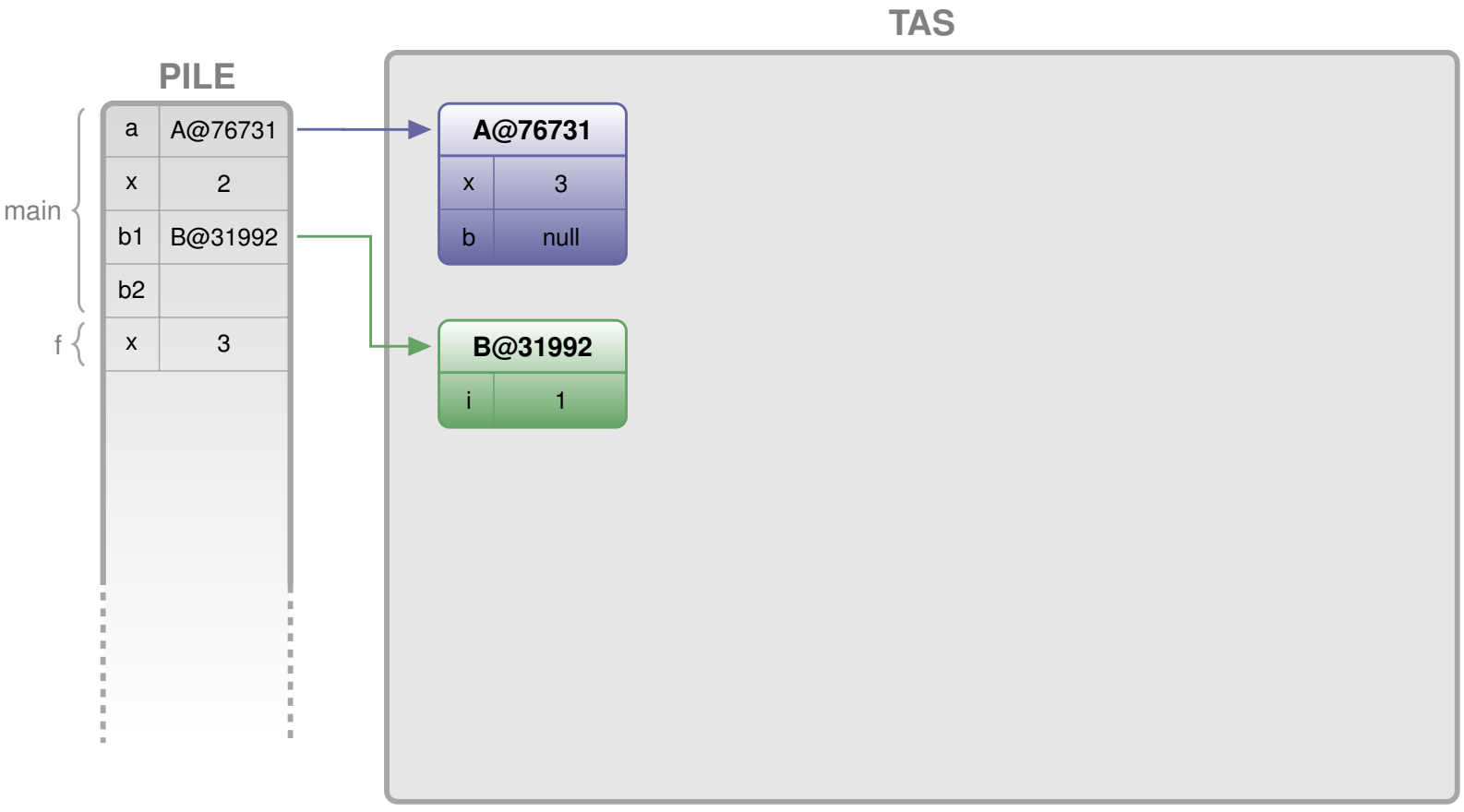
Appels successifs de méthodes



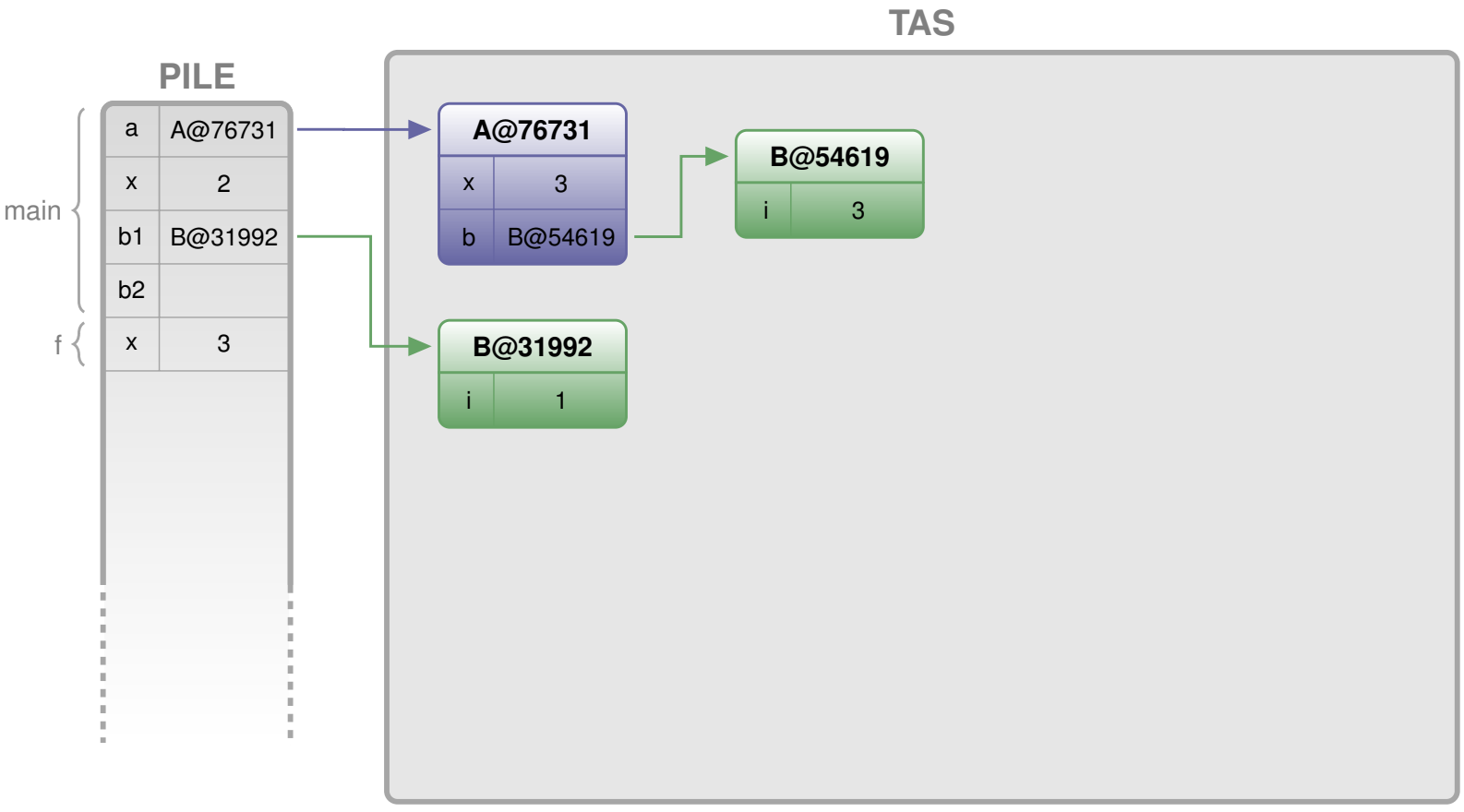
Appels successifs de méthodes



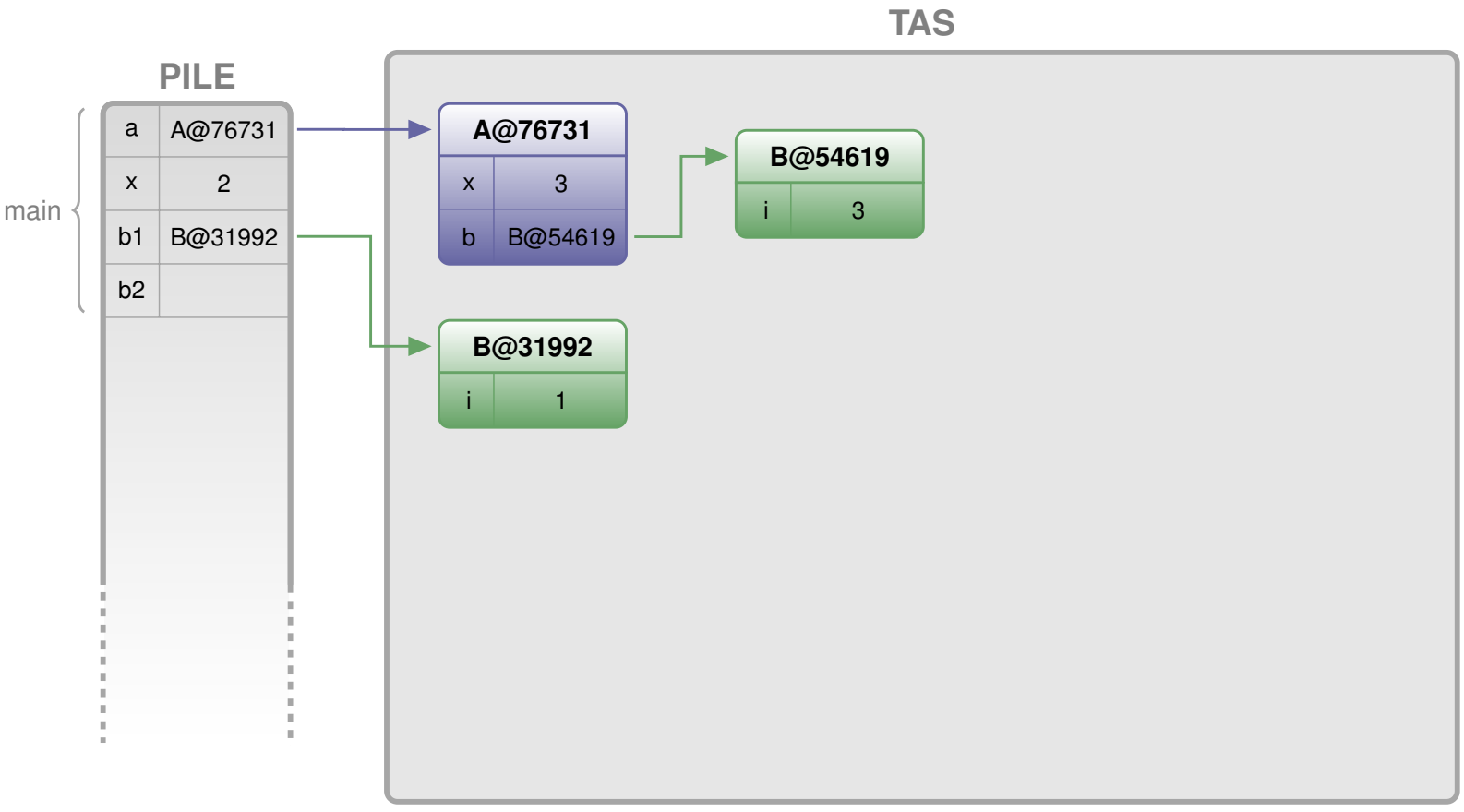
Appels successifs de méthodes



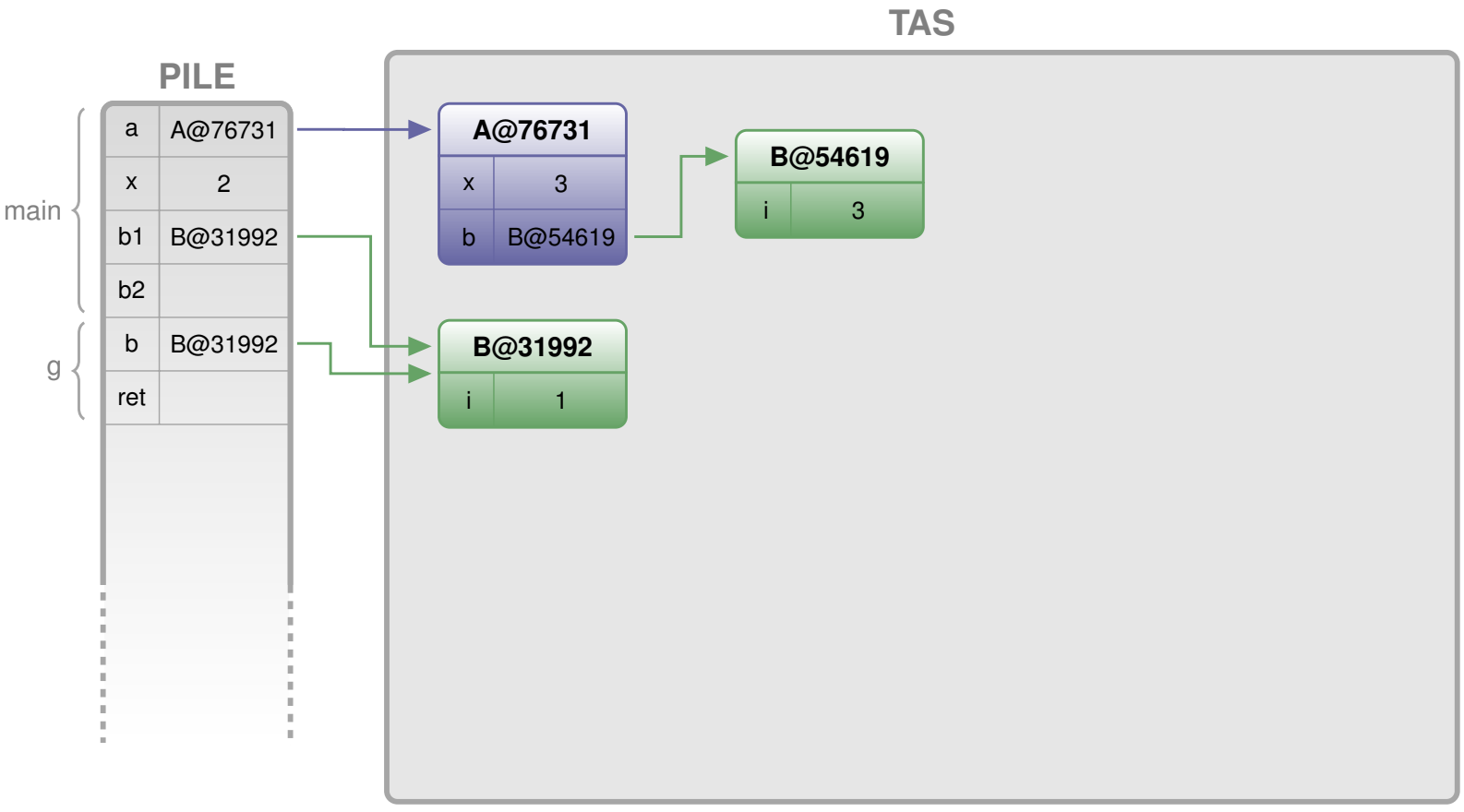
Appels successifs de méthodes



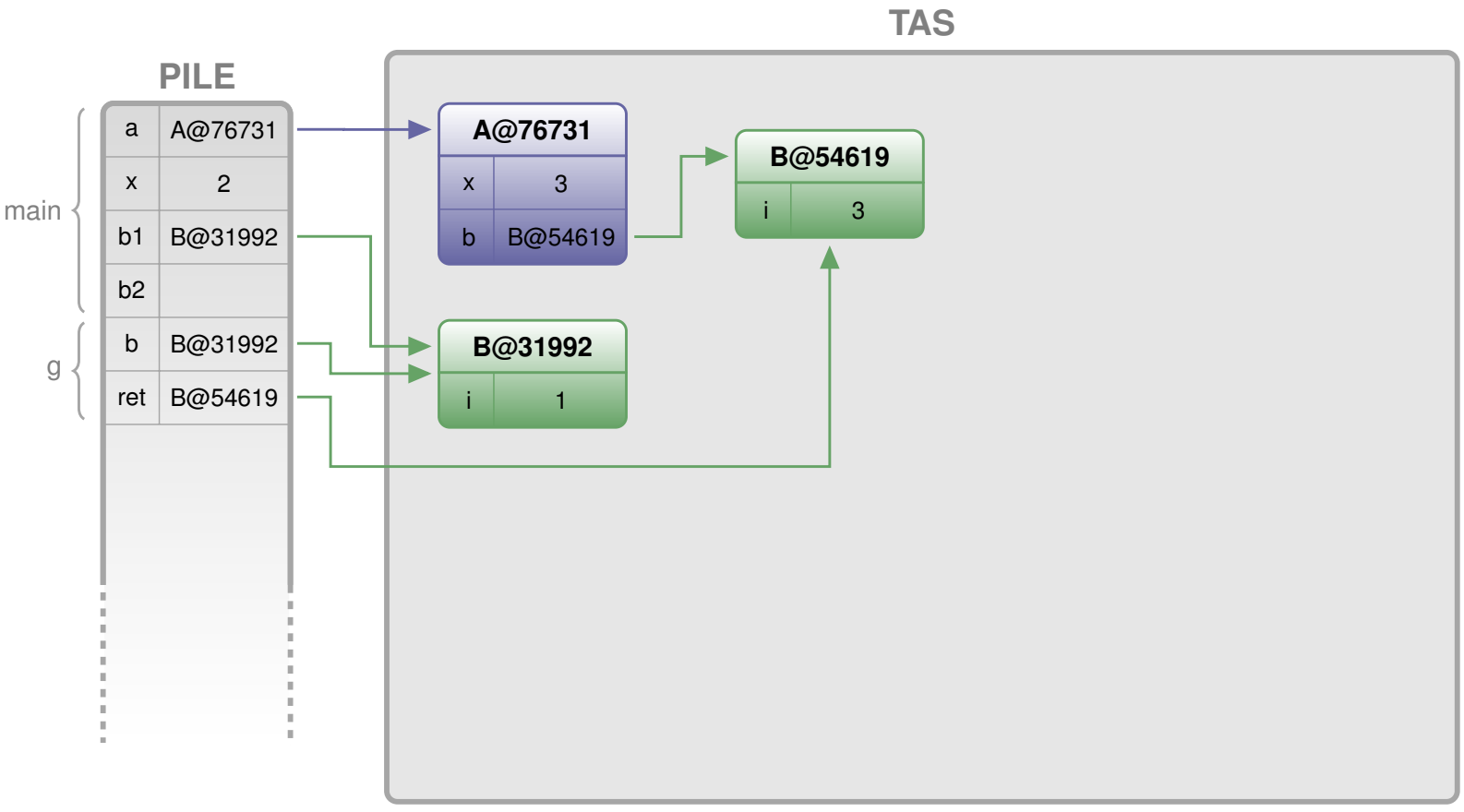
Appels successifs de méthodes



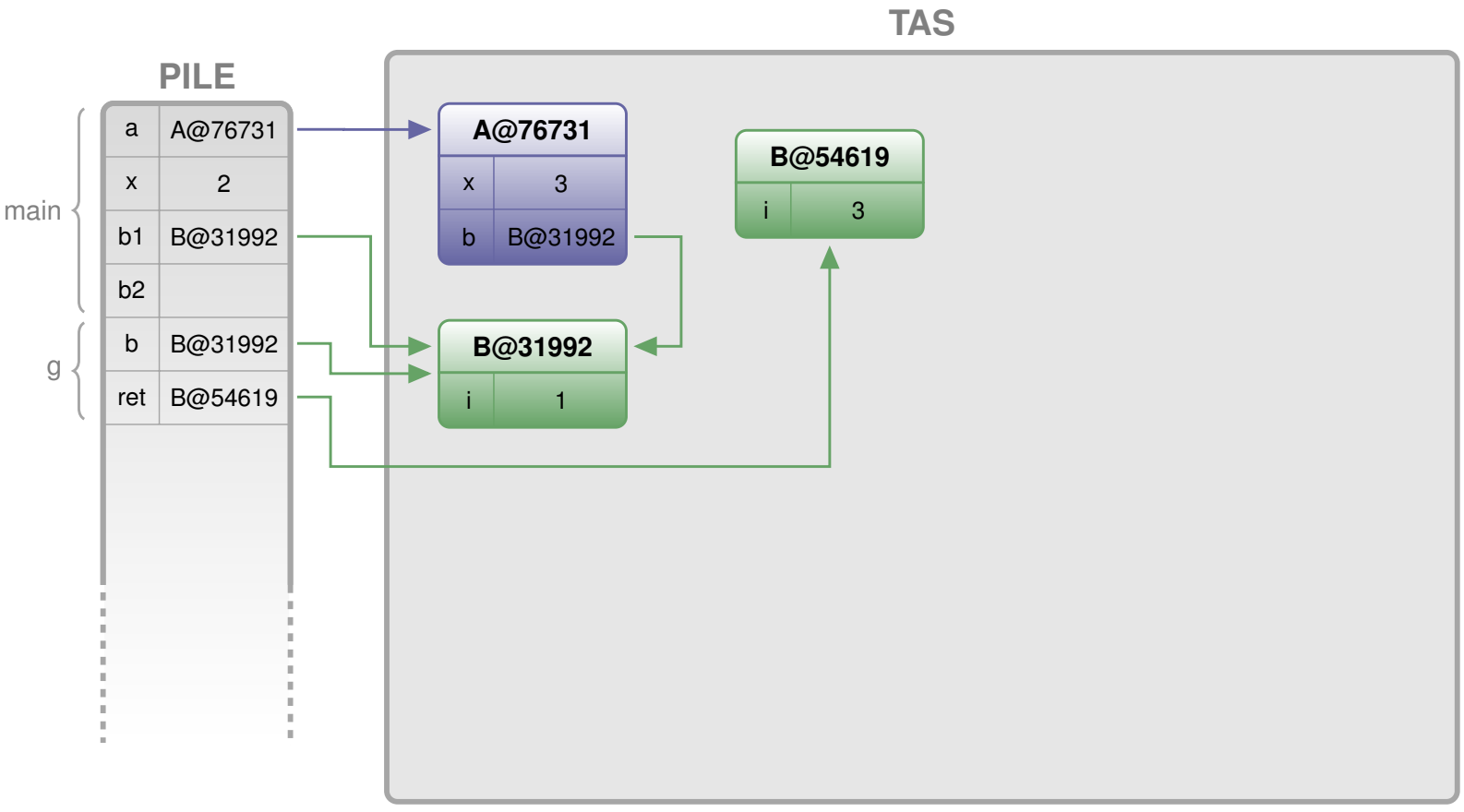
Appels successifs de méthodes



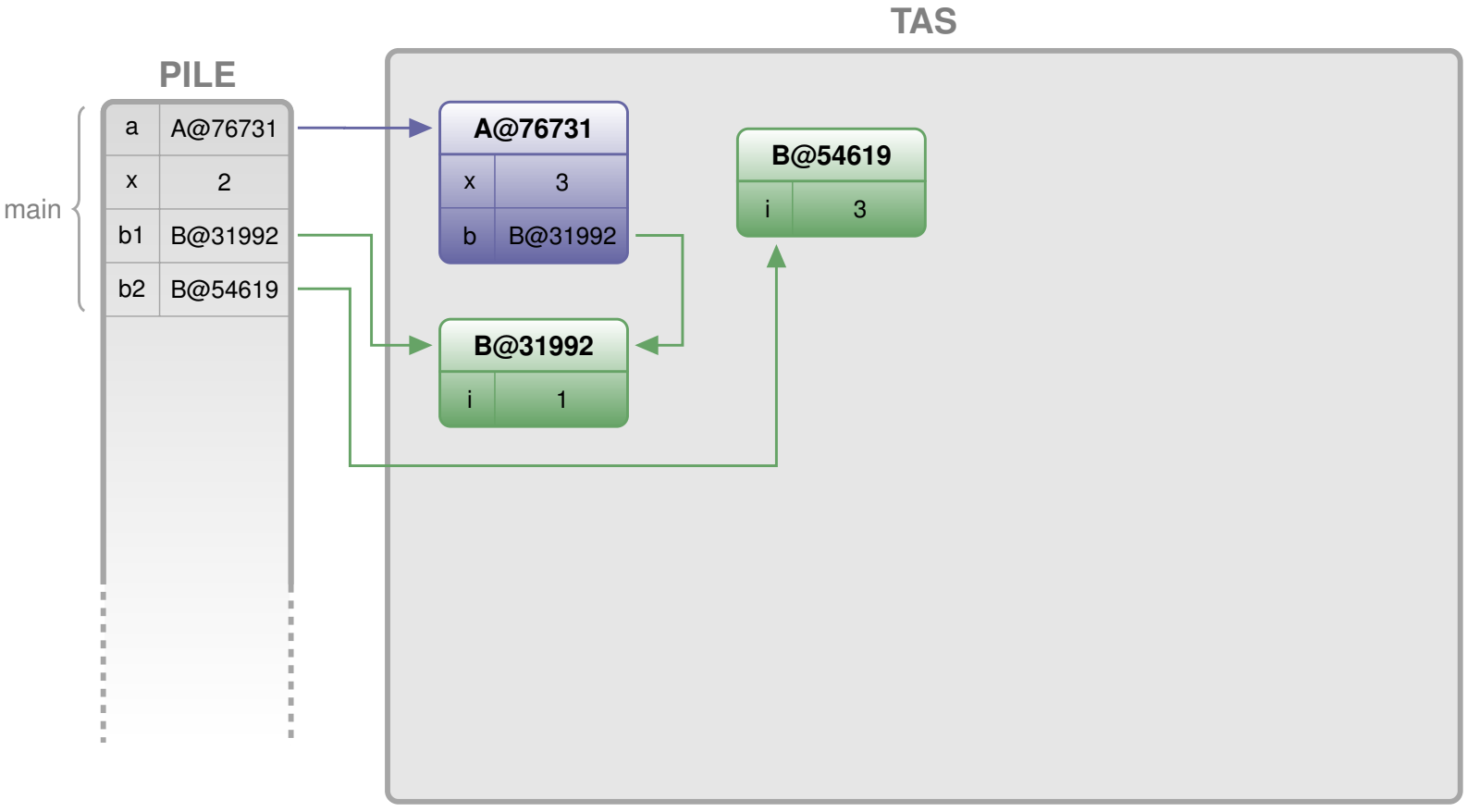
Appels successifs de méthodes



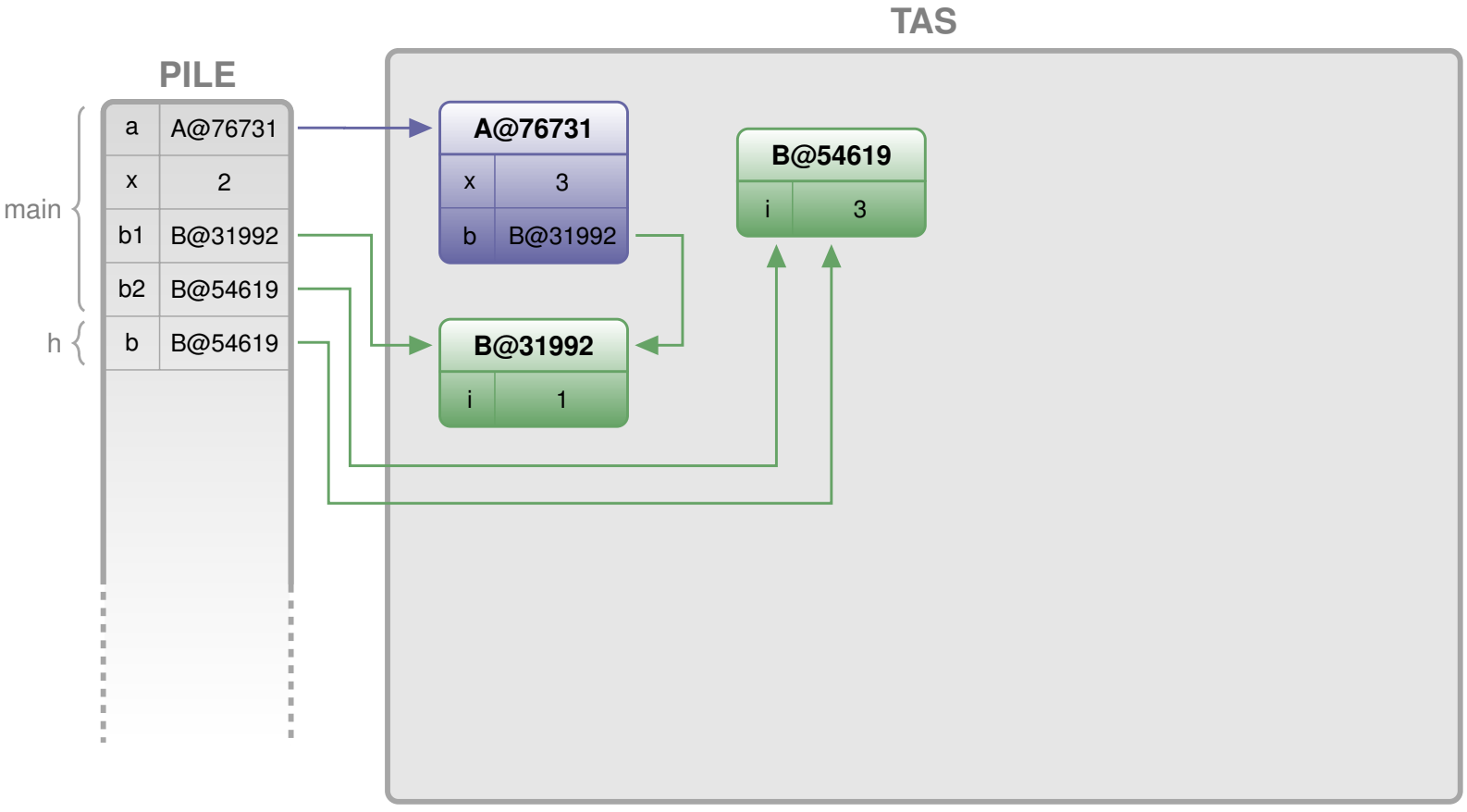
Appels successifs de méthodes



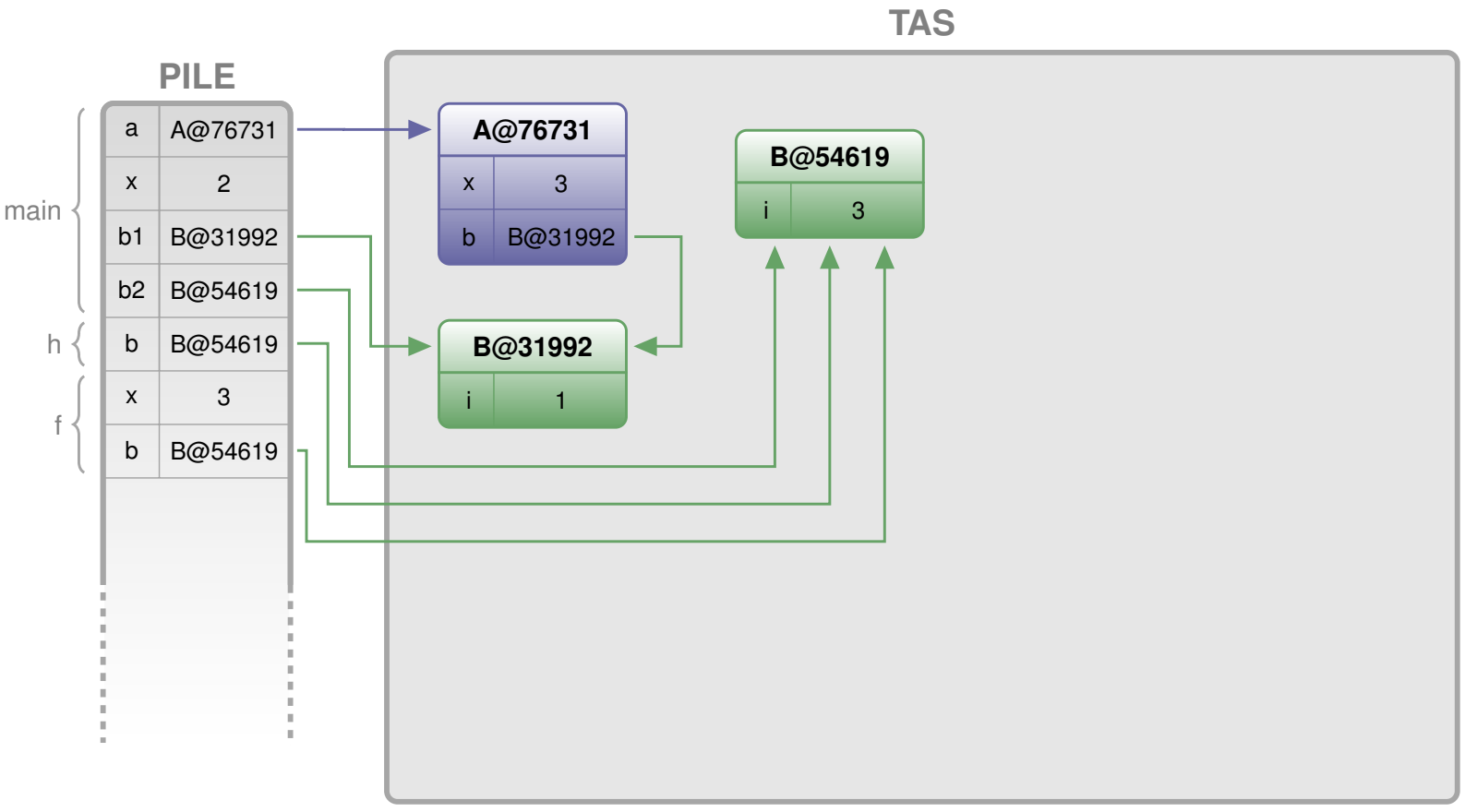
Appels successifs de méthodes



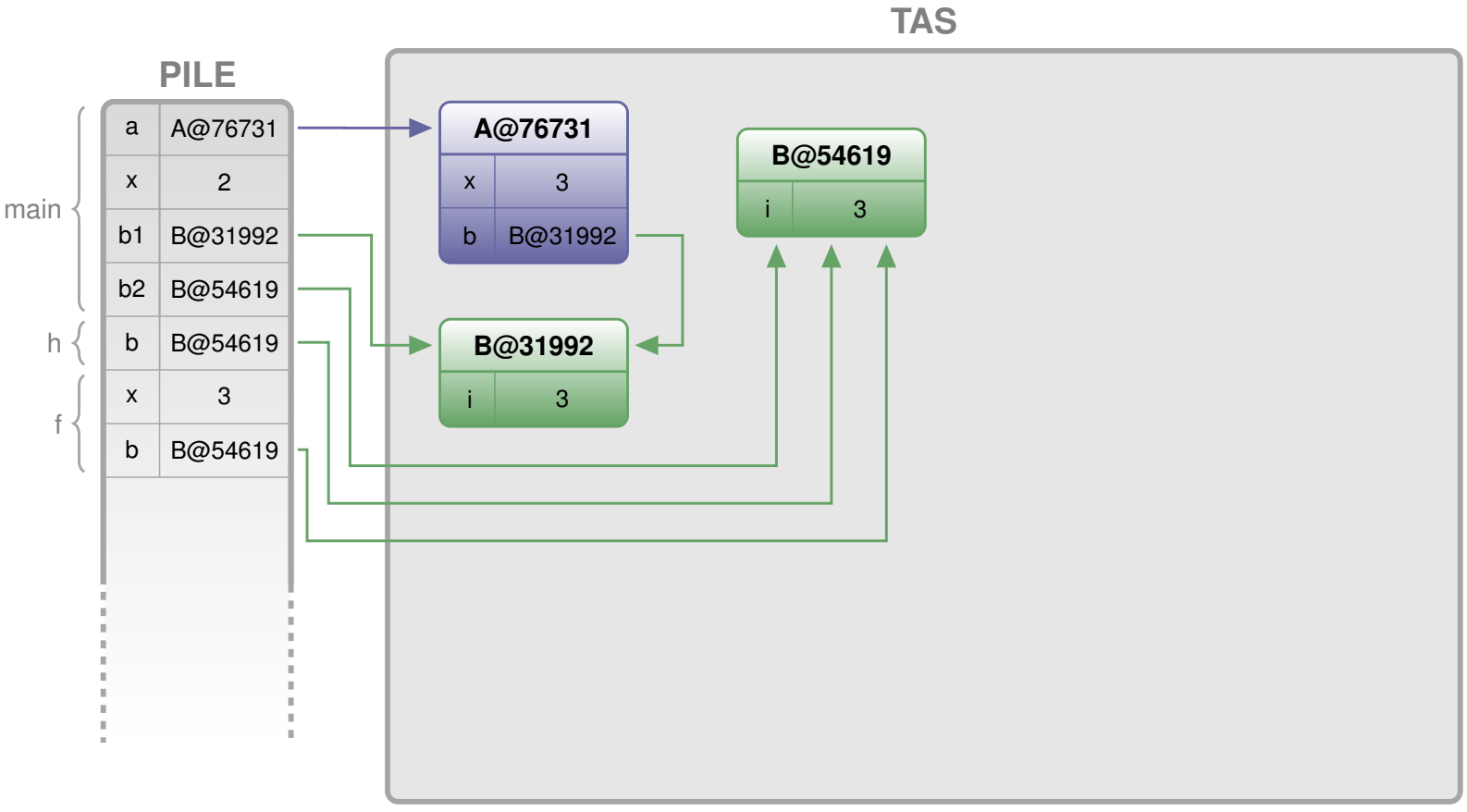
Appels successifs de méthodes



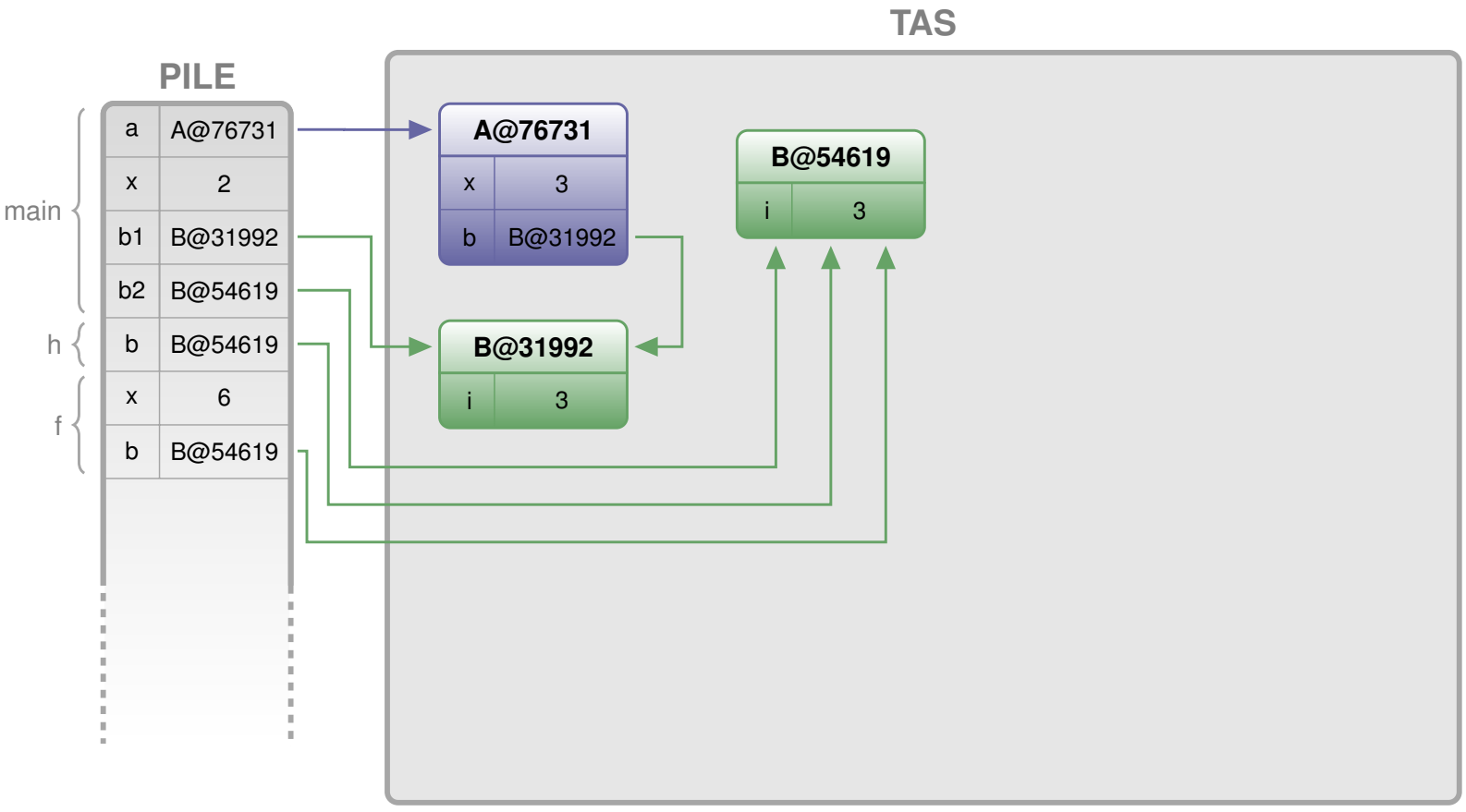
Appels successifs de méthodes



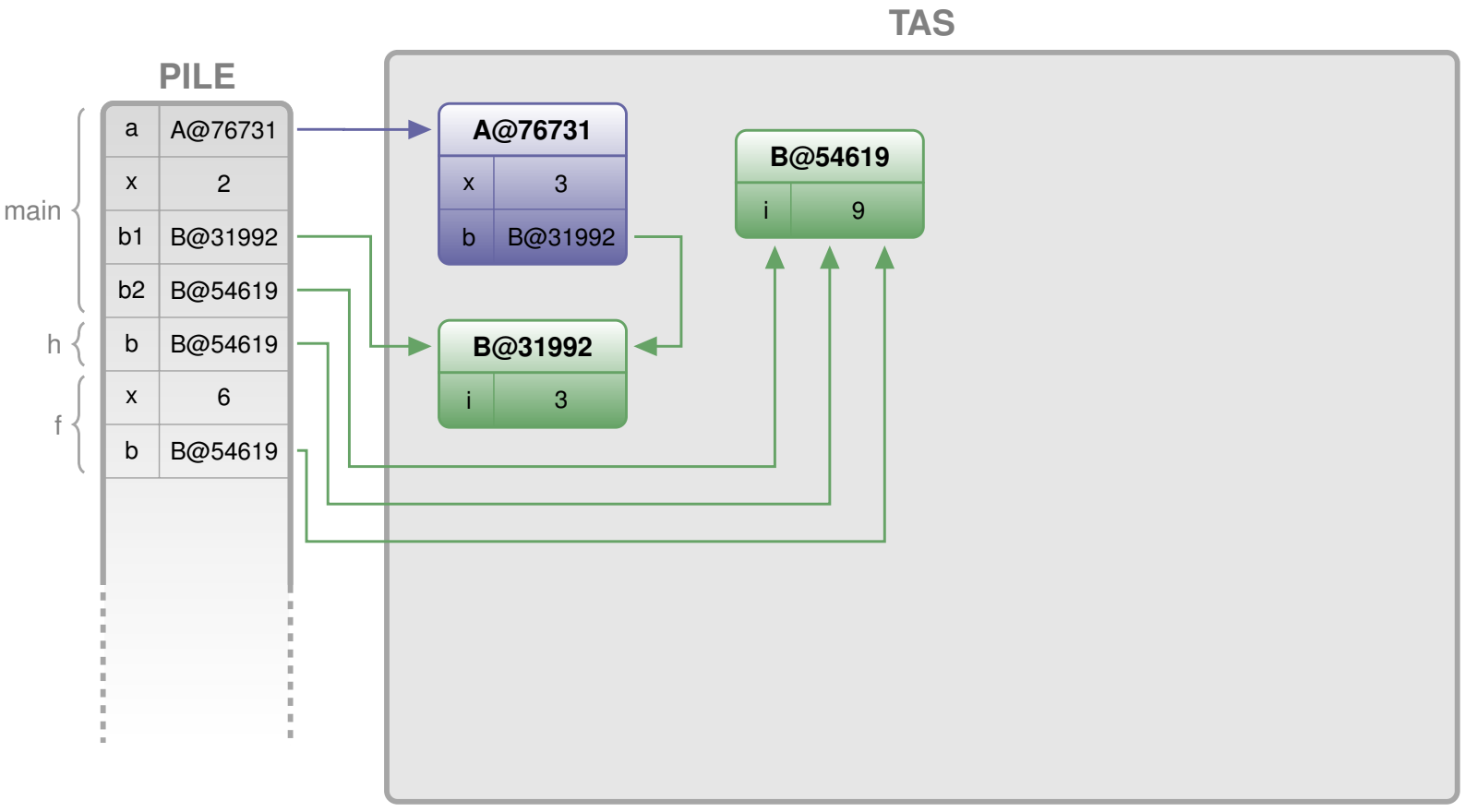
Appels successifs de méthodes



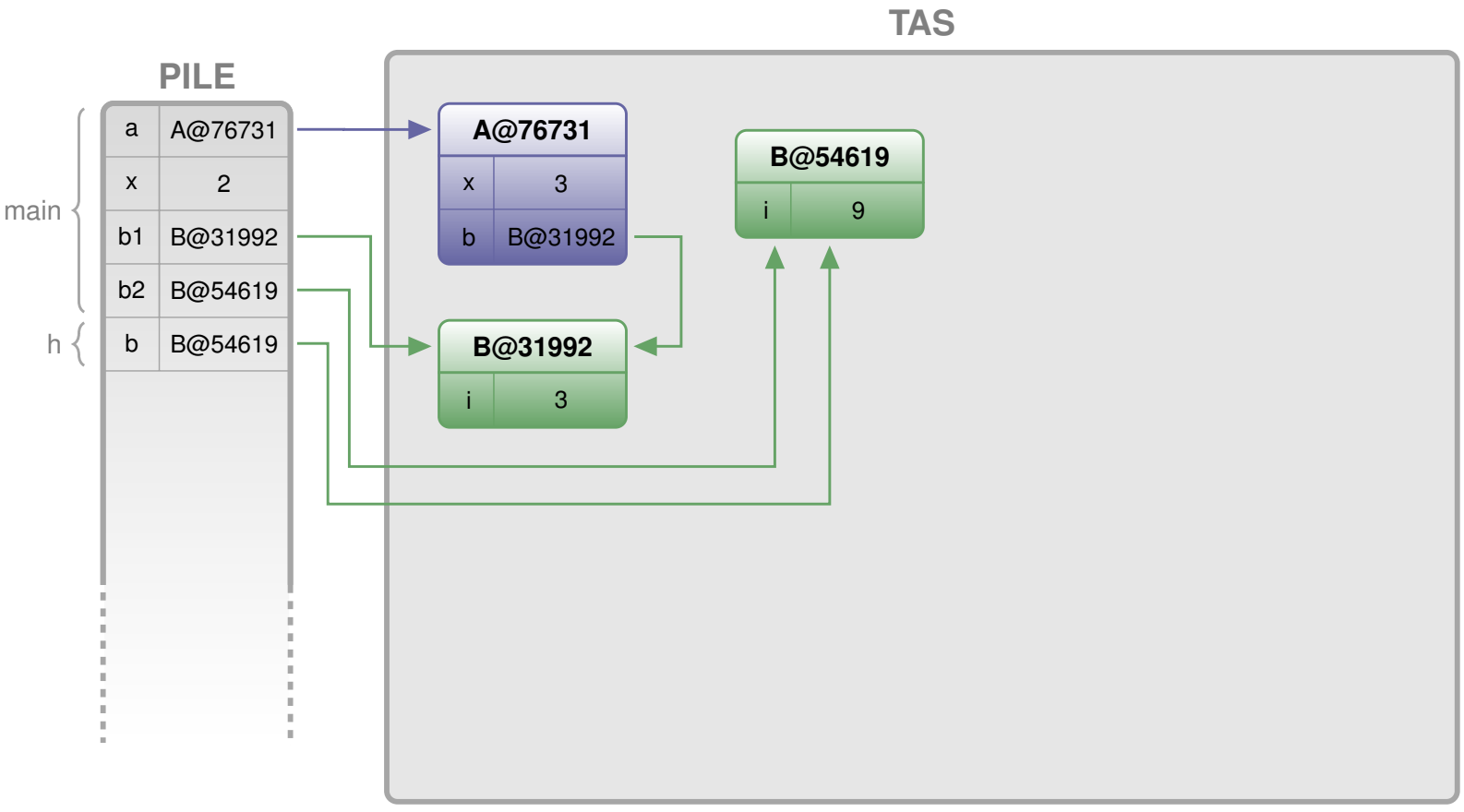
Appels successifs de méthodes



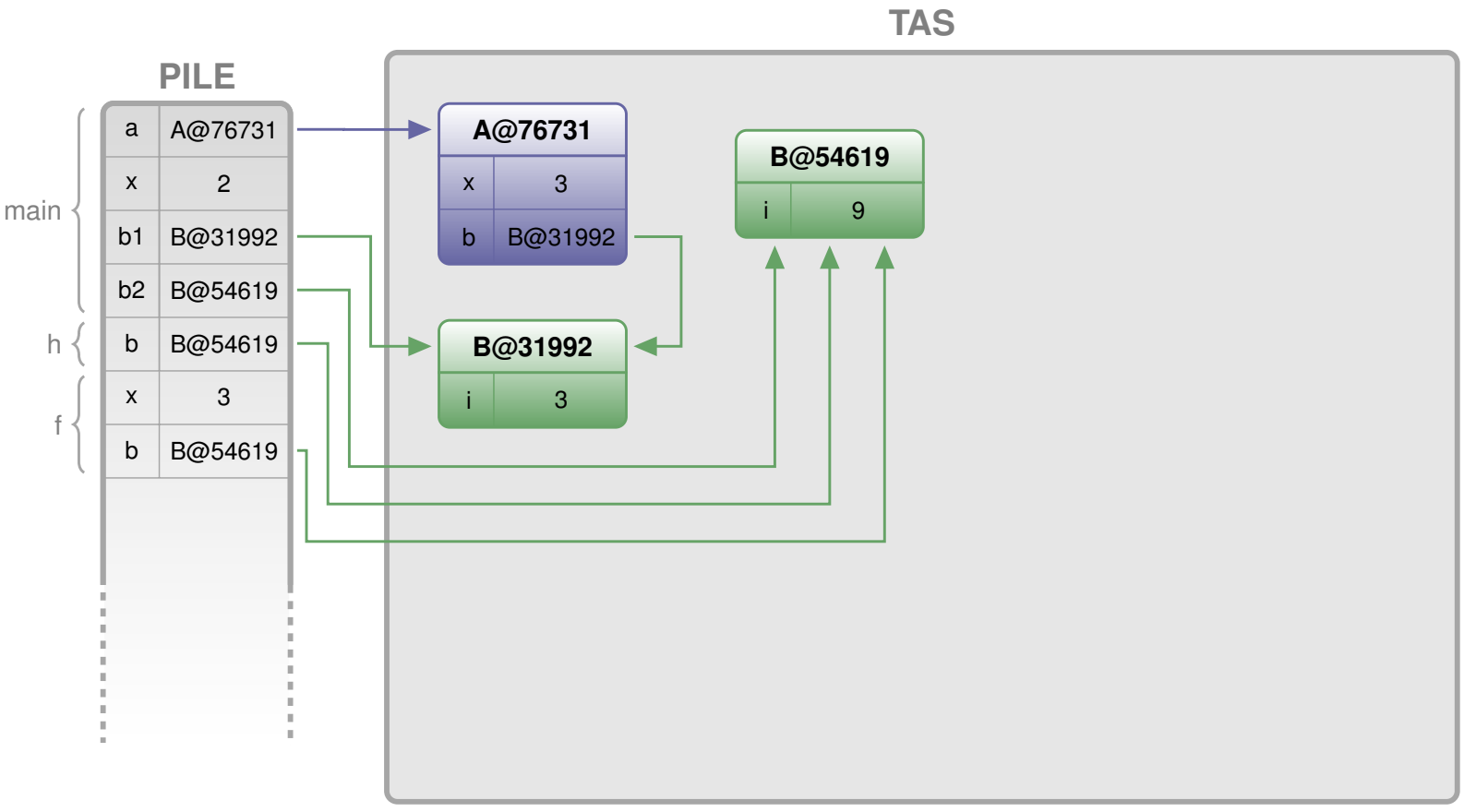
Appels successifs de méthodes



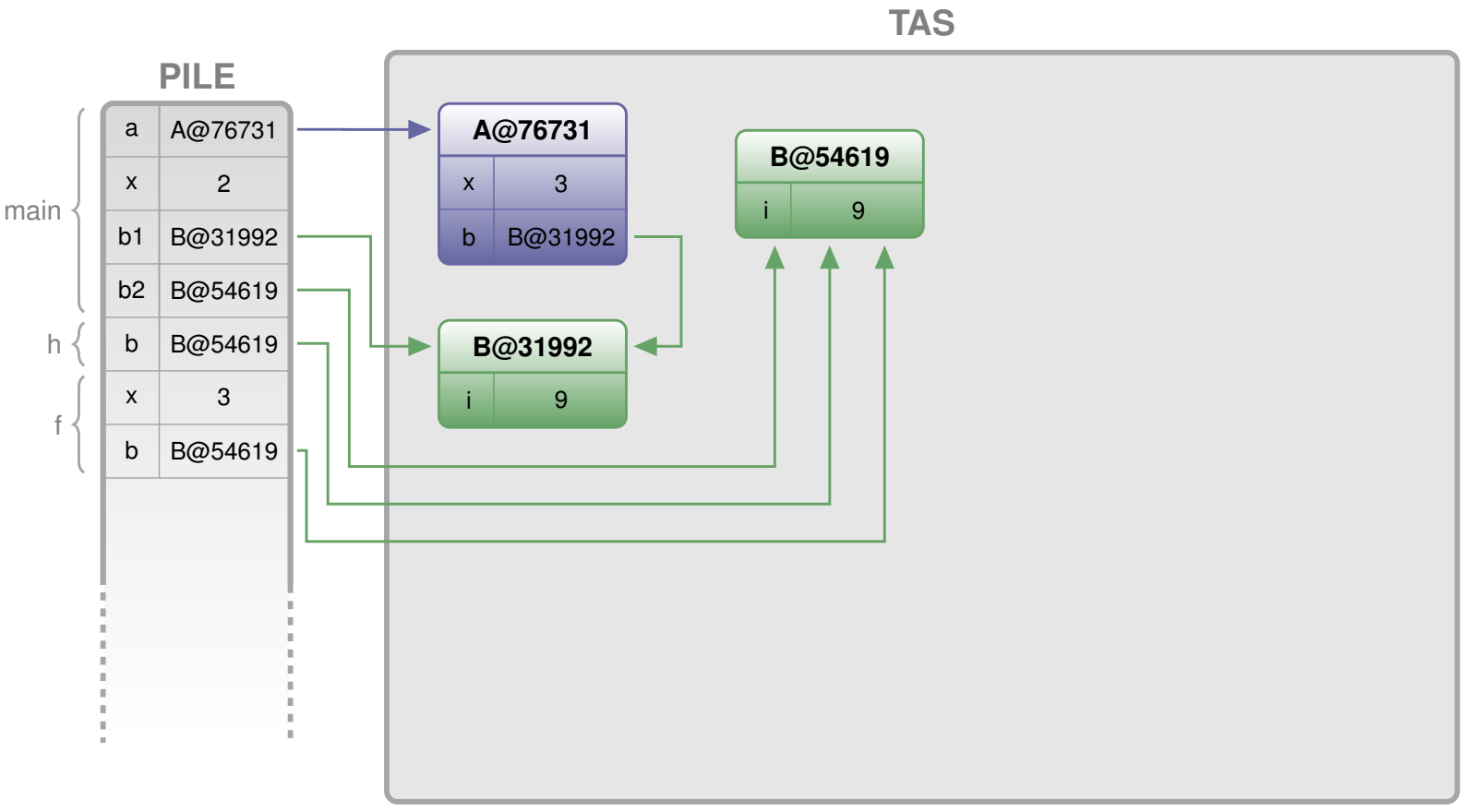
Appels successifs de méthodes



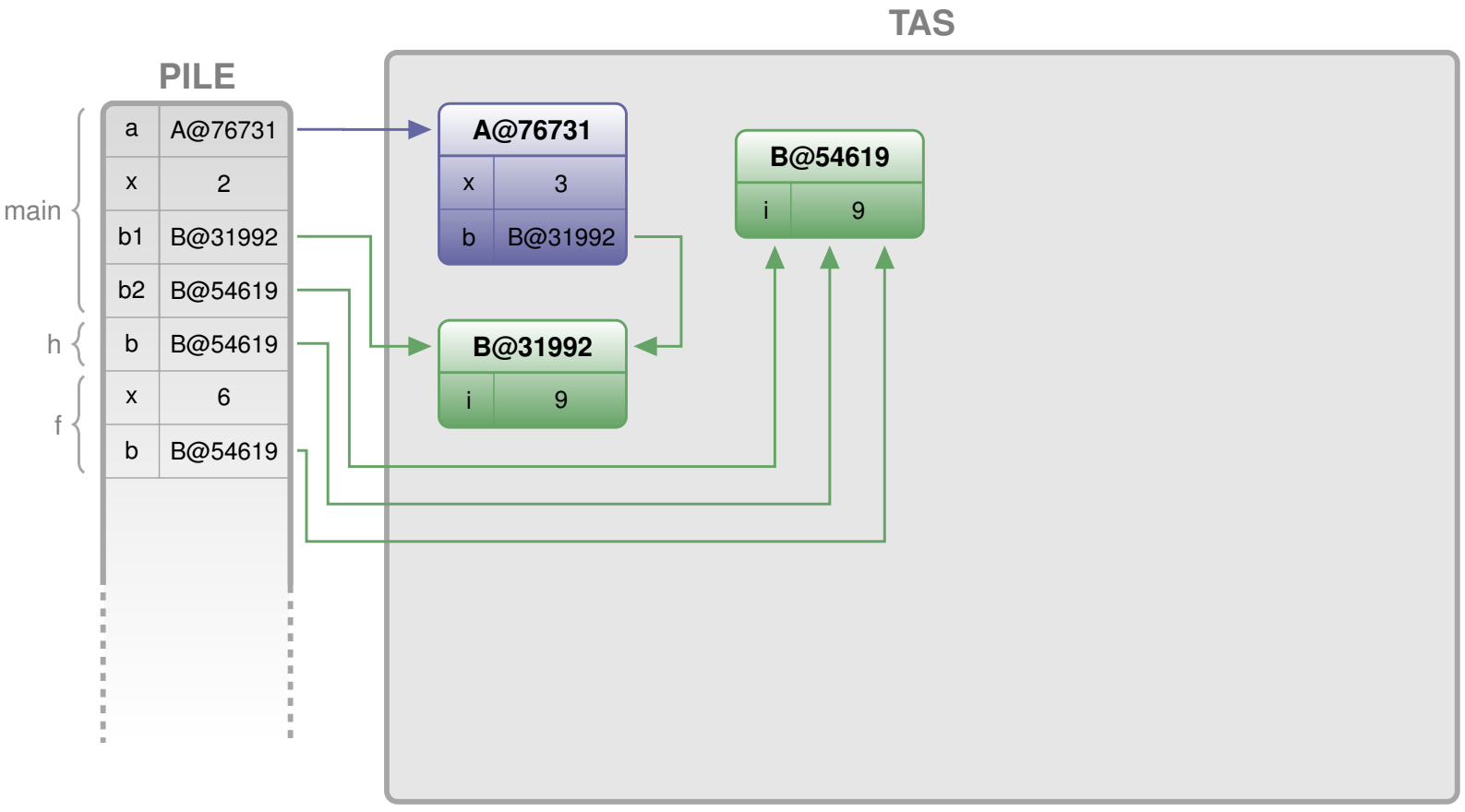
Appels successifs de méthodes



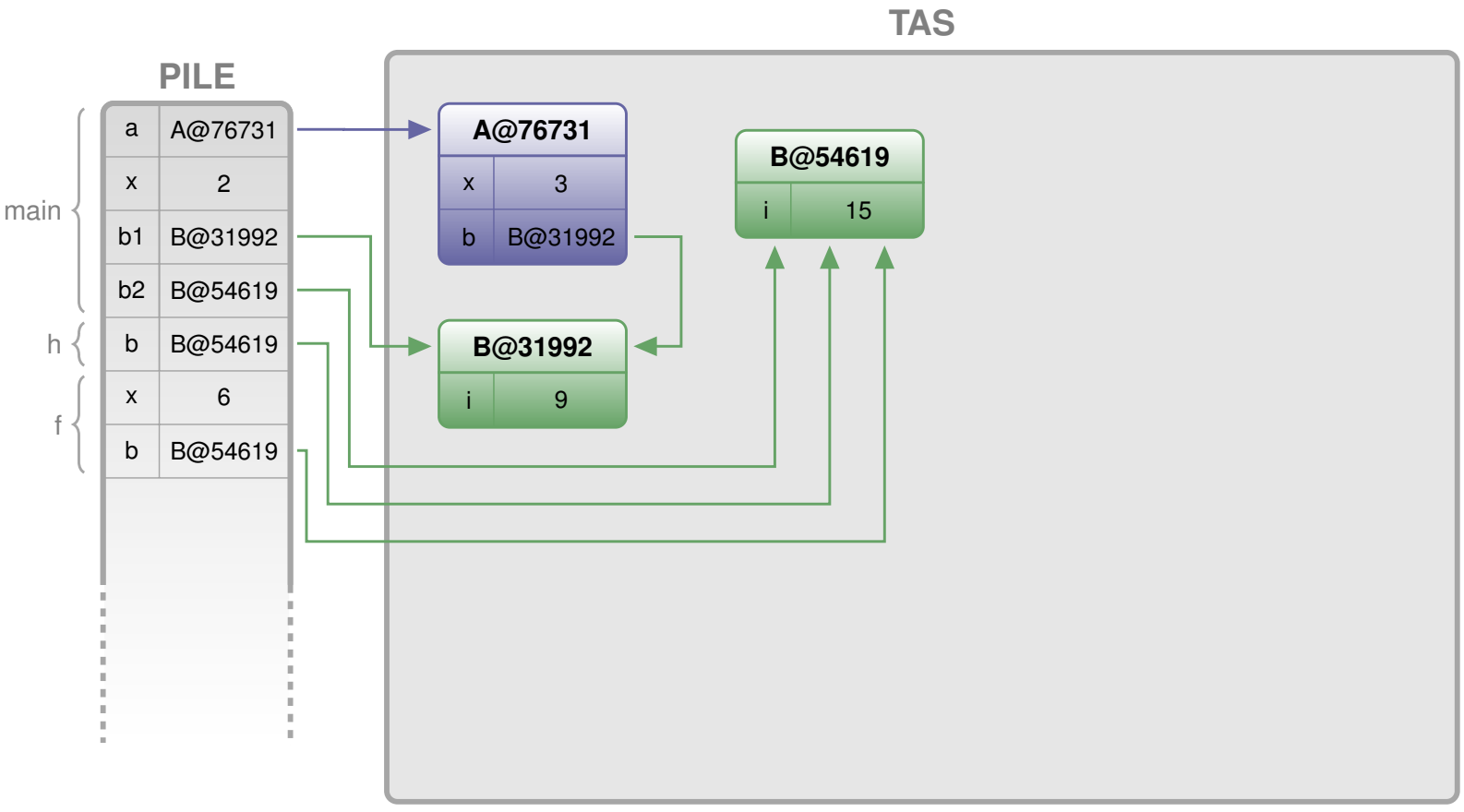
Appels successifs de méthodes



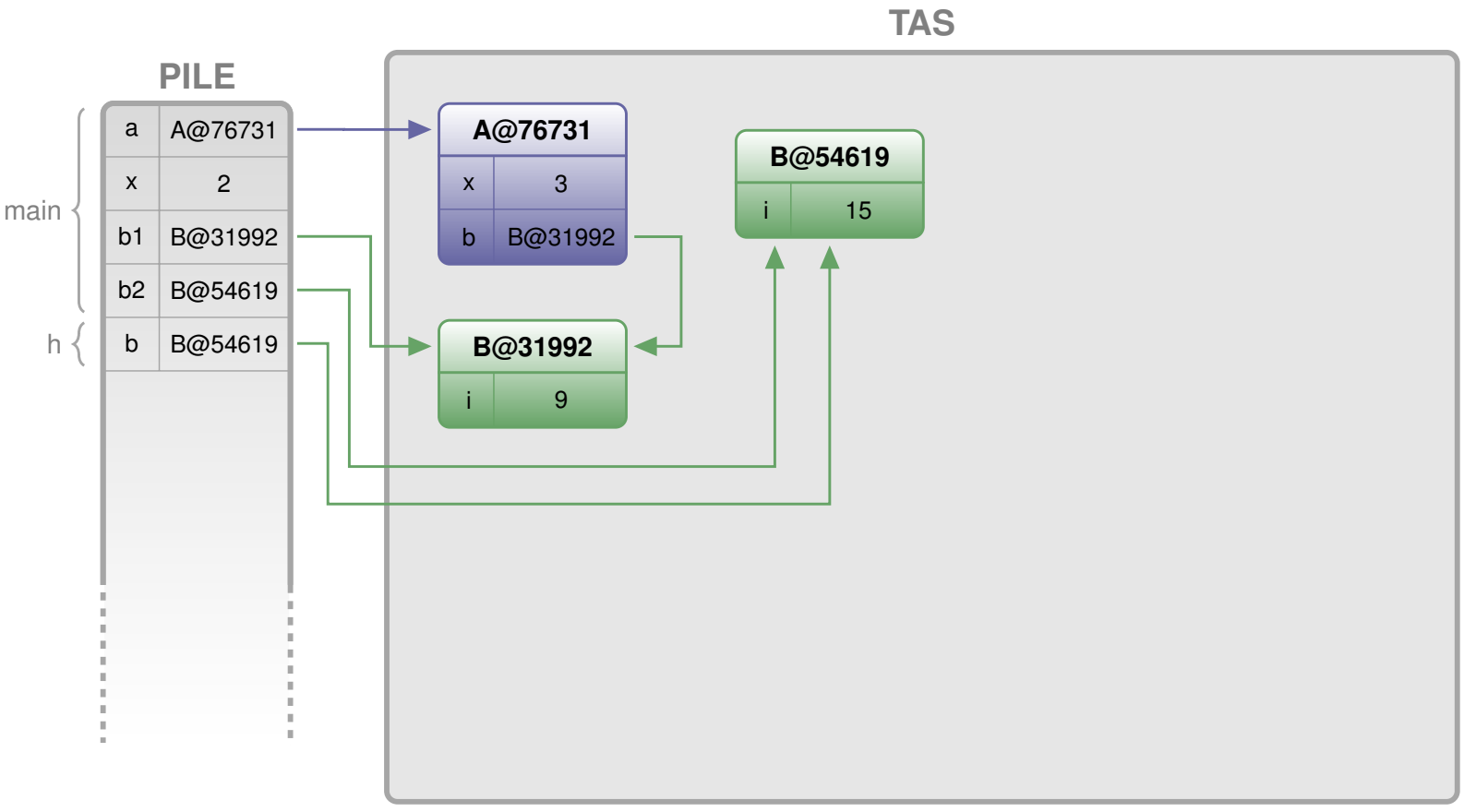
Appels successifs de méthodes



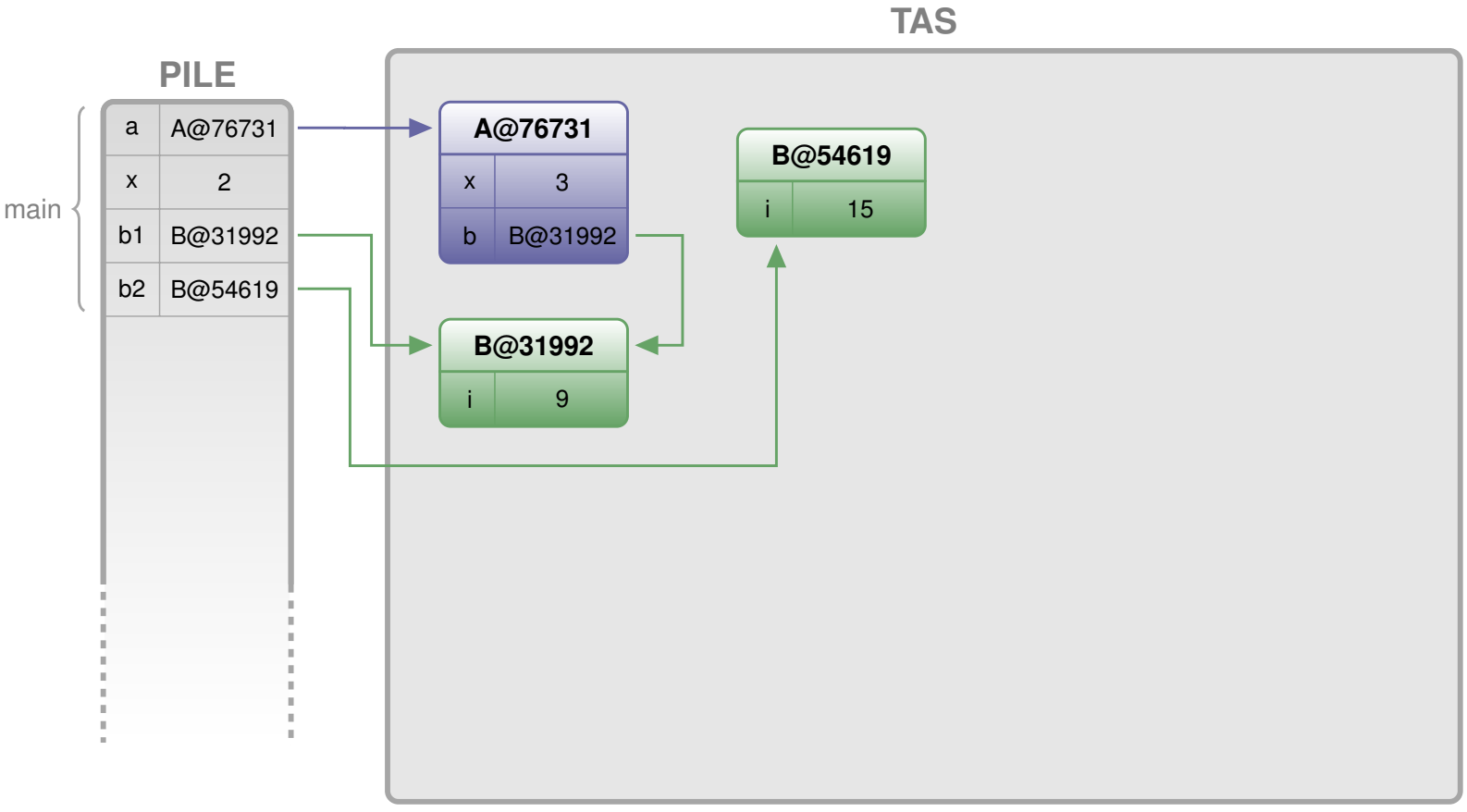
Appels successifs de méthodes



Appels successifs de méthodes



Appels successifs de méthodes



Outline

Appels récursifs d'une méthode

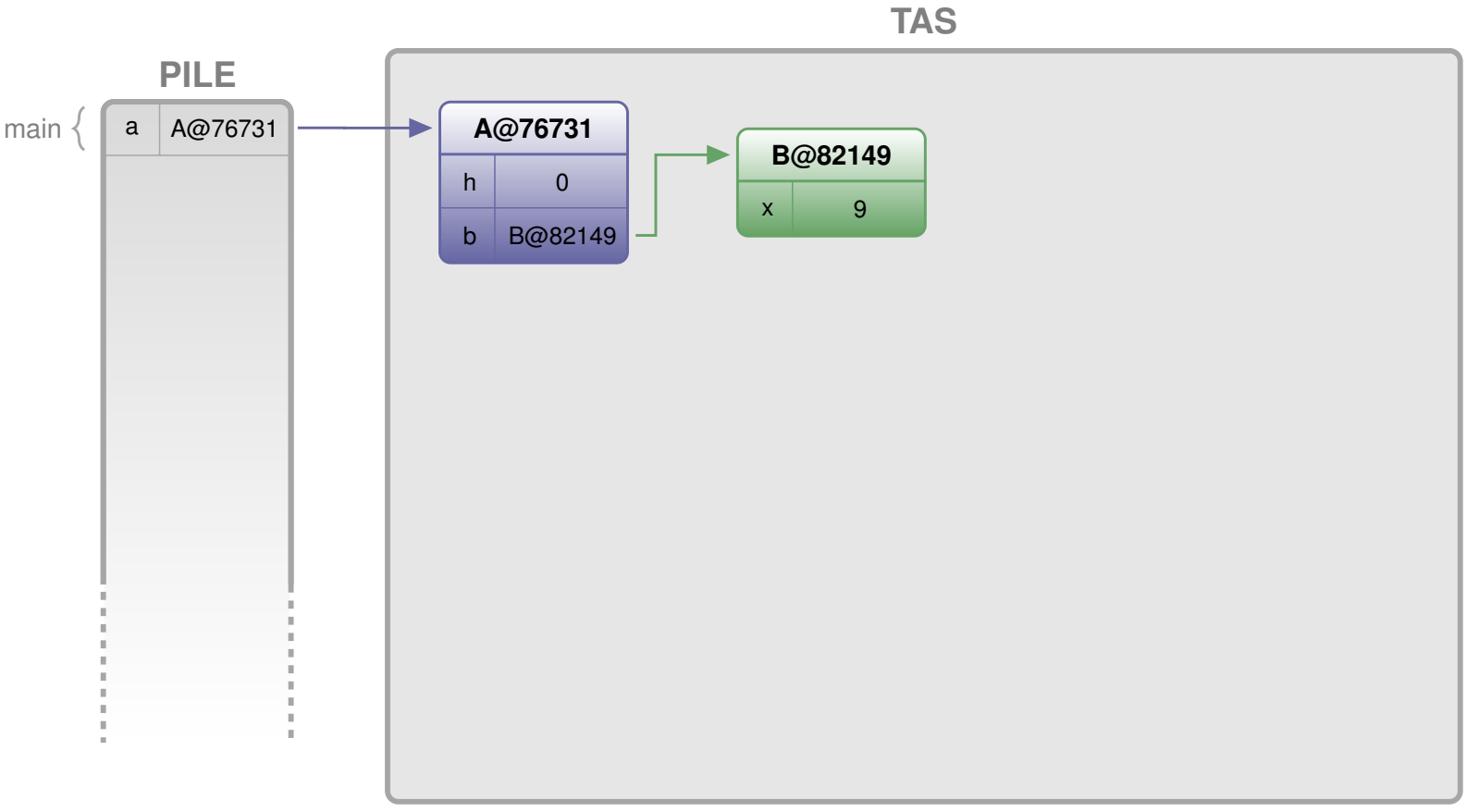
```
public class A {
  B b ;
  int h;
  public A (int x) {
    b = new B(x);
  }
  public boolean pyramide (int max) {
    f(this.b, 1) ;
    return p .
  }
  private void f (B b, int x) {
    B tmp = new B(b.x-x) ;
    if (tmp.x >= x+1) {
      B.f(tmp,x+1);
    } else {
      this.h = x ;
    }
  }
}
```

```
public class B {
  public int x;
  public B(int x) {
    this.x = x;
  }
}
```

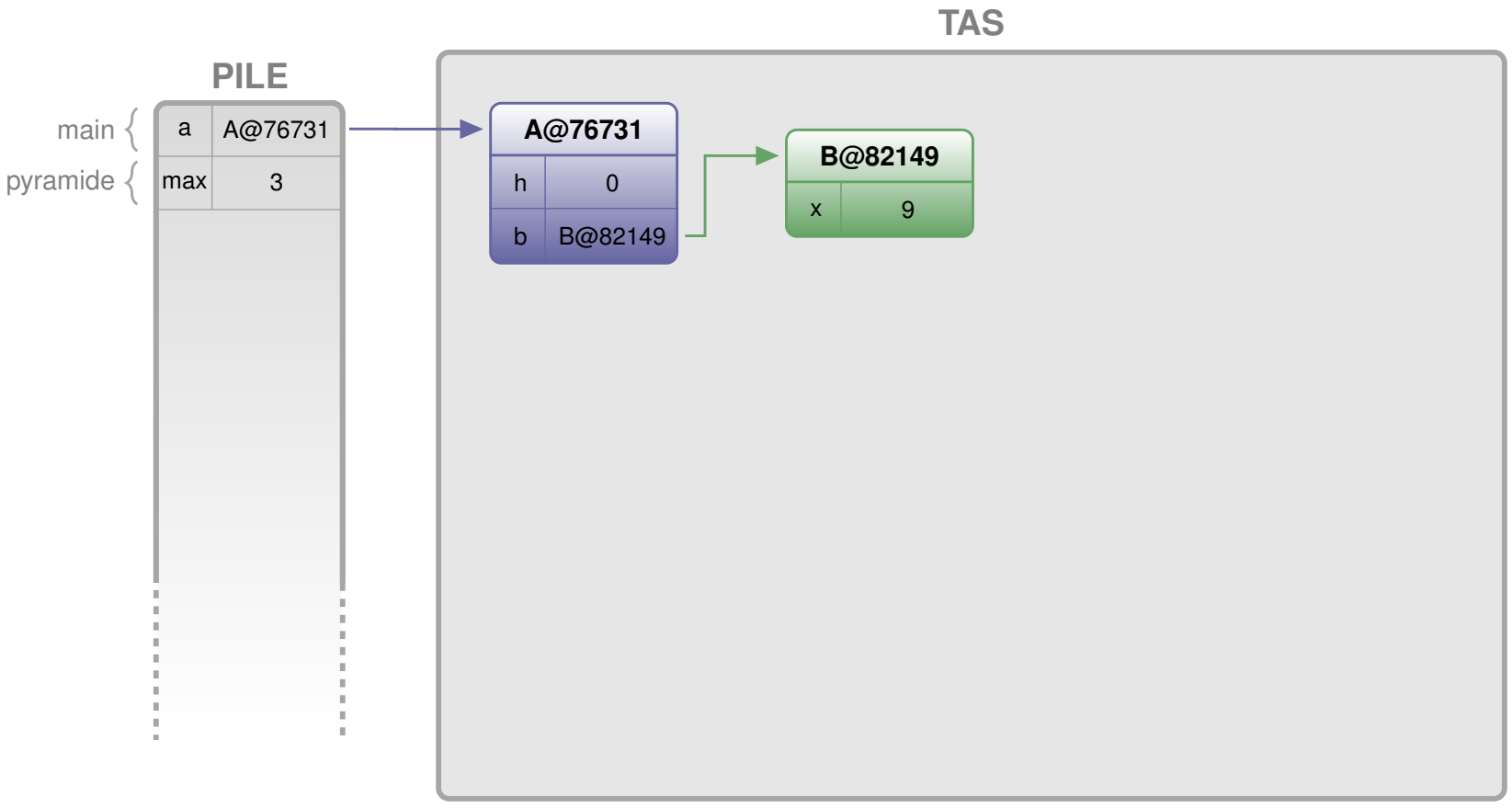
```
public class Main {
  public static void main(String[] args) {
    A a = new A(9);
    a.pyramide(3);
  }
}
```

Appels récursifs d'une méthode

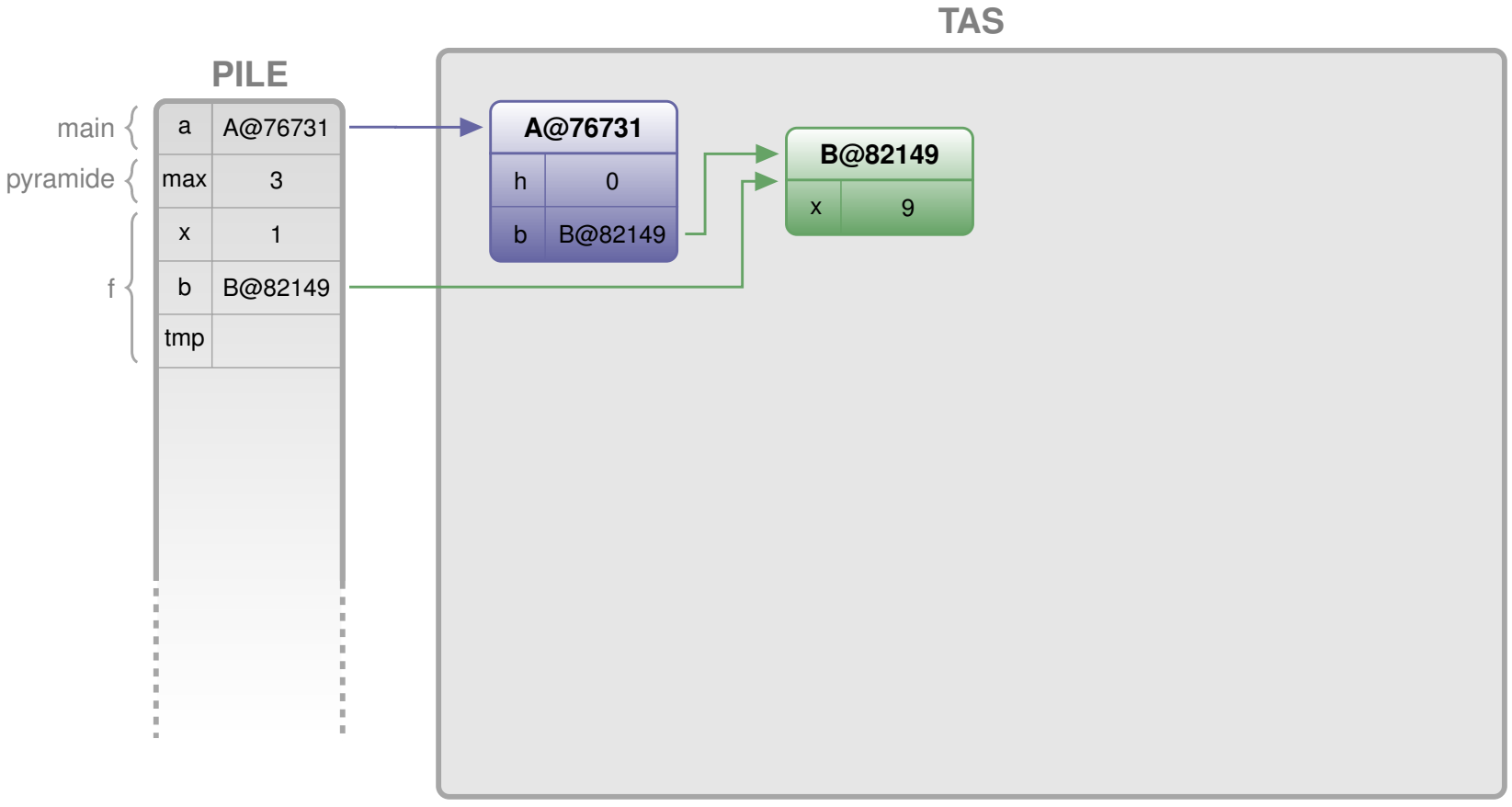
Appels récursifs d'une méthode



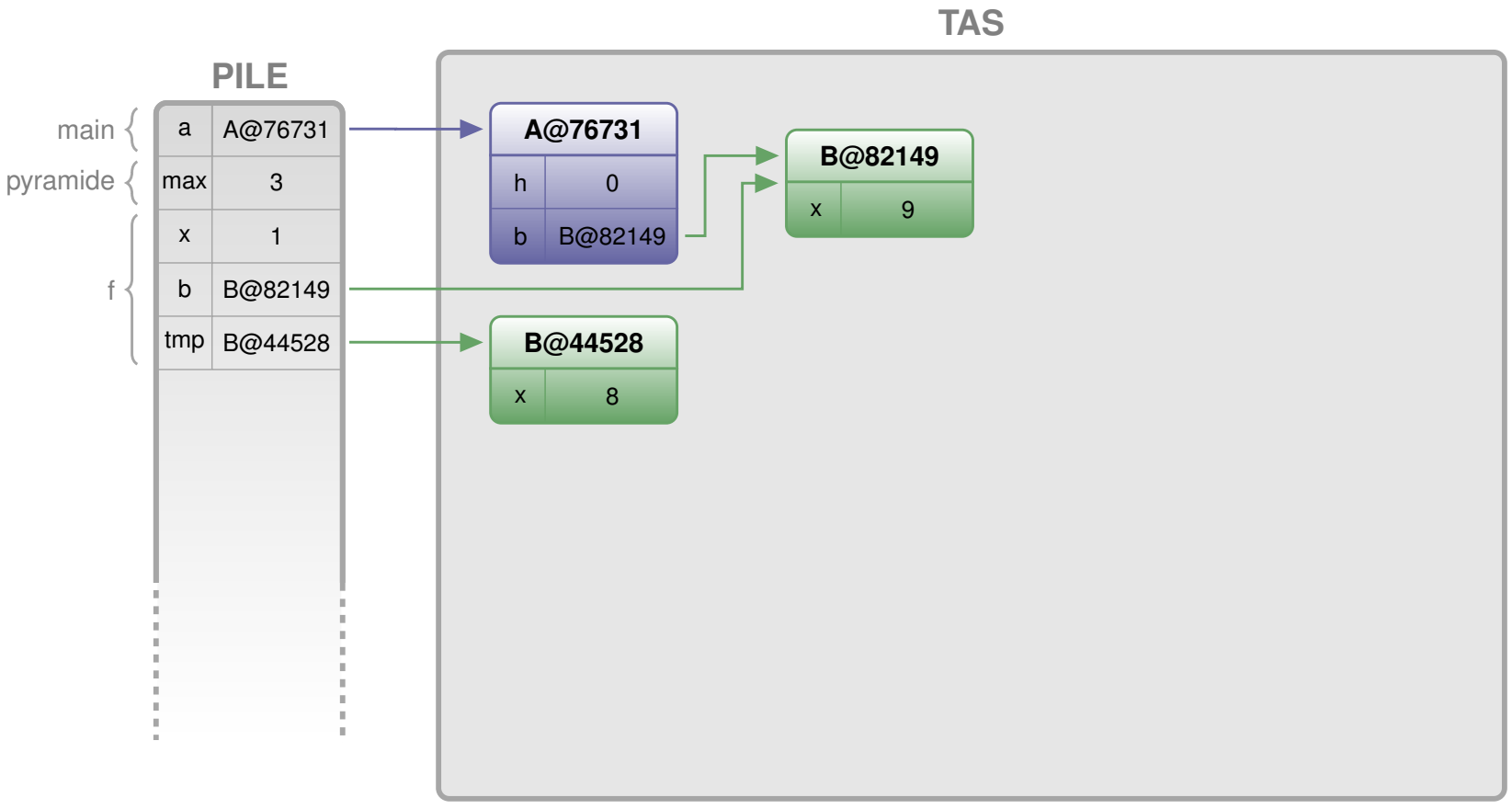
Appels récursifs d'une méthode



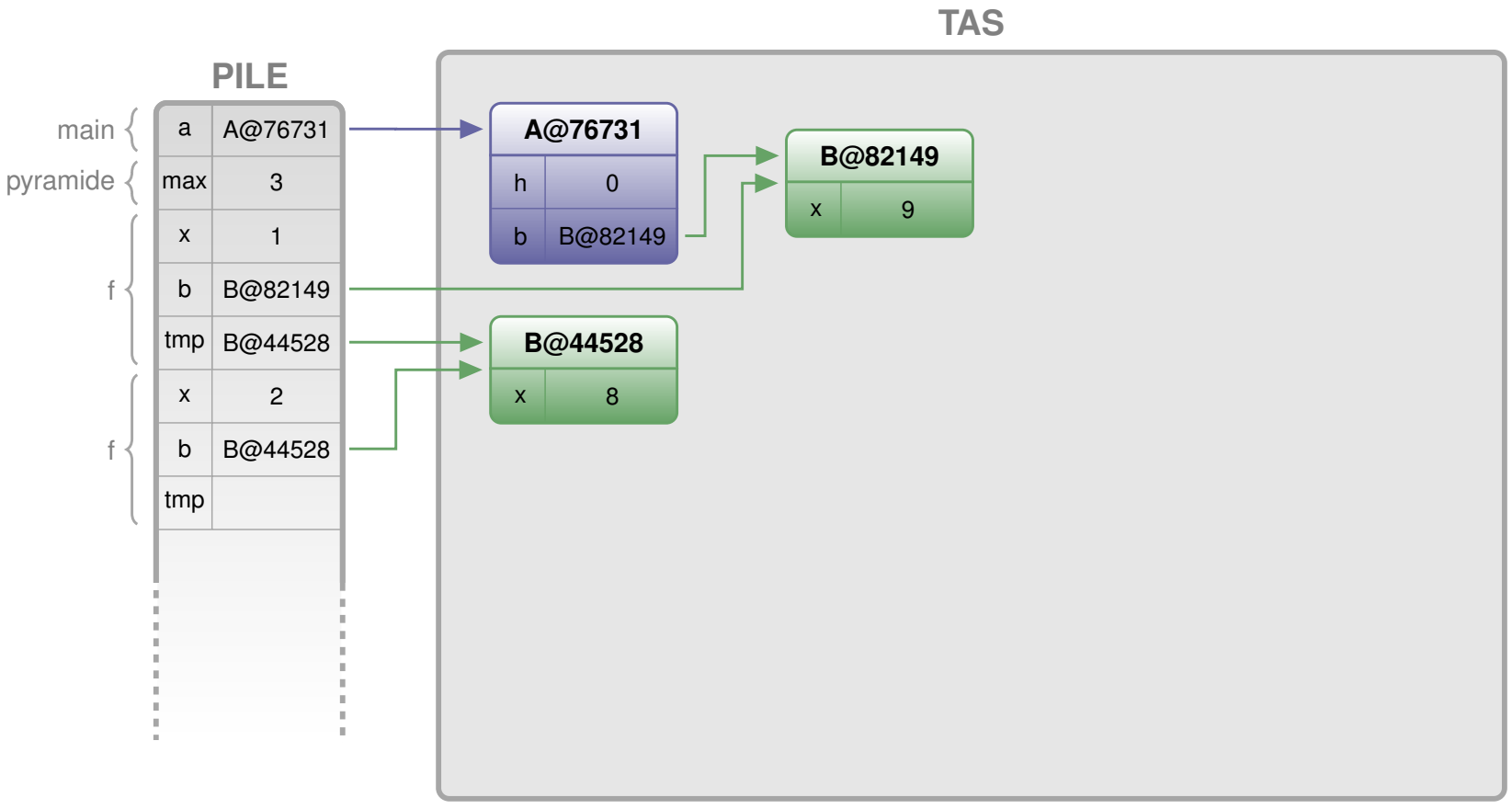
Appels récursifs d'une méthode



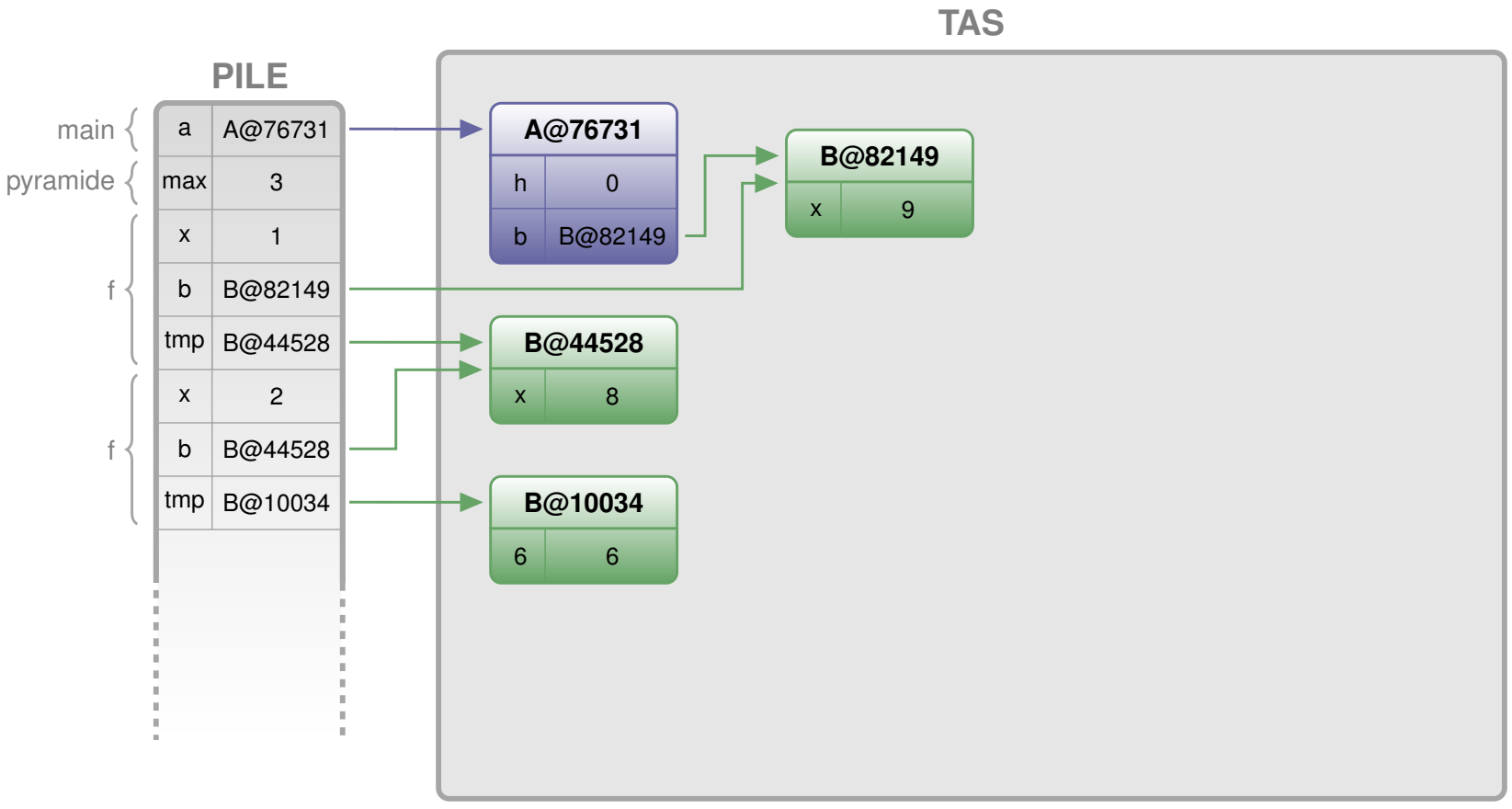
Appels récursifs d'une méthode



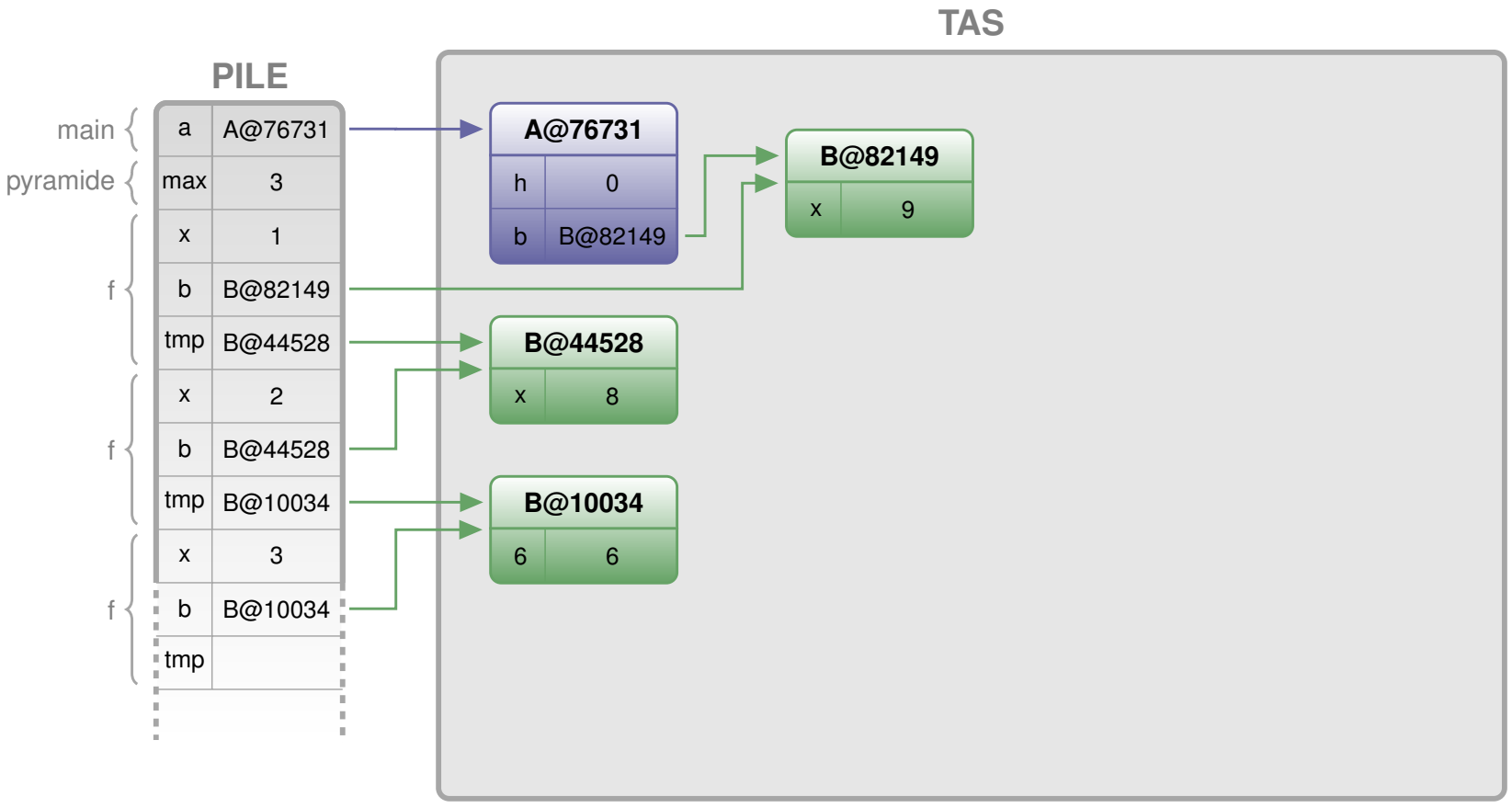
Appels récursifs d'une méthode



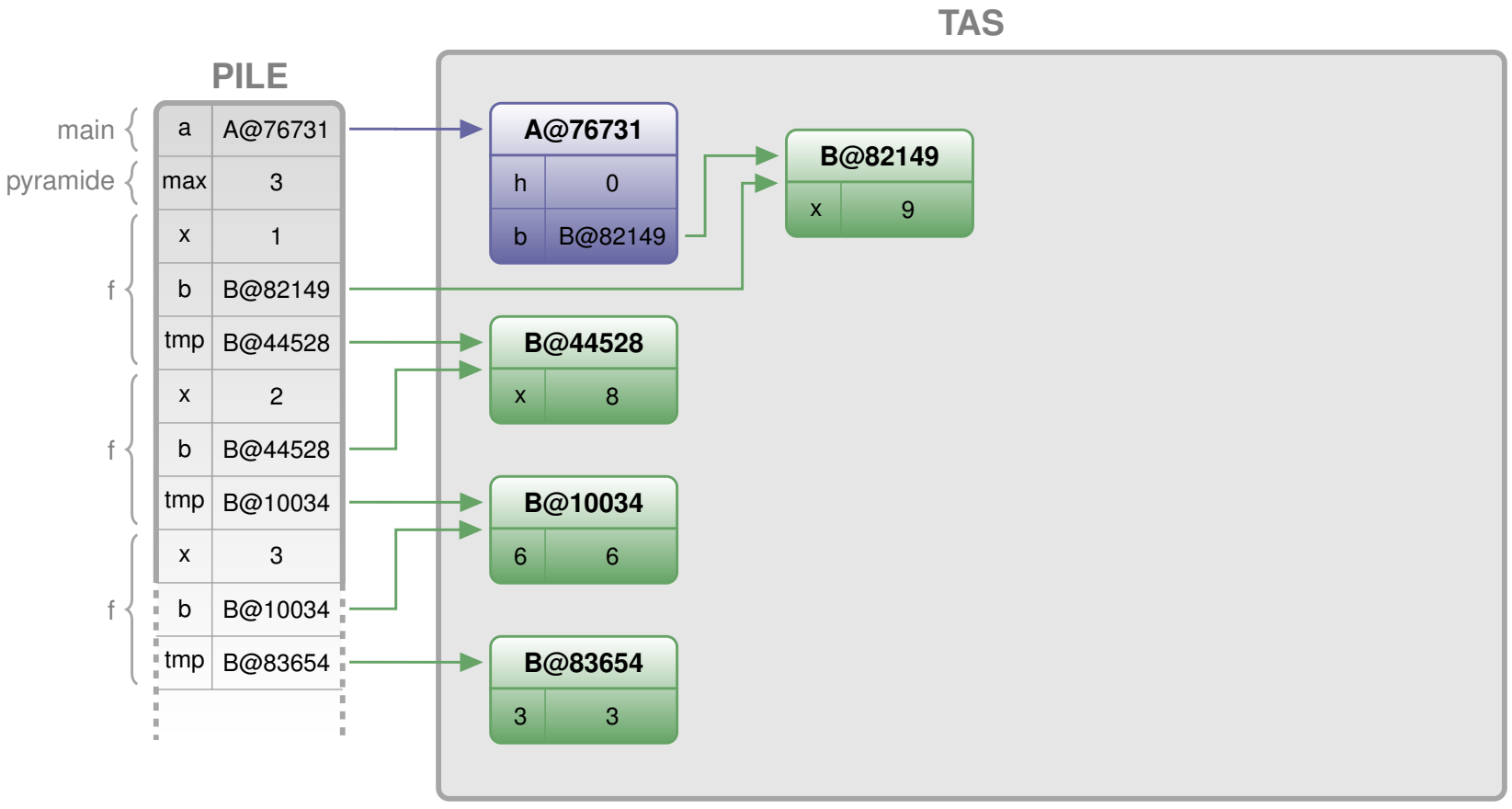
Appels récursifs d'une méthode



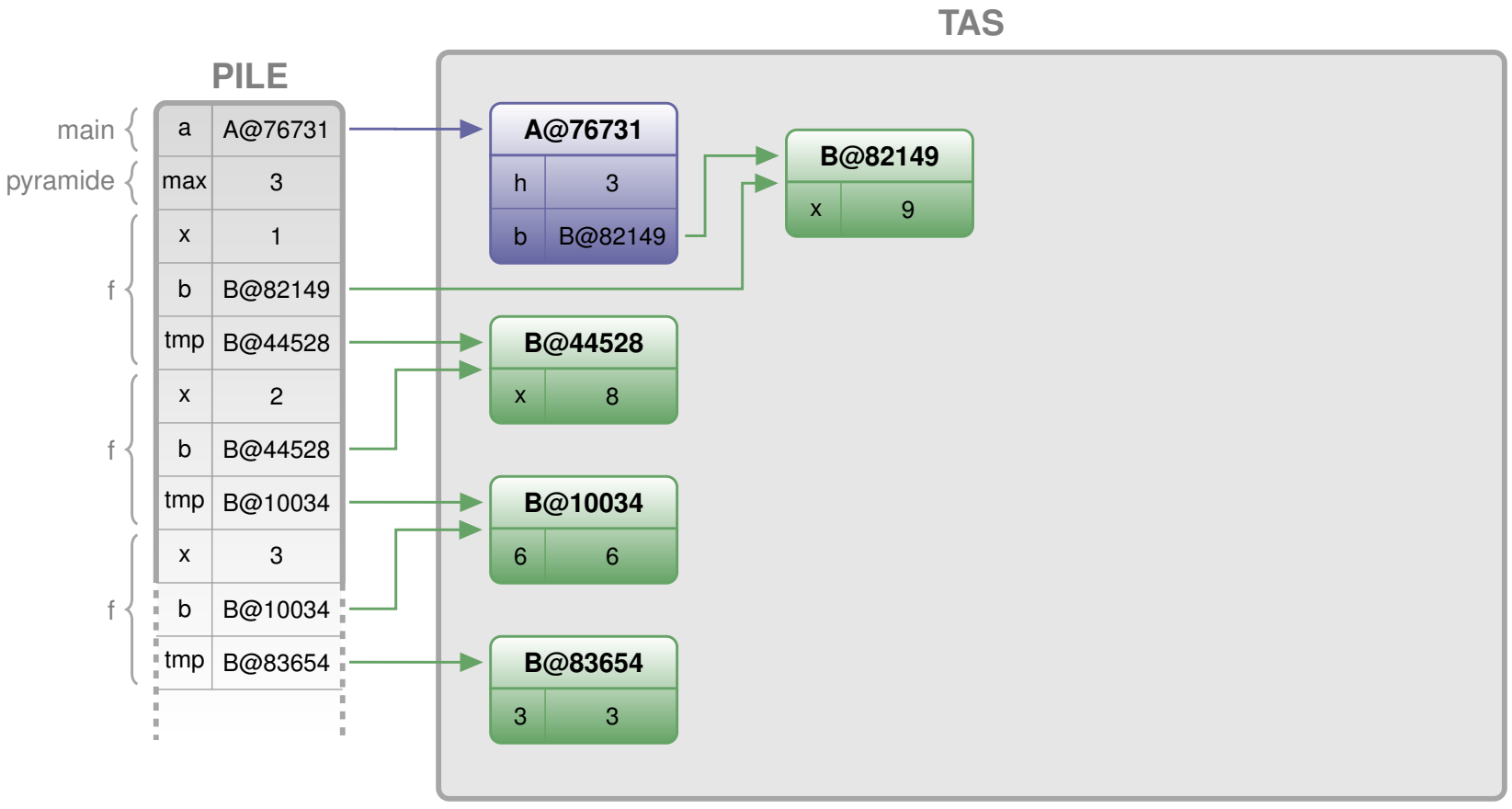
Appels récursifs d'une méthode



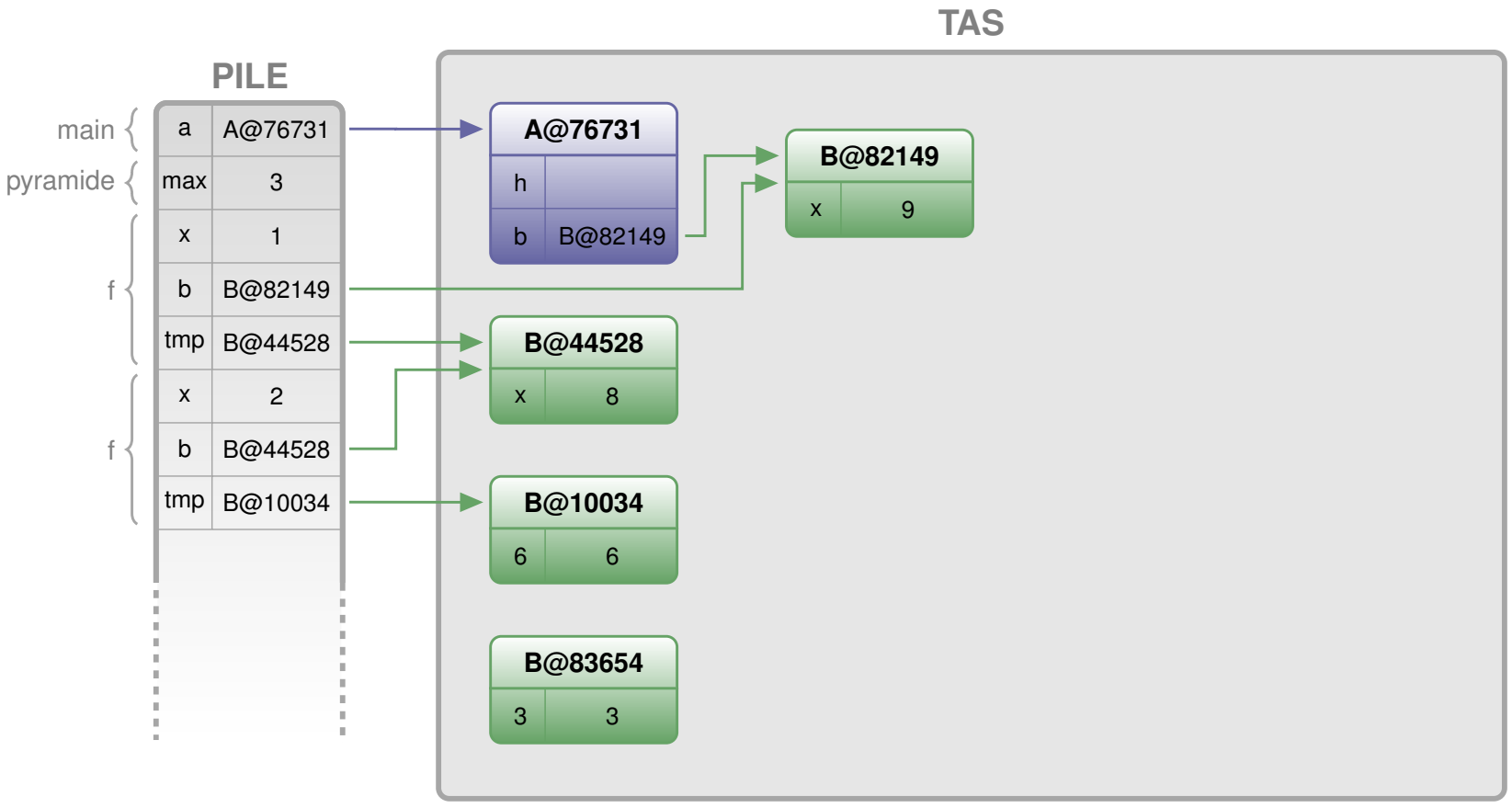
Appels récursifs d'une méthode



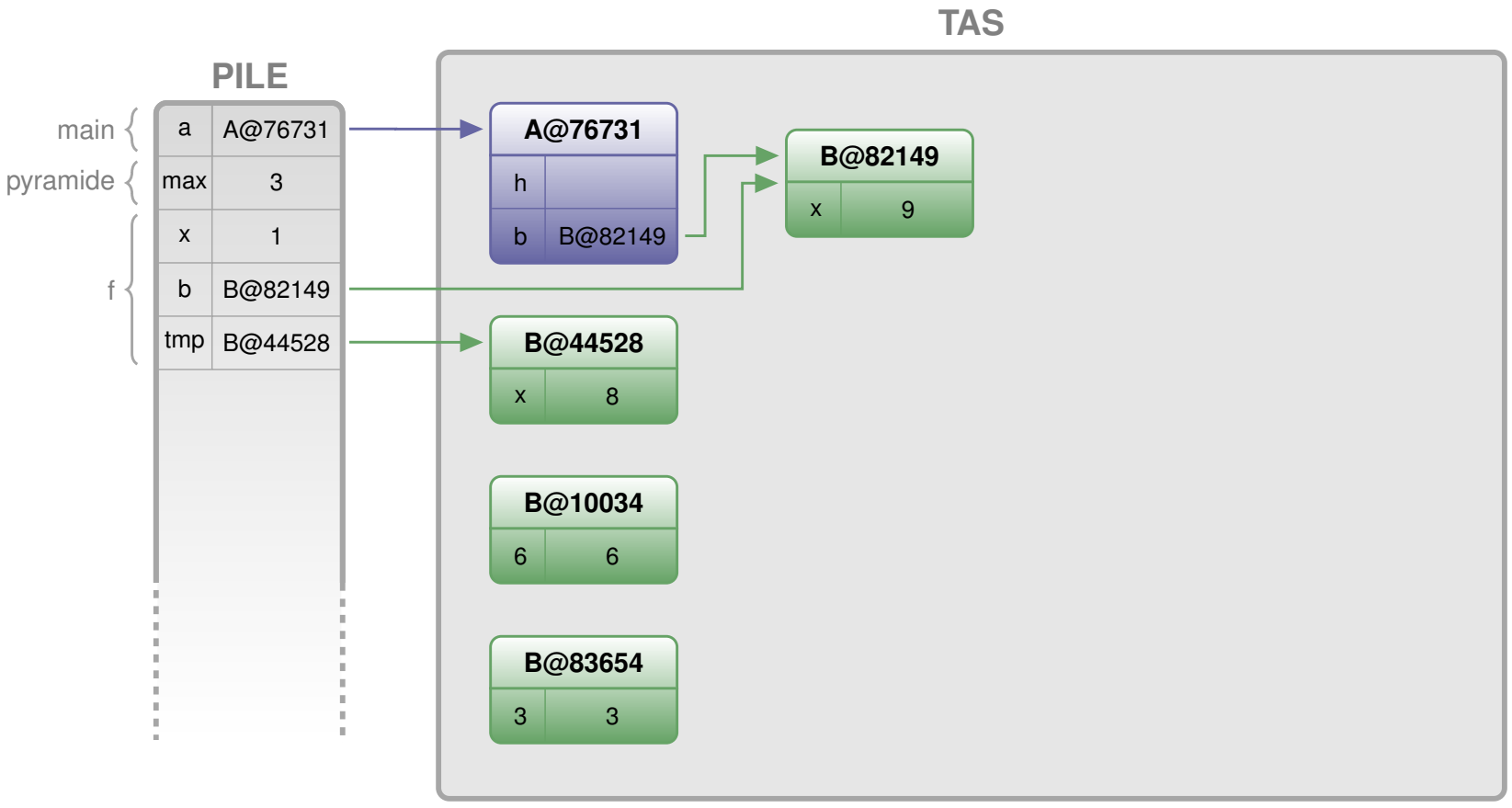
Appels récursifs d'une méthode



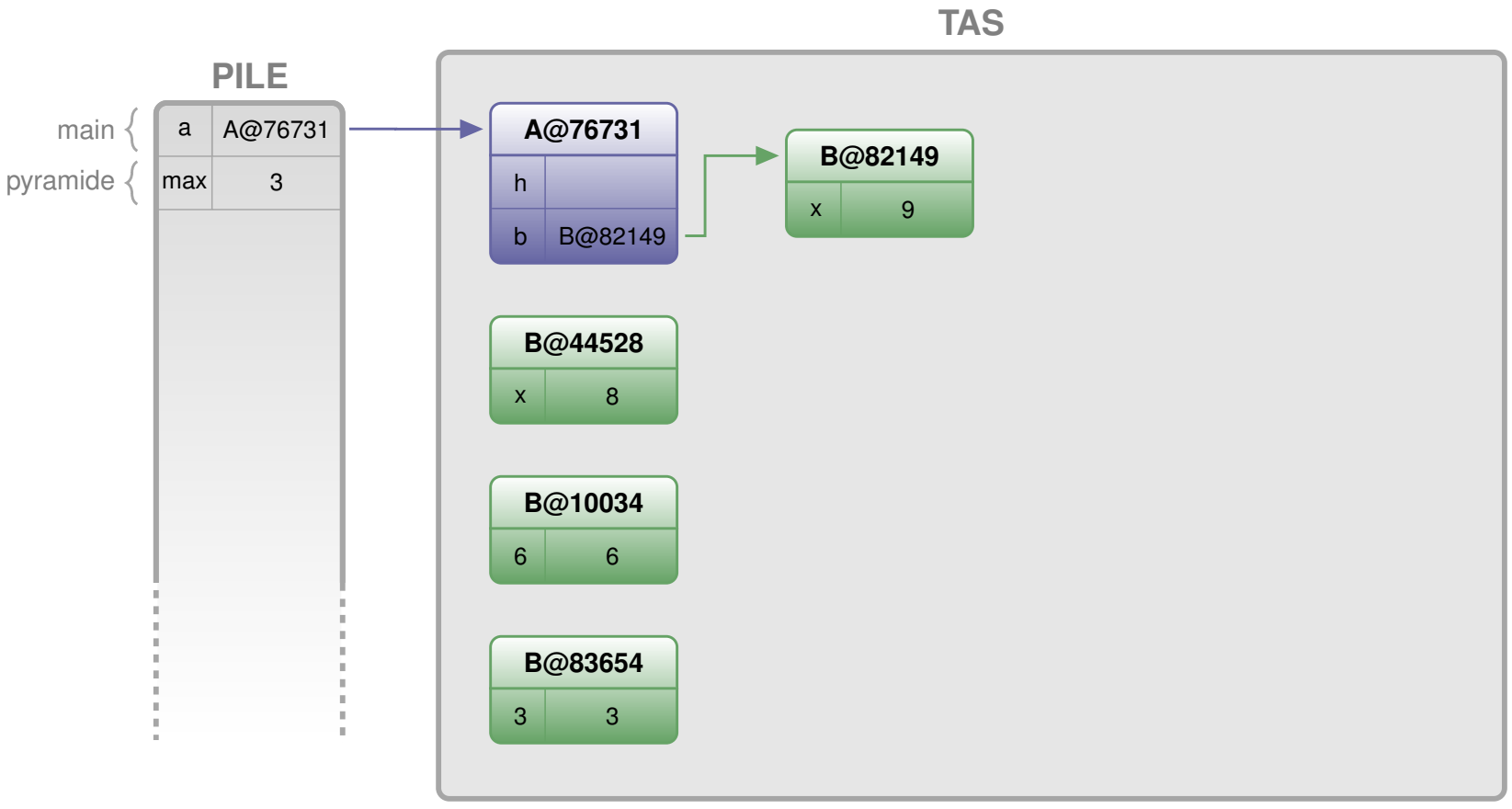
Appels récursifs d'une méthode



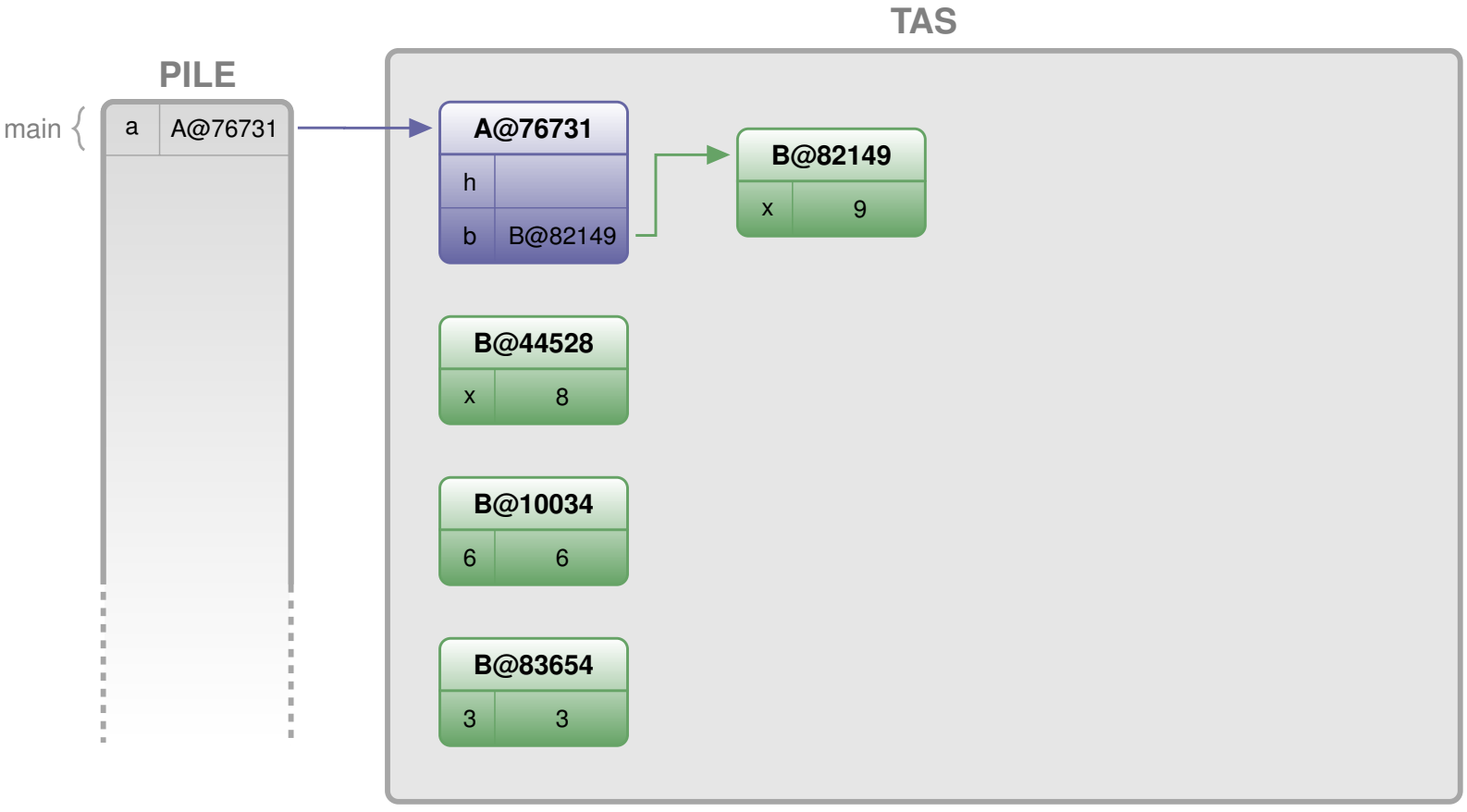
Appels récursifs d'une méthode



Appels récursifs d'une méthode



Appels récursifs d'une méthode



Outline

Les tableaux en Java

```
public class A {
  B b;
  B[] tab;
  public A() {
    b = new B();
    tab = new B[3];
  }
}
```

```
public class B {
  int x;
}
```

```
public class Main {
  public static void main(String[] args) {
    A a = new A();
    B[] t = a.tab ;
    t[1].x = 1 ;
    a.tab[0].x = 2 ;
    a.tab[2] = a.b ;
    a.b.x ++ ;
    for(int i=0 ; i<tab.length ; i++ ) {
      System.out.println("tab["+i+"]="+tab[i].x);
    }
  }
}
```

```
java Main
```

Les tableaux en Java

```java
public class A {
  B b;
  B[] tab;
  public A() {
    b = new B();
    tab = new B[3];
  }
}
```

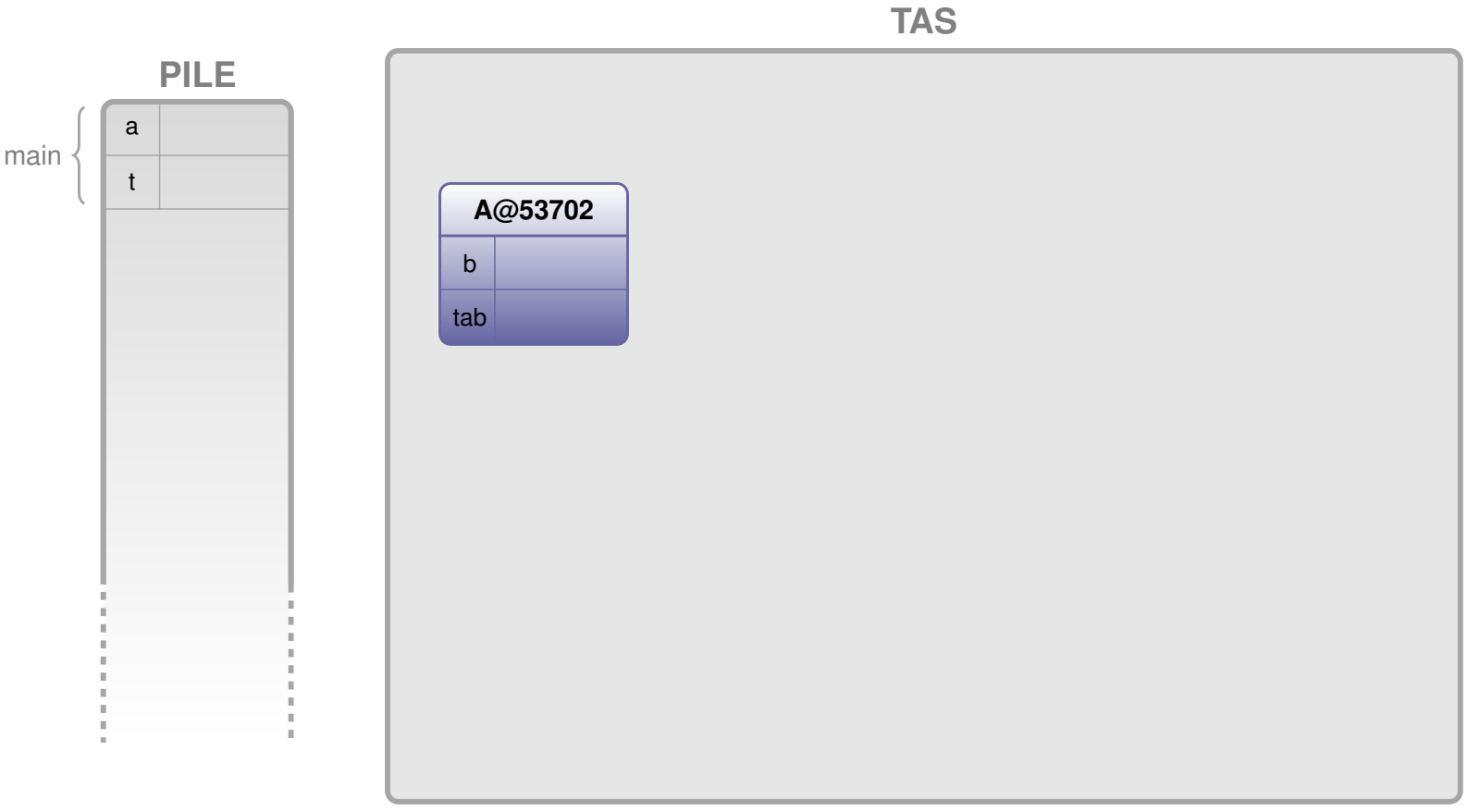
```java
public class B {
  int x;
}
```

```java
public class Main {
  public static void main(String[] args) {
    A a = new A();
    B[] t = a.tab ;
    t[1].x = 1 ;
    a.tab[0].x = 2 ;
    a.tab[2] = a.b ;
    a.b.x ++ ;
    for(int i=0 ; i<tab.length ; i++ ) {
      System.out.println("tab["+i+"]="+tab[i].x);
    }
  }
}
```

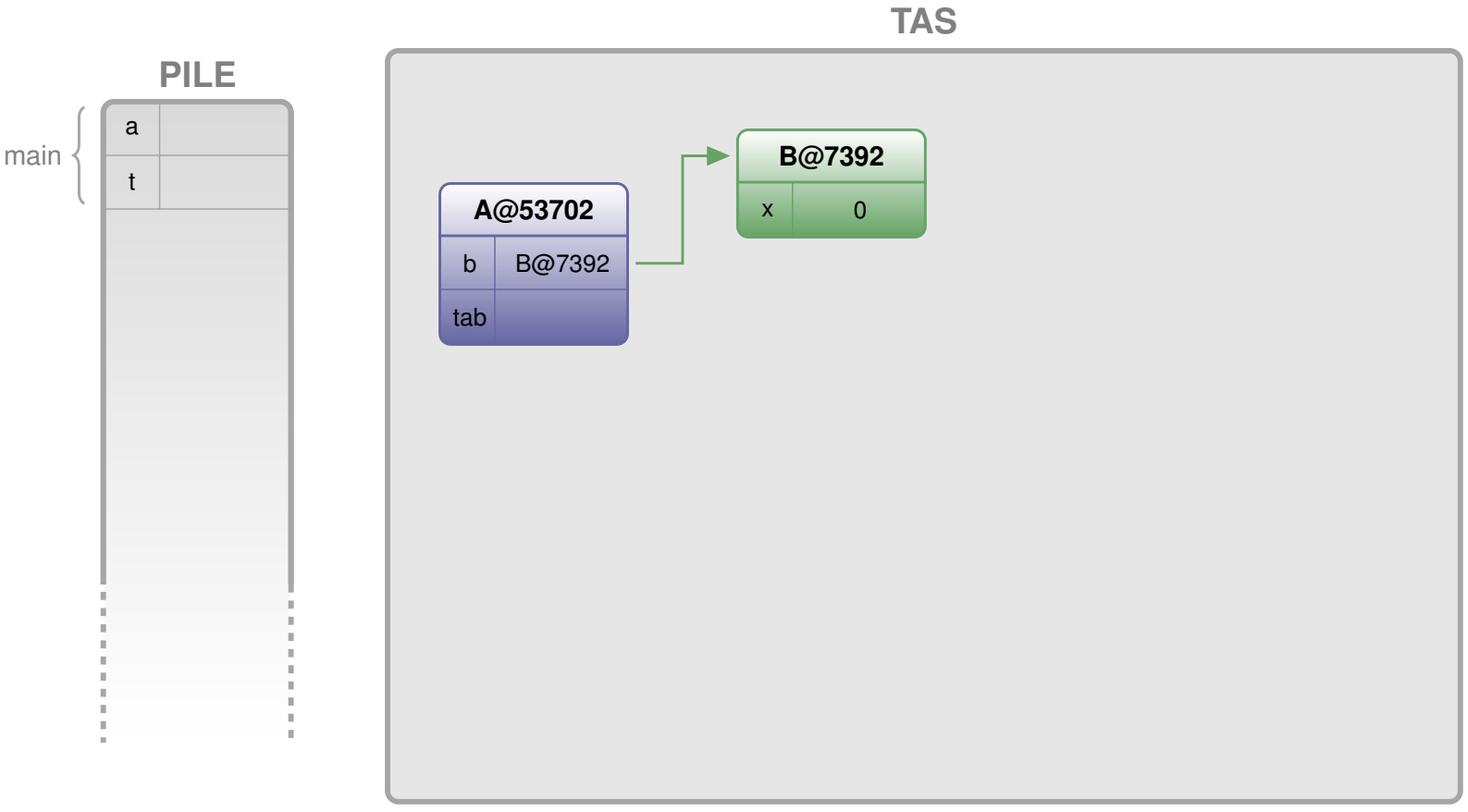
```
java Main
     Exception in thread "main" java.lang.NullPointerException
            at tableaux.Main.main(Main.java:5)
```

Les tableaux en Java

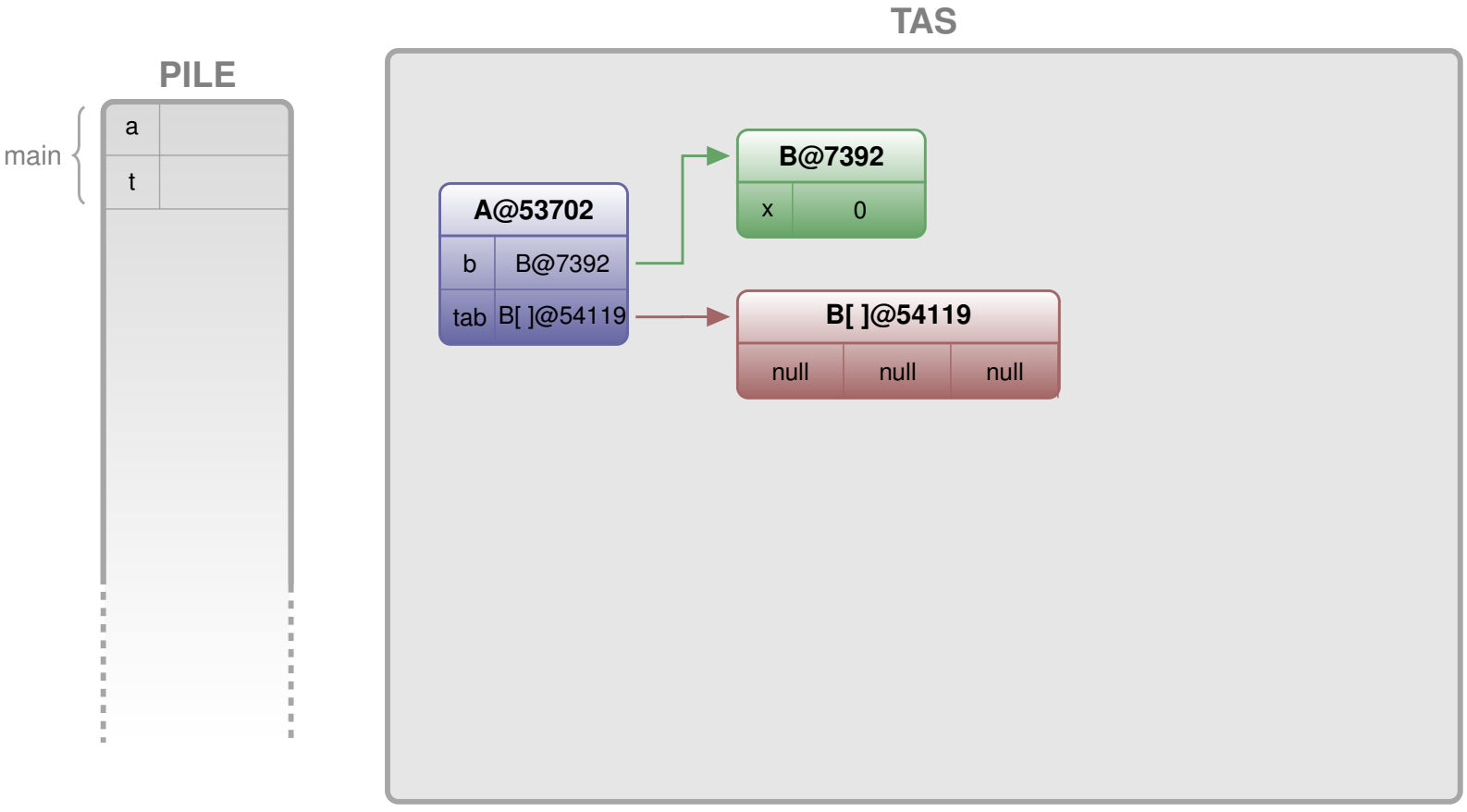
Les tableaux en Java



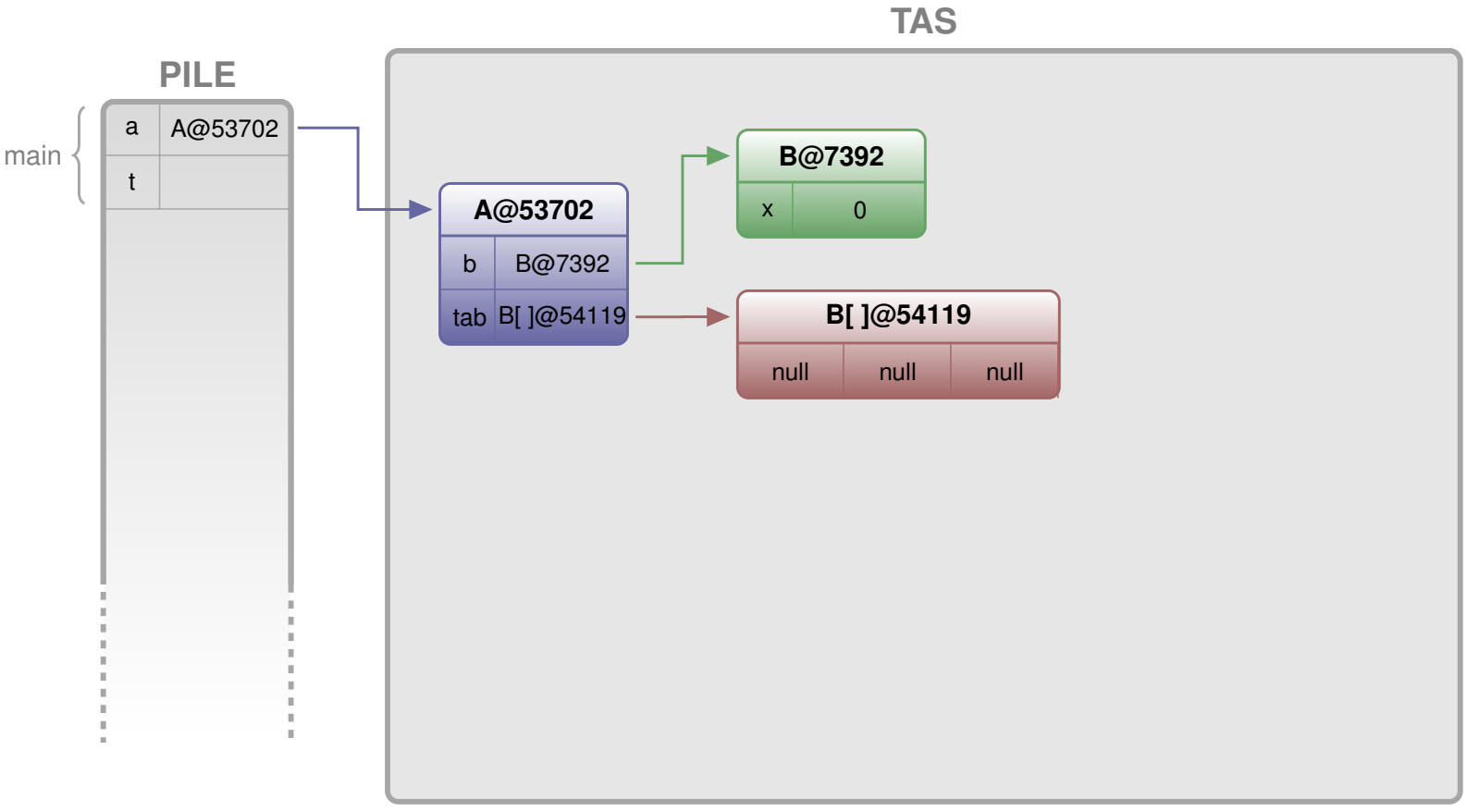
Les tableaux en Java



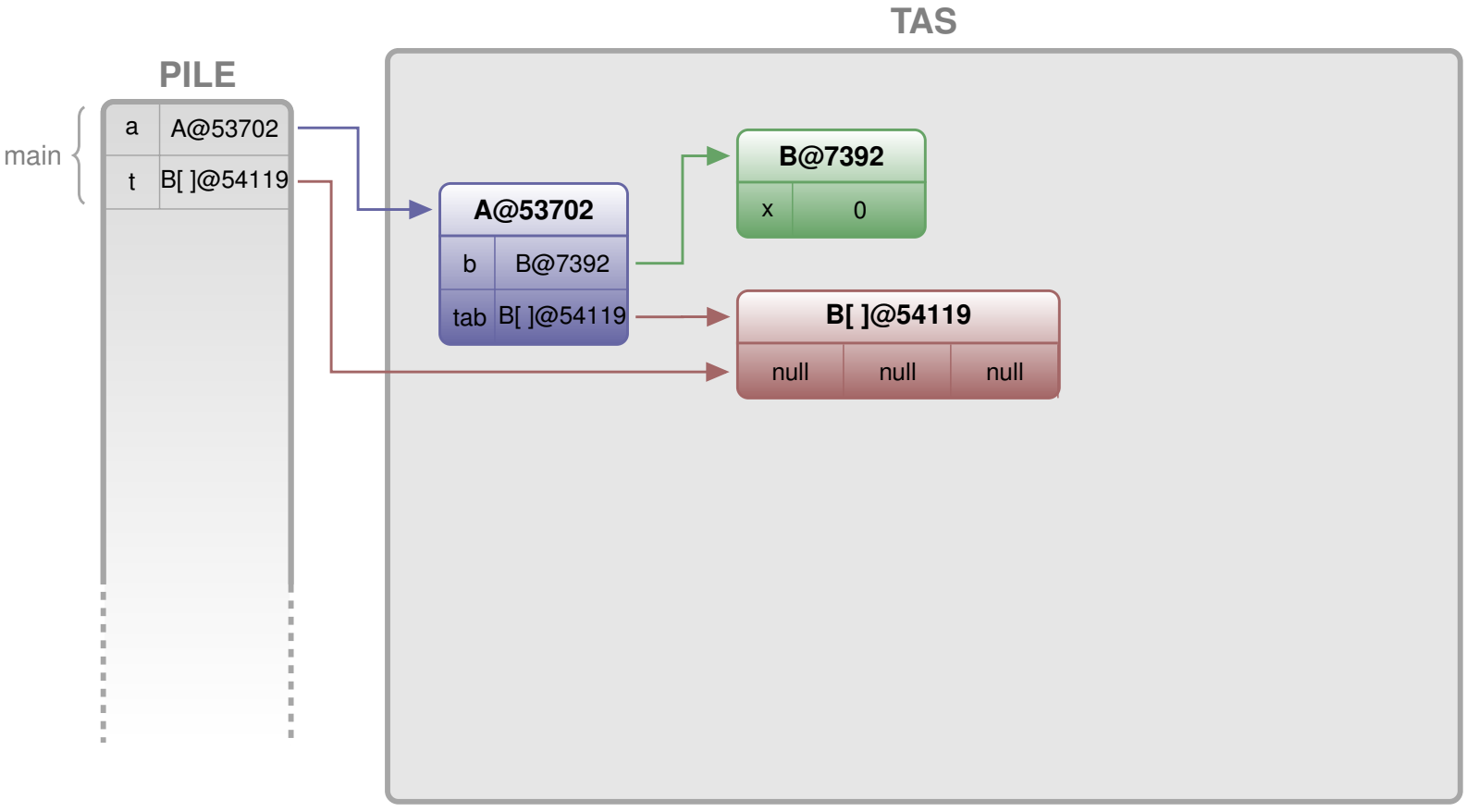
Les tableaux en Java



Les tableaux en Java



Les tableaux en Java



Les tableaux en Java

```
public class A {
  B b;
  B[] tab;
  public A() {
    b = new B();
    tab = new B[3];
    for(int i=0 ; i<tab.length ; i++ ) {
      tab[i] = new B();
    }
  }
}
```

```
public class B {
  int x;
}
```

```
public class Main {
  public static void main(String[] args) {
    A a = new A();
    B[] t = a.tab ;
    t[1].x = 1 ;
    a.tab[0].x = 2 ;
    a.tab[2] = a.b ;
    a.b.x ++ ;
    for(int i=0 ; i<tab.length ; i++ ) {
      System.out.println("tab["+i+"]="+tab[i].x);
    }
  }
}
```

```
java Main
```

Les tableaux en Java

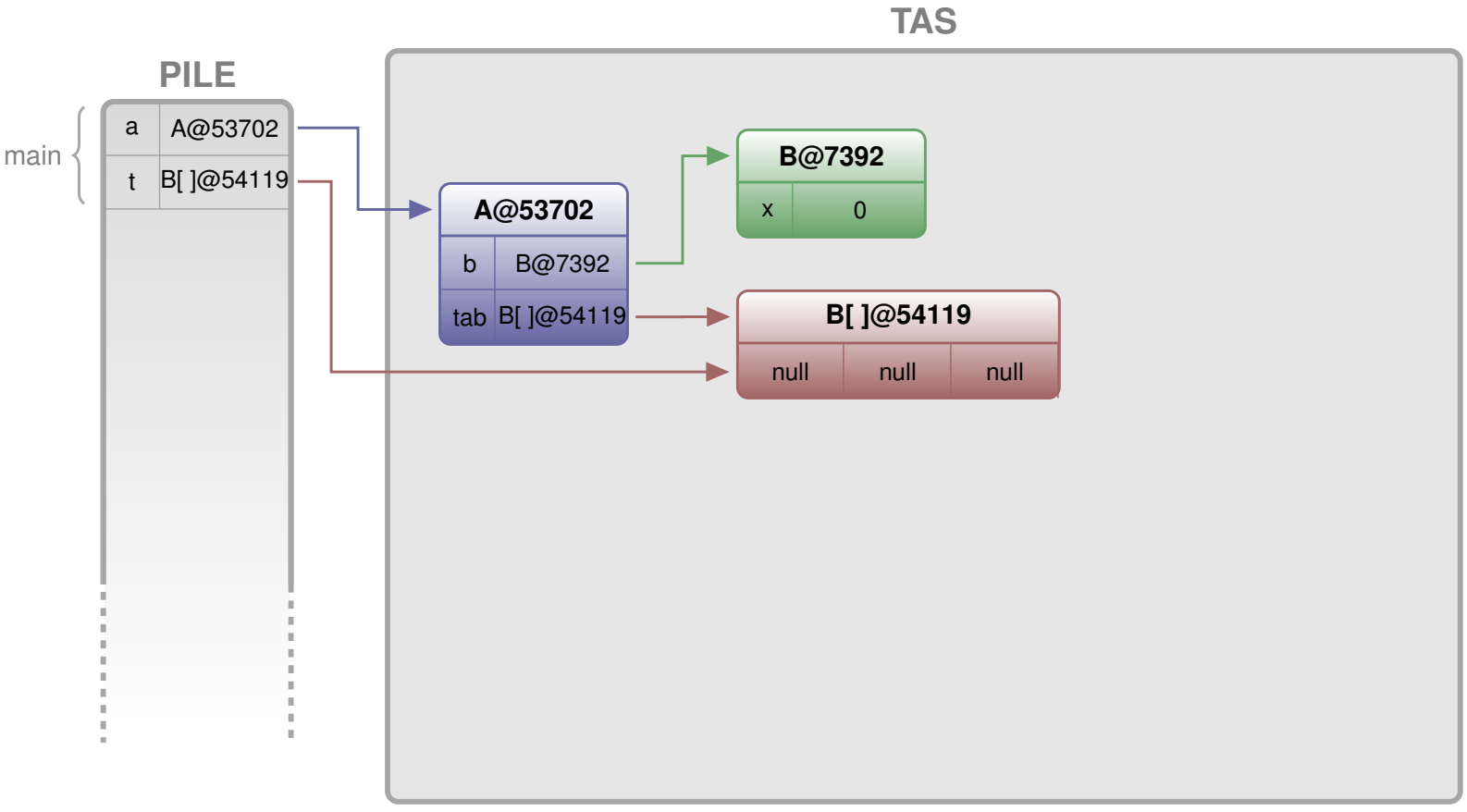
```
public class A {
  B b;
  B[] tab;
  public A() {
    b = new B();
    tab = new B[3];
    for(int i=0 ; i<tab.length ; i++ ) {
      tab[i] = new B();
    }
  }
}
```

```
public class B {
  int x;
}
```

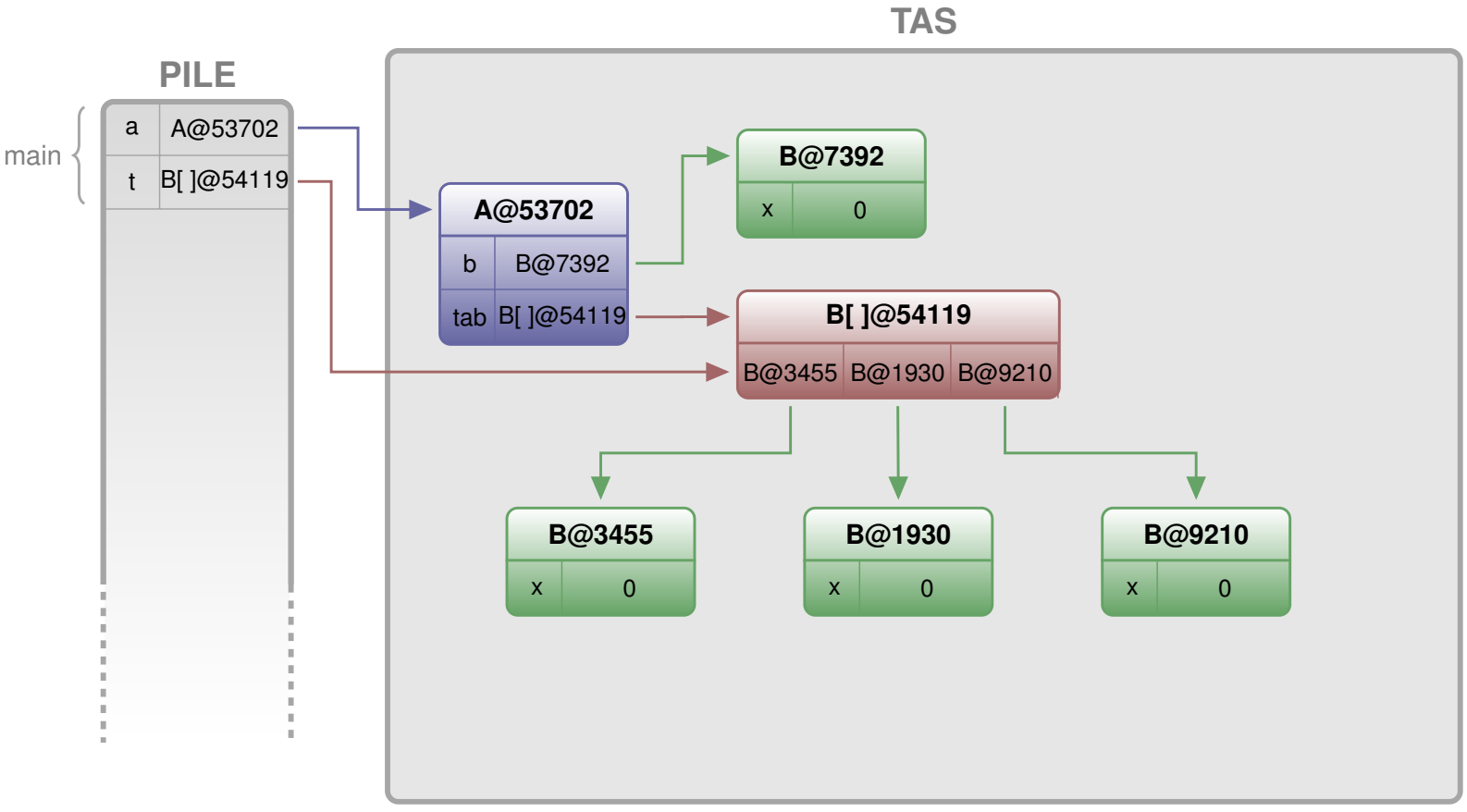
```
public class Main {
  public static void main(String[] args) {
    A a = new A();
    B[] t = a.tab ;
    t[1].x = 1 ;
    a.tab[0].x = 2 ;
    a.tab[2] = a.b ;
    a.b.x ++ ;
    for(int i=0 ; i<tab.length ; i++ ) {
      System.out.println("tab["+i+"]="+tab[i].x);
    }
  }
}
```

```
java Main
     tab[0]=2 tab[1]=1 tab[2]=0
```

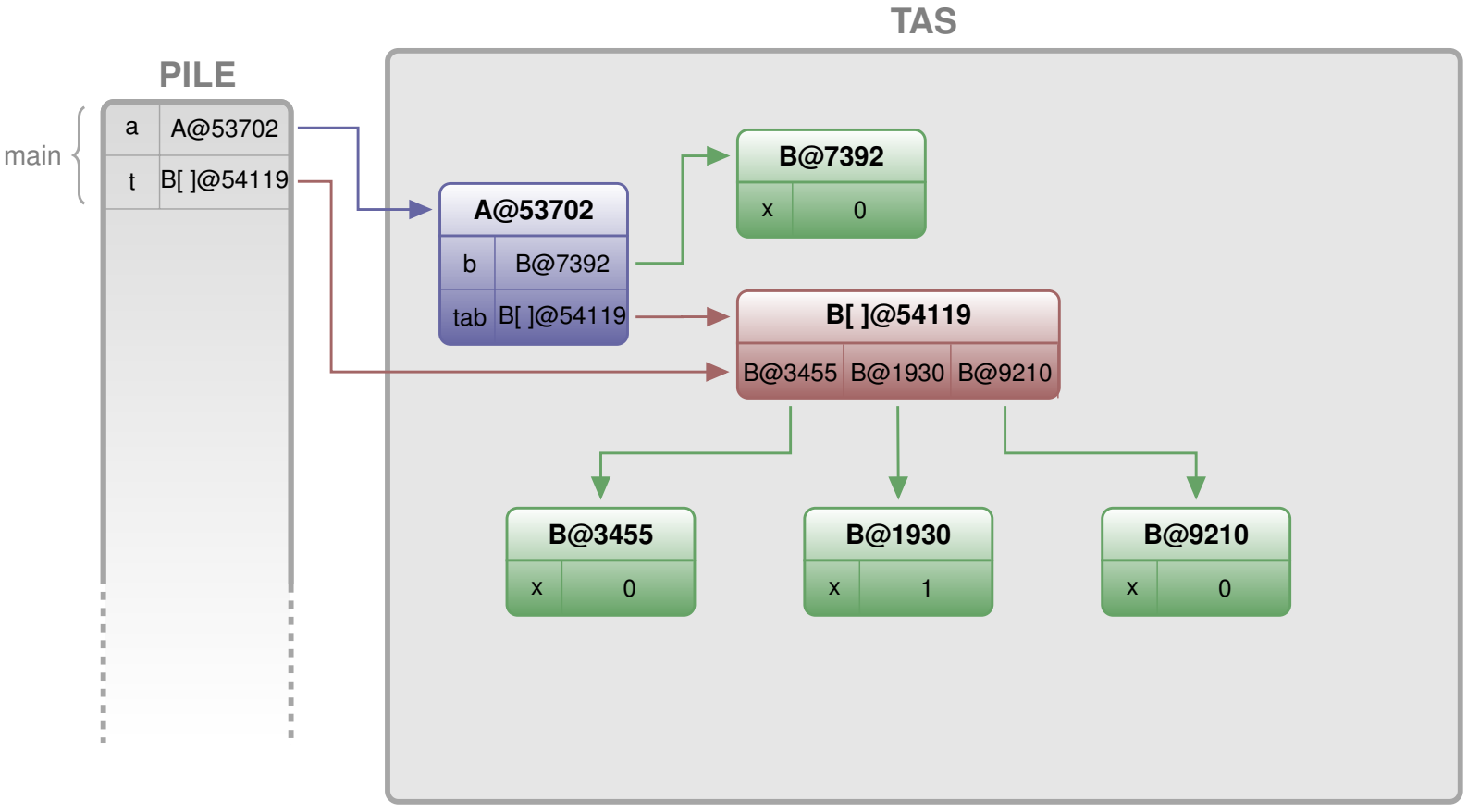
Les tableaux en Java



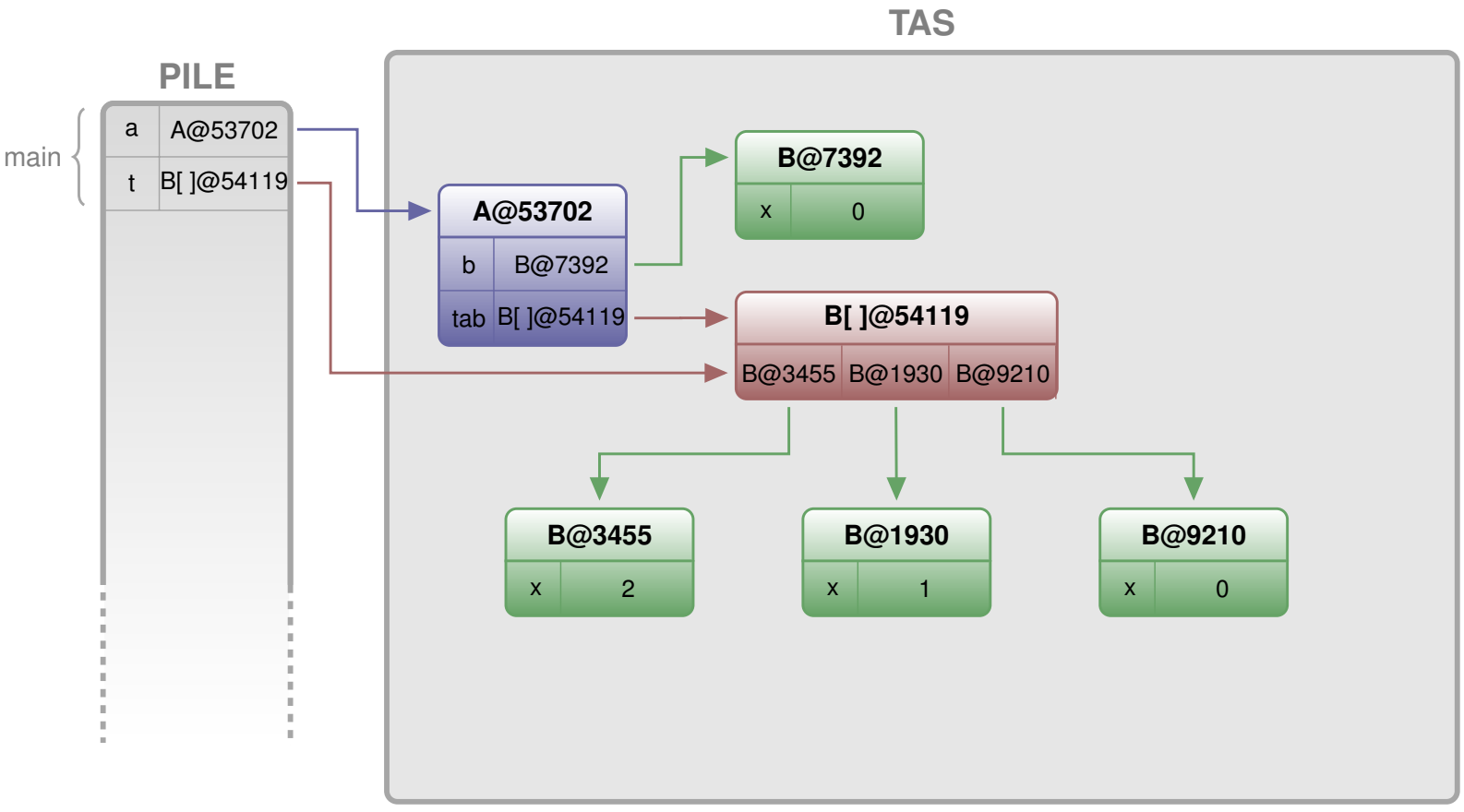
Les tableaux en Java



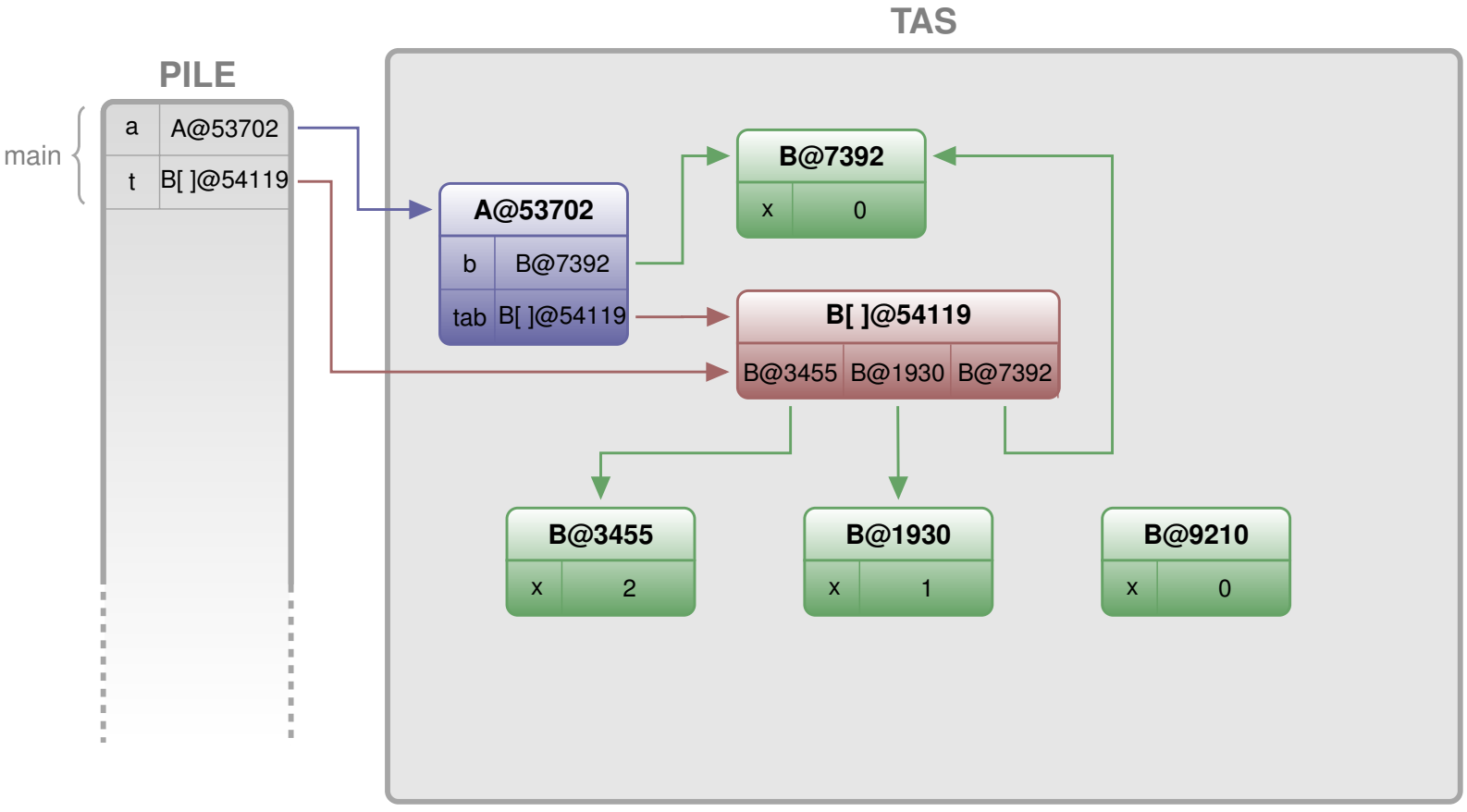
Les tableaux en Java



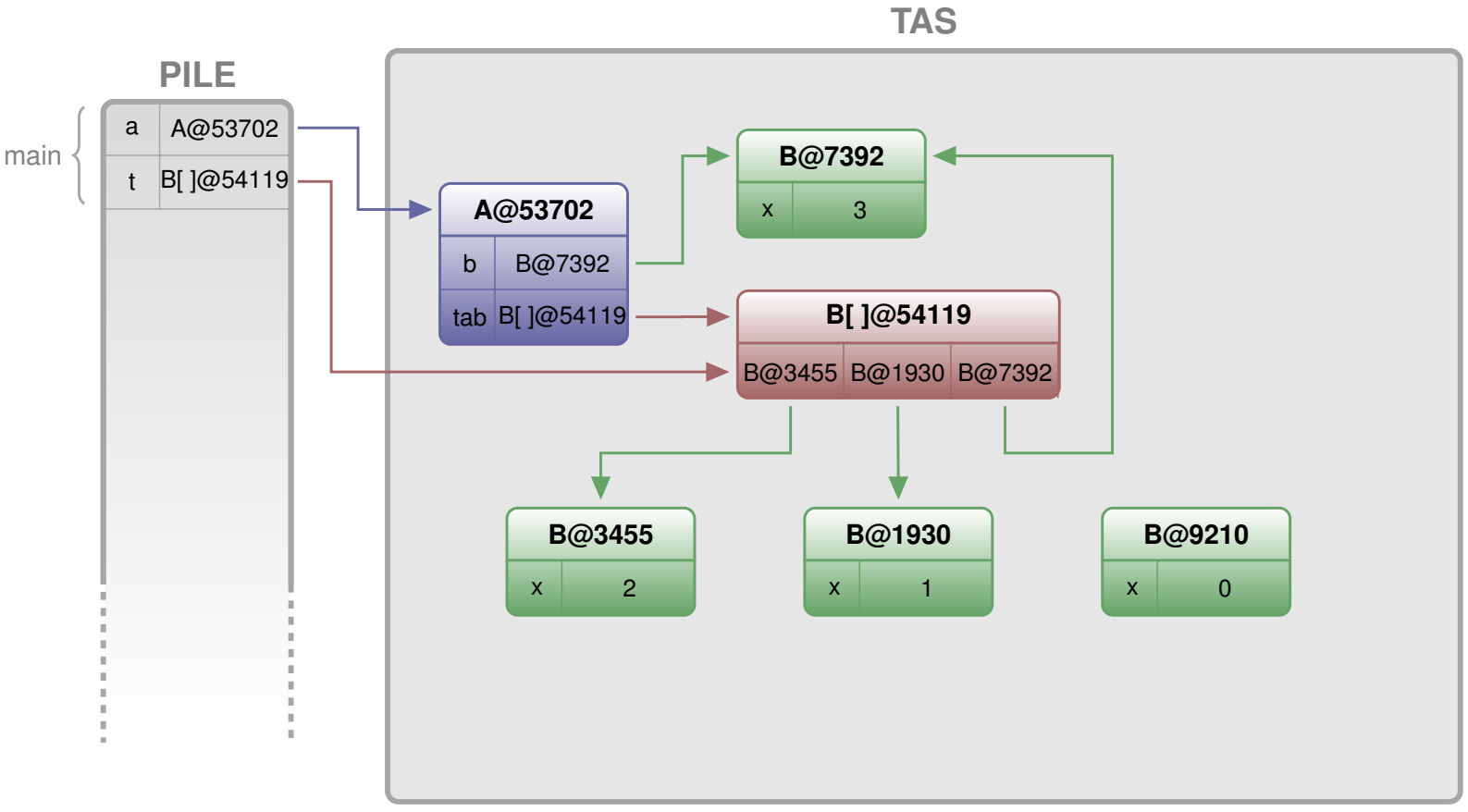
Les tableaux en Java



Les tableaux en Java



Les tableaux en Java



Outline

Outline

Les classes enveloppes : la classe Long.

```
public class A {
  Long i;
  B b;
  public A(int x,int y){
    i = x;
    b = new B(y);
  }
}
```

```
public class B {
  long i;
  public B(int y) {
    i = y;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    A a2 = new A(1,2);
    A a2 = new A(3,4);
    a1.b.i = a2.b.i ;
    a1.i = a2.i;
    a1.b.i ++;
    a1.i ++;
    System.out.println("a1.i=" + a1.i);
    System.out.println("a1.b1.i=" + a1.b.i);
    System.out.println("a2.i=" + a2.i);
    System.out.println("a2.b.i=" + a2.b.i);
  }
}
```

```
java Main
```

Les classes enveloppes : la classe Long.

```
public class A {
  Long i;
  B b;
  public A(int x,int y){
    i = x;
    b = new B(y);
  }
}
```

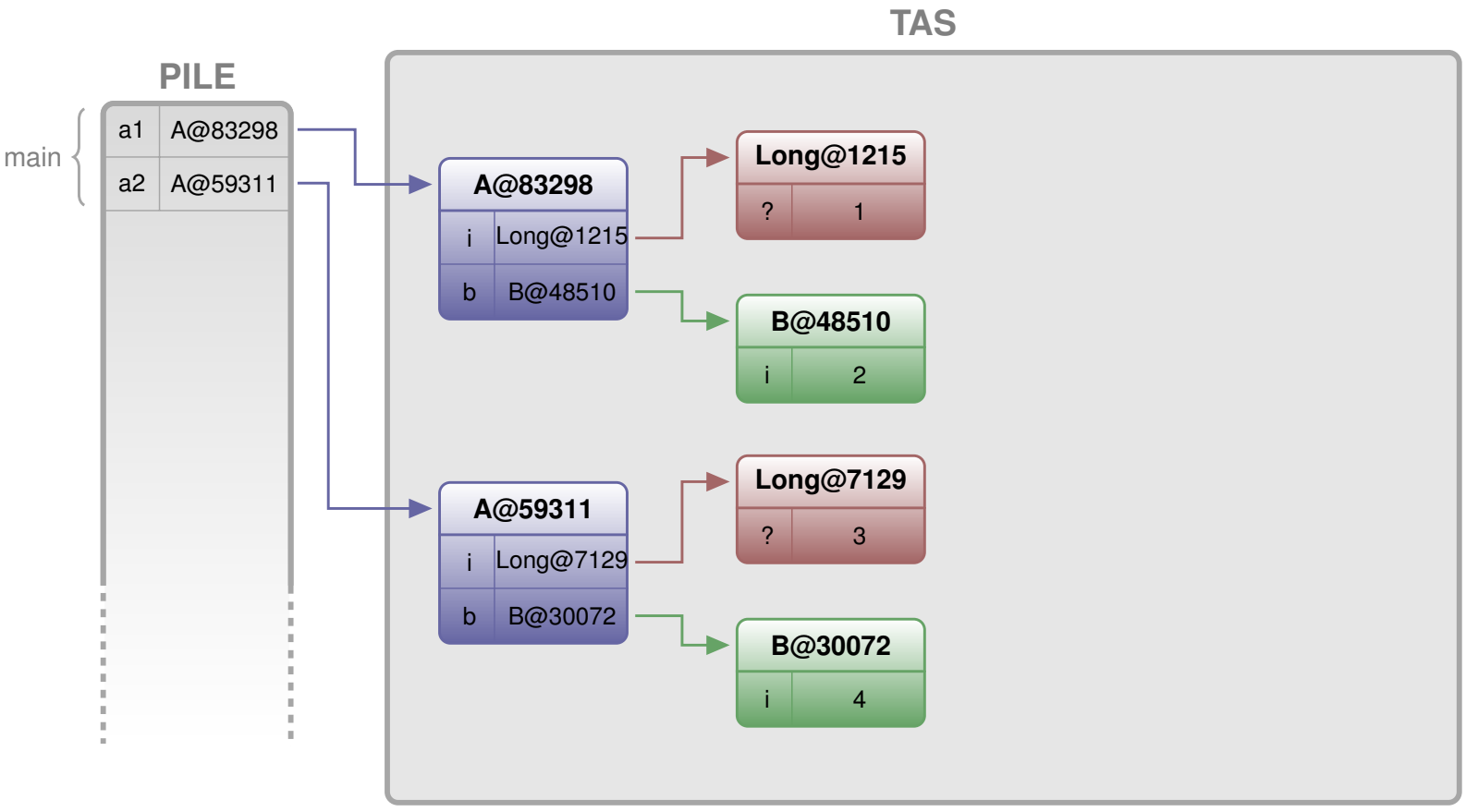
```
public class B {
  long i;
  public B(int y) {
    i = y;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    A a2 = new A(1,2);
    A a2 = new A(3,4);
    a1.b.i = a2.b.i ;
    a1.i = a2.i;
    a1.b.i ++;
    a1.i ++;
    System.out.println("a1.i=" + a1.i);
    System.out.println("a1.b1.i=" + a1.b.i);
    System.out.println("a2.i=" + a2.i);
    System.out.println("a2.b.i=" + a2.b.i);
  }
}
```

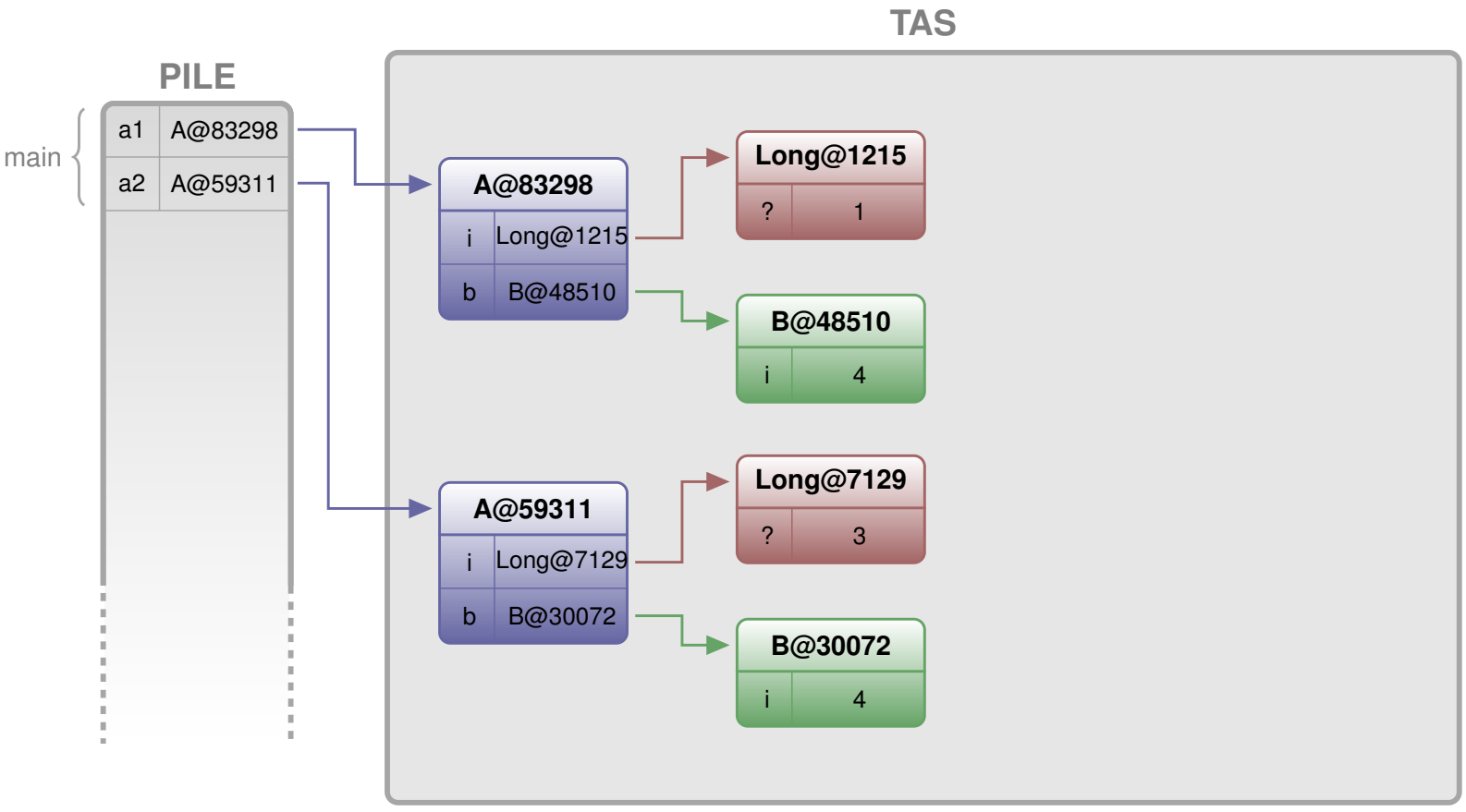
```
java Main
    a1.i=4
    a1.b.i=5
    a2.i=3
    a2.b.i=4
```

Les classes enveloppes : la classe Long.

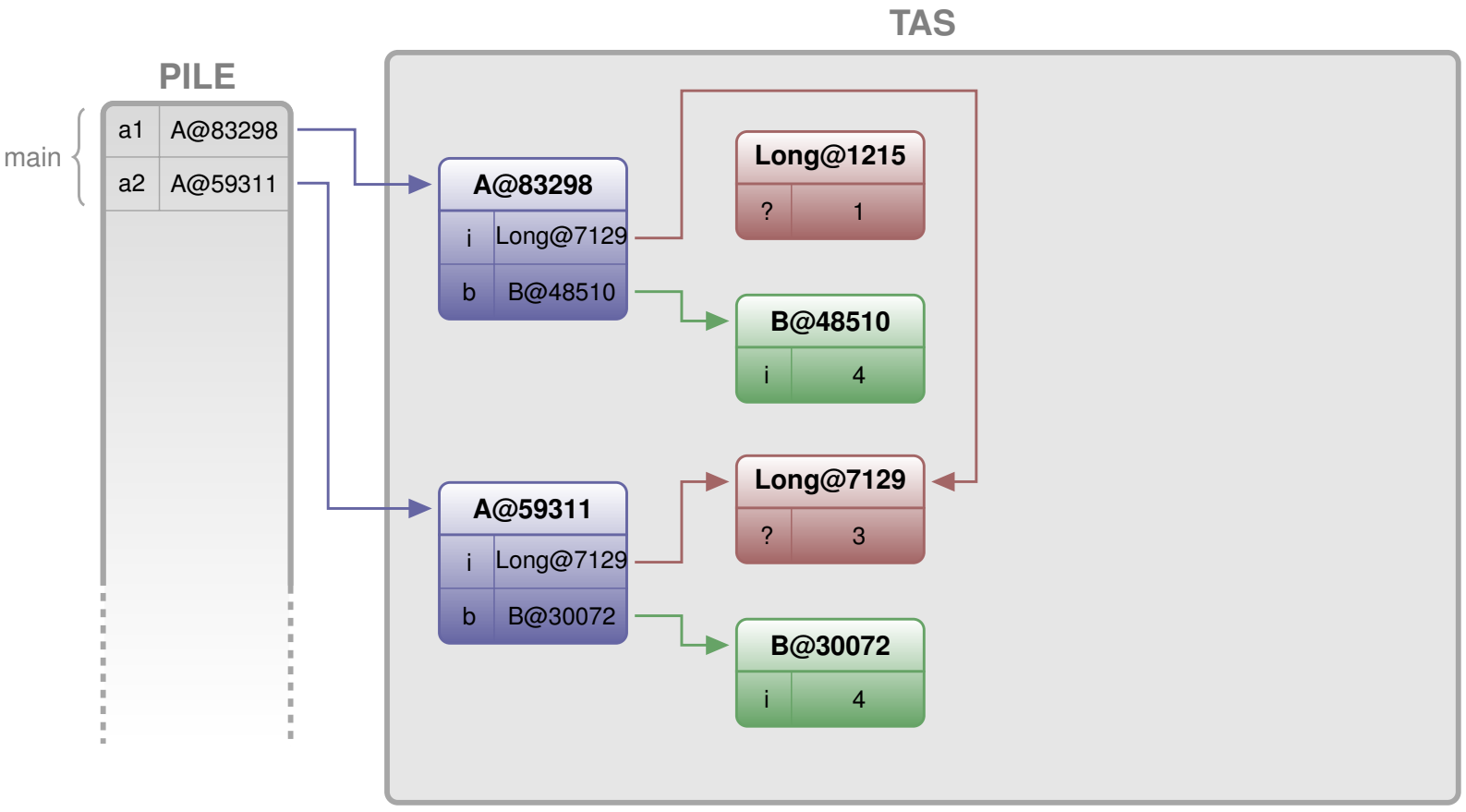
Les classes enveloppes : la classe Long.



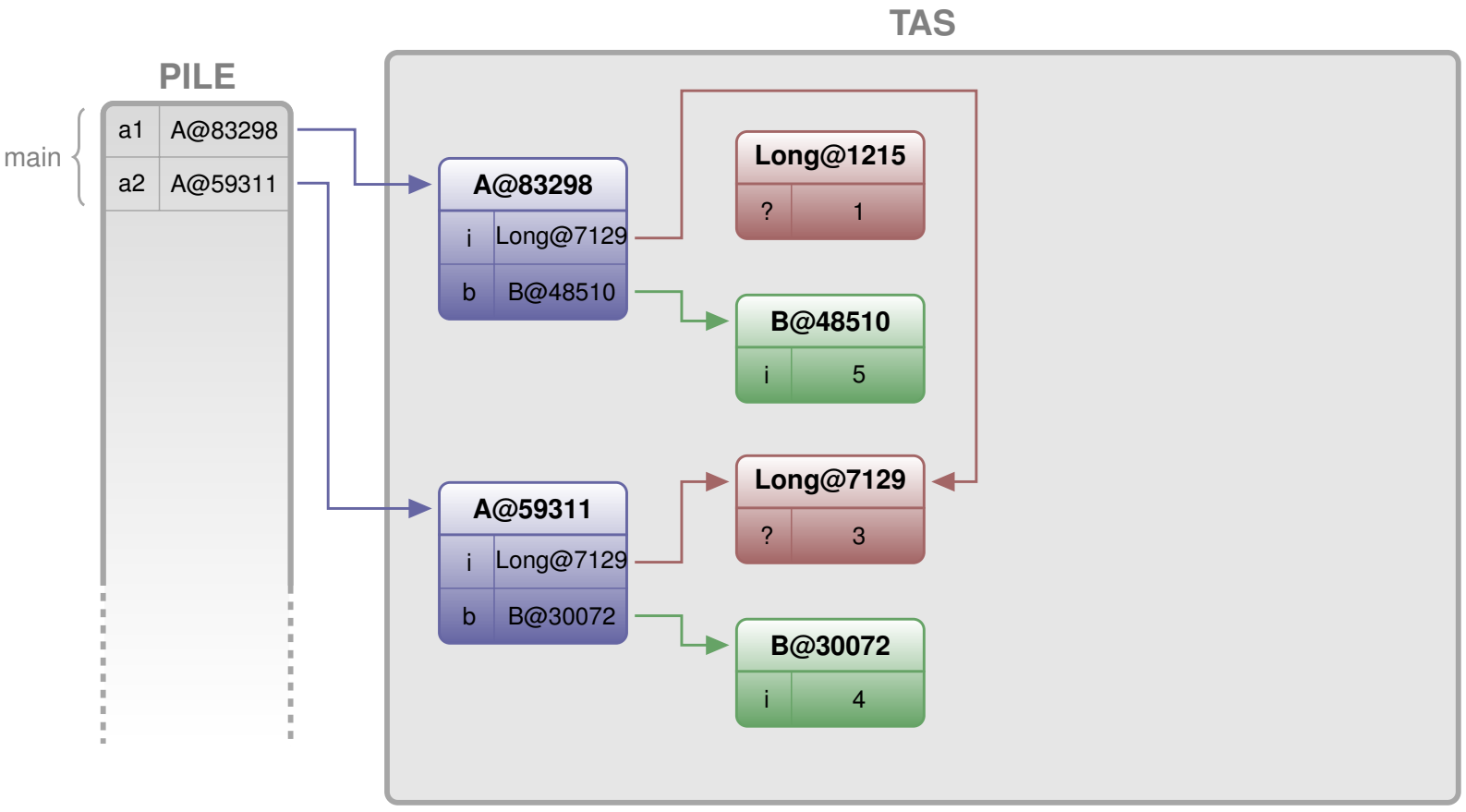
Les classes enveloppes : la classe Long.



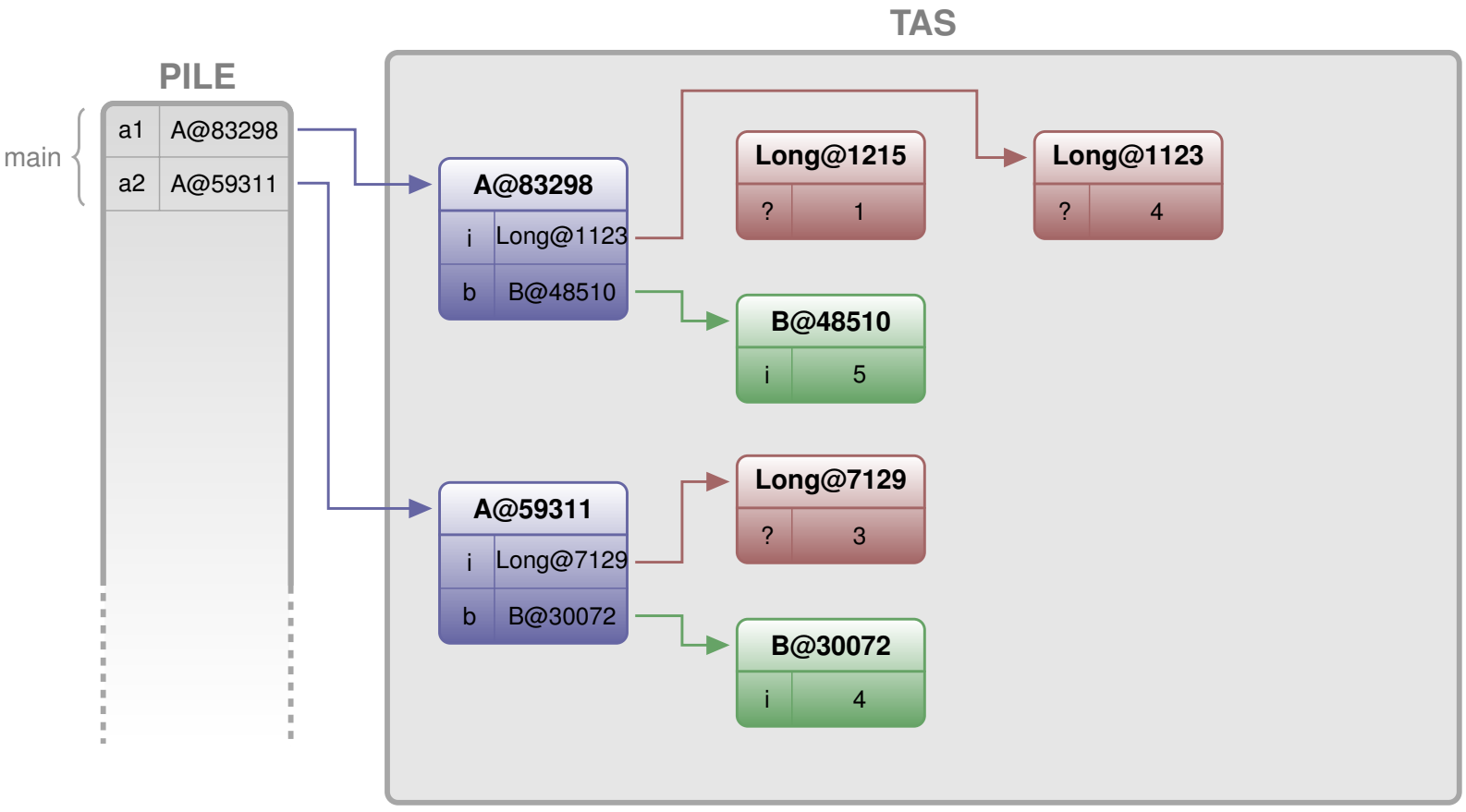
Les classes enveloppes : la classe Long.



Les classes enveloppes : la classe Long.



Les classes enveloppes : la classe Long.



Outline

Le problème classique du toString

```java
public class A {
  int [] tab ;
  public A(int x) {
    tab = new int[x];
    for (int i = 0; i < tab.length; i++) {
      tab[i]=i+1;
    }
  }
  @Override
  public String toString() {
    String tmp = "";
    if (tab.length != 0){
      tmp = "[" + tab[0];
      for (int i = 1; i < tab.length; i++) {
        tmp += "|" + tab[i];
      }
      tmp += "]";
    }
    return tmp;
  }
}
```

```java
public class Main {
  public static void main(String[] args) {
    A a = new A(4);
    System.out.println("tab -> " + a);
  }
}
```

```
java Main
```

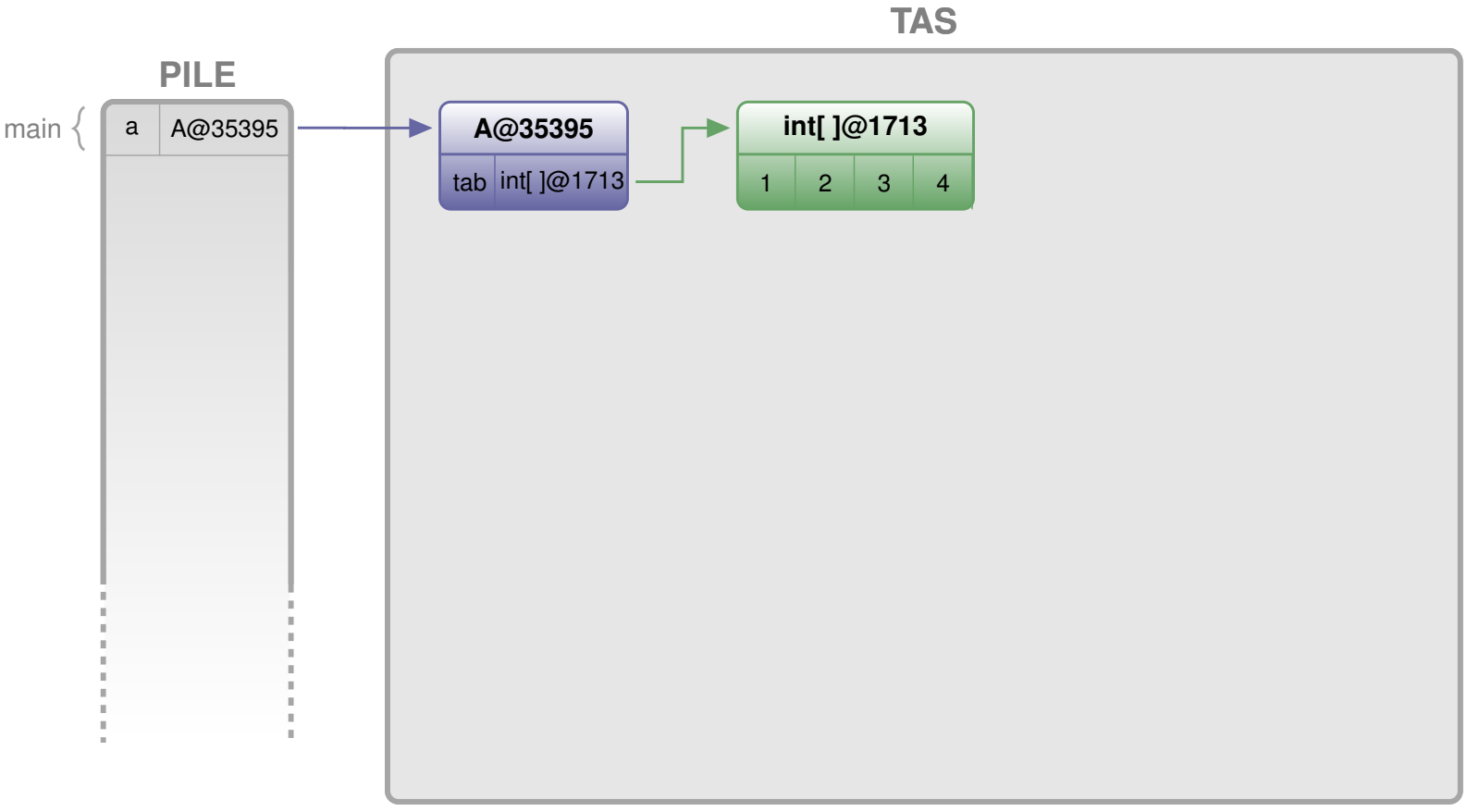
Le problème classique du toString

```java
public class A {
  int [] tab ;
  public A(int x) {
    tab = new int[x];
    for (int i = 0; i < tab.length; i++) {
      tab[i]=i+1;
    }
  }
  @Override
  public String toString() {
    String tmp = "";
    if (tab.length != 0){
      tmp = "[" + tab[0];
      for (int i = 1; i < tab.length; i++) {
        tmp += "|" + tab[i];
      }
      tmp += "]";
    }
    return tmp;
  }
}
```

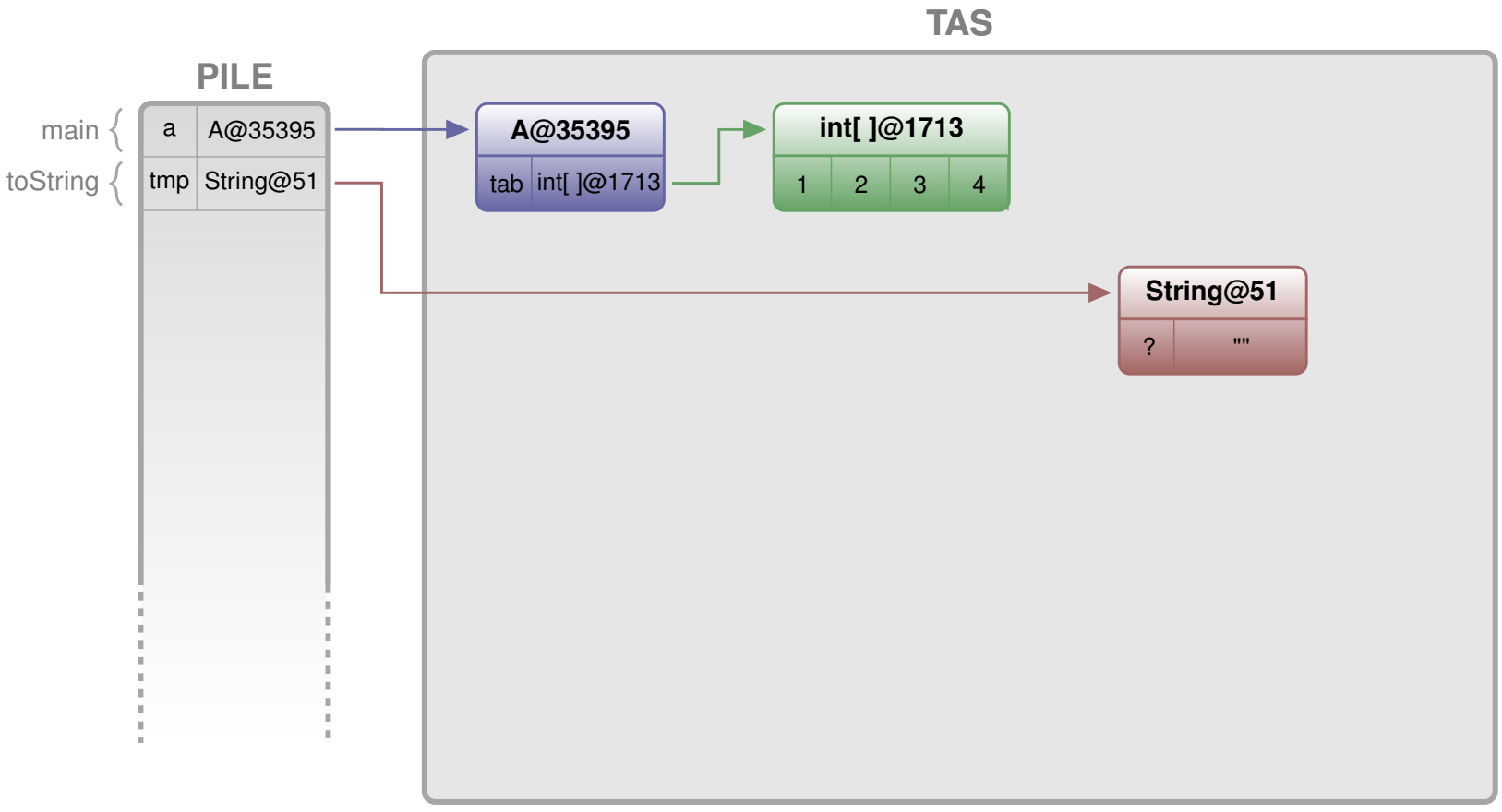
```java
public class Main {
  public static void main(String[] args) {
    A a = new A(4);
    System.out.println("tab -> " + a);
  }
}
```

```
java Main
    tab -> [1|2|3|4]
```

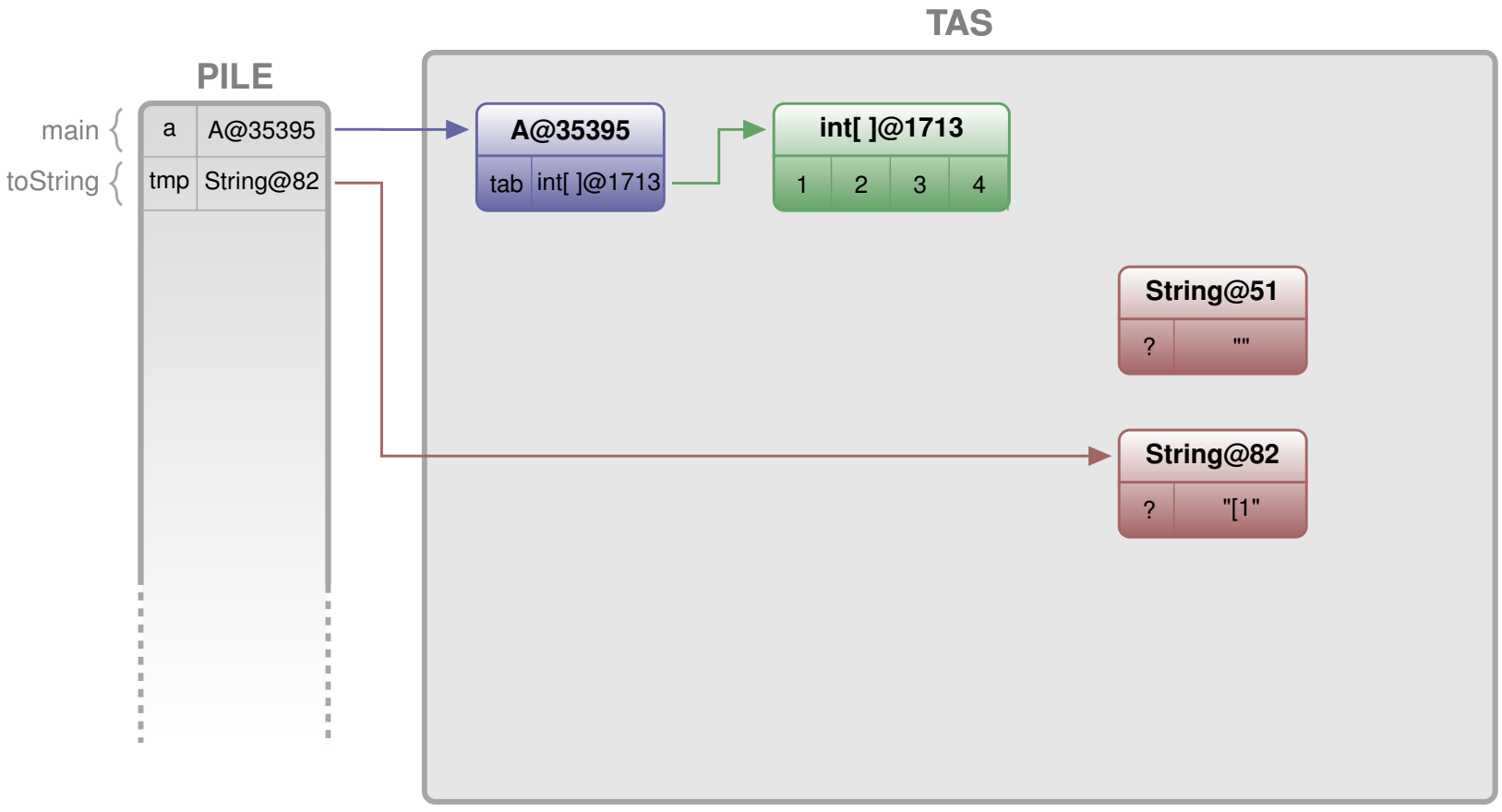
Le problème classique du toString



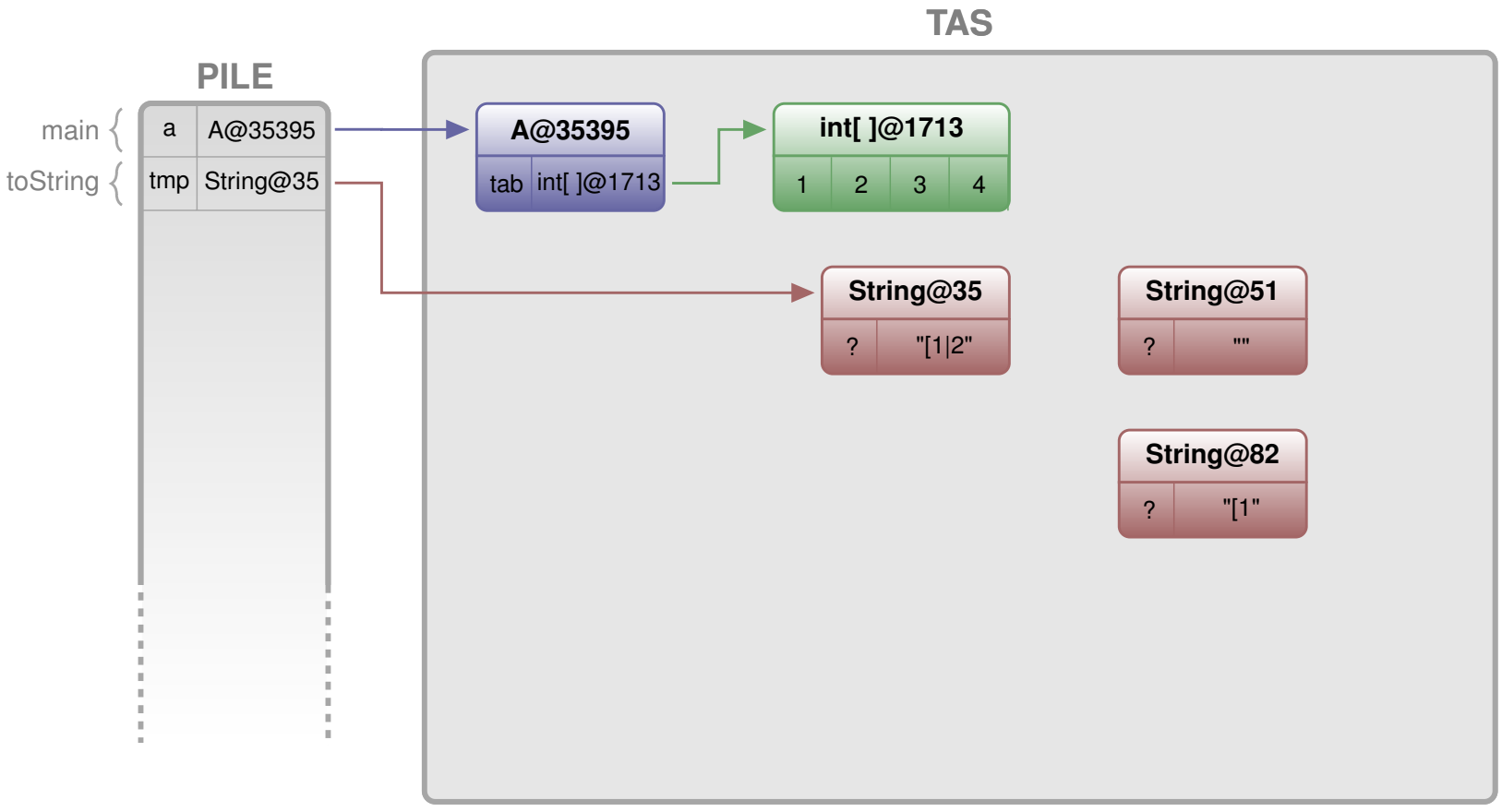
Le problème classique du toString



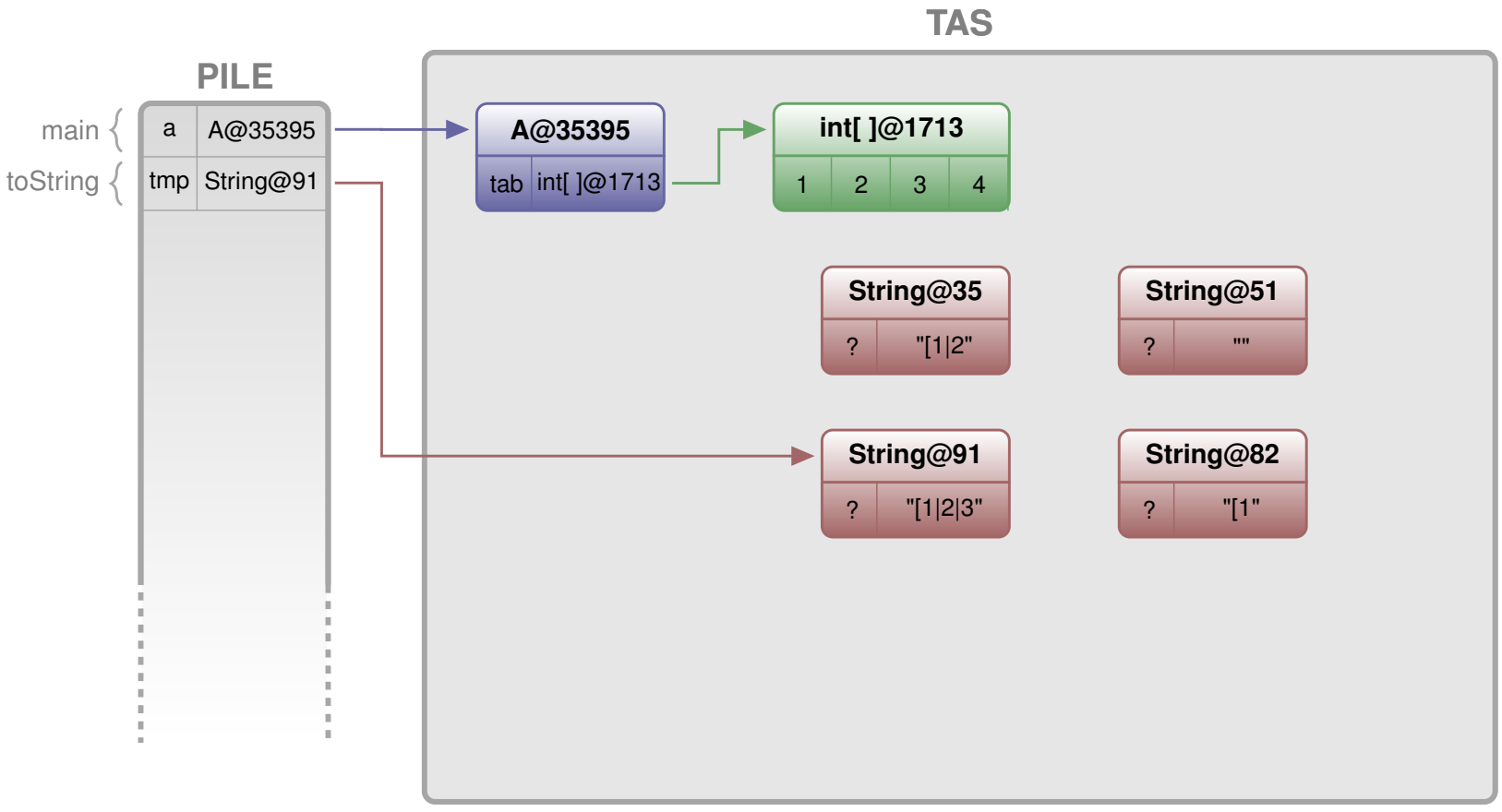
Le problème classique du toString



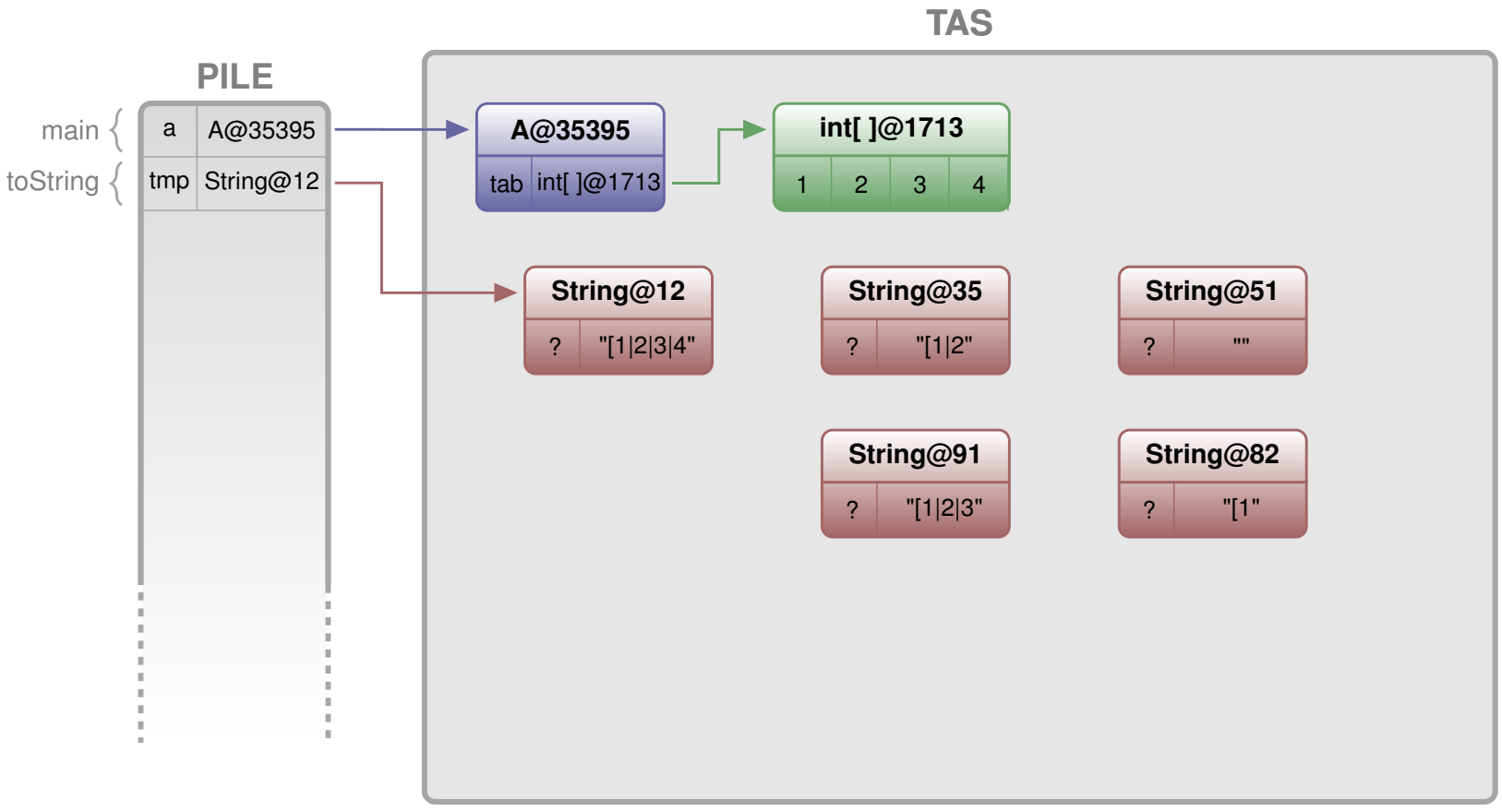
Le problème classique du toString



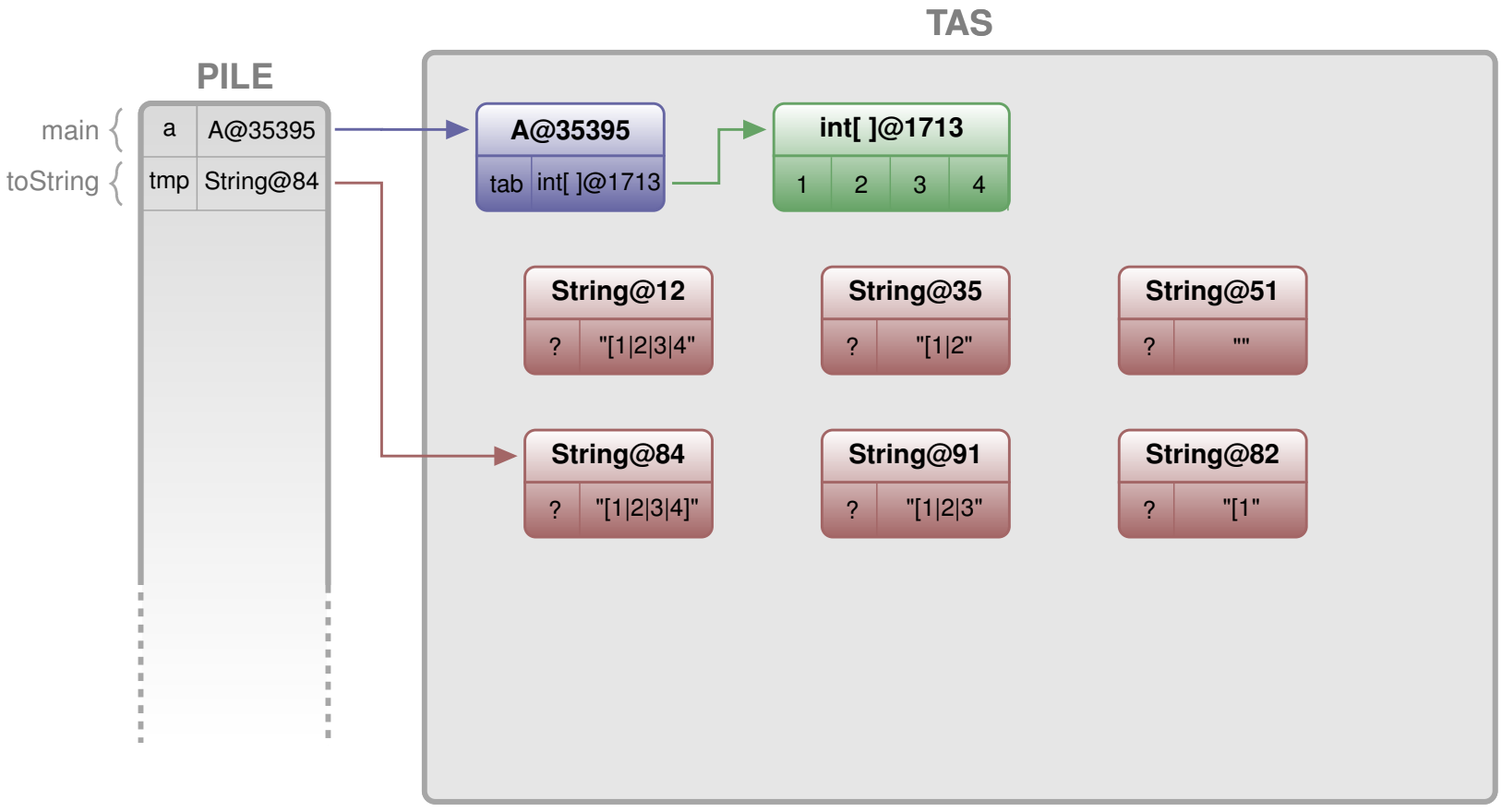
Le problème classique du toString



Le problème classique du toString



Le problème classique du toString



De l'intérêt du StringBuffer

```
public class A {
  int [] tab ;
  public A(int x) {
    tab = new int[x];
    for (int i = 0; i < tab.length; i++) {
      tab[i]=i+1;
    }
  }
  @Override
  public String toString() {
    StringBuffer tmp = "";
    if (tab.length != 0){
      tmp = "[" + tab[0];
      for (int i = 1; i < tab.length; i++) {
        tmp += "|" + tab[i];
      }
      tmp += "]";
    }
    return tmp;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    A a = new A(4);
    System.out.println("tab -> " + a);
  }
}
```

```
javac A.java
```

De l'intérêt du StringBuffer

```java
public class A {
  int [] tab ;
  public A(int x) {
    tab = new int[x];
    for (int i = 0; i < tab.length; i++) {
      tab[i]=i+1;
    }
  }
  @Override
  public String toString() {
    StringBuffer tmp = "";
    if (tab.length != 0){
      tmp = "[" + tab[0];
      for (int i = 1; i < tab.length; i++) {
        tmp += "|" + tab[i];
      }
      tmp += "]";
    }
    return tmp;
  }
}
```

```java
public class Main {
  public static void main(String[] args) {
    A a = new A(4);
    System.out.println("tab -> " + a);
  }
}
```

```
javac A.java
    A.java:11: error: incompatible types: String cannot be converted to StringBuffer
            StringBuffer tmp = "";
                               ^
```

De l'intérêt du StringBuffer

```
public class A {
  int [] tab ;
  public A(int x) {
    tab = new int[x];
    for (int i = 0; i < tab.length; i++) {
      tab[i]=i+1;
    }
  }
  @Override
  public String toString() {
    StringBuffer tmp = new StringBuffer();
    if (tab.length != 0){
      tmp = "[" + tab[0];
      for (int i = 1; i < tab.length; i++) {
        tmp += "|" + tab[i];
      }
      tmp += "]";
    }
    return tmp;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    A a = new A(4);
    System.out.println("tab -> " + a);
  }
}
```

```
javac A.java
```

De l'intérêt du StringBuffer

```
public class A {
  int [] tab ;
  public A(int x) {
    tab = new int[x];
    for (int i = 0; i < tab.length; i++) {
      tab[i]=i+1;
    }
  }
  @Override
  public String toString() {
    StringBuffer tmp = new StringBuffer();
    if (tab.length != 0){
      tmp = "[" + tab[0];
      for (int i = 1; i < tab.length; i++) {
        tmp += "|" + tab[i];
      }
      tmp += "]";
    }
    return tmp;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    A a = new A(4);
    System.out.println("tab -> " + a);
  }
}
```

```
javac A.java
    A.java:13: error: incompatible types: String cannot be converted to StringBuffer
             tmp = "[" + tab[0];
                         ^
```

De l'intérêt du StringBuffer

```
public class A {
  int [] tab ;
  public A(int x) {
    tab = new int[x];
    for (int i = 0; i < tab.length; i++) {
      tab[i]=i+1;
    }
  }
  @Override
  public String toString() {
    StringBuffer tmp = new StringBuffer();
    if (tab.length != 0){
      tmp.append("[" + tab[0]);
      for (int i = 1; i < tab.length; i++) {
        tmp.append("," + tab[i]);
      }
      tmp.append("]");
    }
    return tmp;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    A a = new A(4);
    System.out.println("tab -> " + a);
  }
}
```

```
javac A.java
```

De l'intérêt du StringBuffer

```
public class A {
  int [] tab ;
  public A(int x) {
    tab = new int[x];
    for (int i = 0; i < tab.length; i++) {
      tab[i]=i+1;
    }
  }
  @Override
  public String toString() {
    StringBuffer tmp = new StringBuffer();
    if (tab.length != 0){
      tmp.append("[" + tab[0]);
      for (int i = 1; i < tab.length; i++) {
        tmp.append("," + tab[i]);
      }
      tmp.append("]");
    }
    return tmp;
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    A a = new A(4);
    System.out.println("tab -> " + a);
  }
}
```

```
javac A.java
    A.java:19: error: incompatible types: StringBuffer cannot be converted to String
             return tmp;
                    ^
```

De l'intérêt du StringBuffer

```
public class A {
  int [] tab ;
  public A(int x) {
    tab = new int[x];
    for (int i = 0; i < tab.length; i++) {
      tab[i]=i+1;
    }
  }
  @Override
  public String toString() {
    StringBuffer tmp = new StringBuffer();
    if (tab.length != 0){
      tmp.append("[" + tab[0]);
      for (int i = 1; i < tab.length; i++) {
        tmp.append("," + tab[i]);
      }
      tmp.append("]");
    }
    return tmp.toString();
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    A a = new A(4);
    System.out.println("tab -> " + a);
  }
}
```

```
javac A.java
```

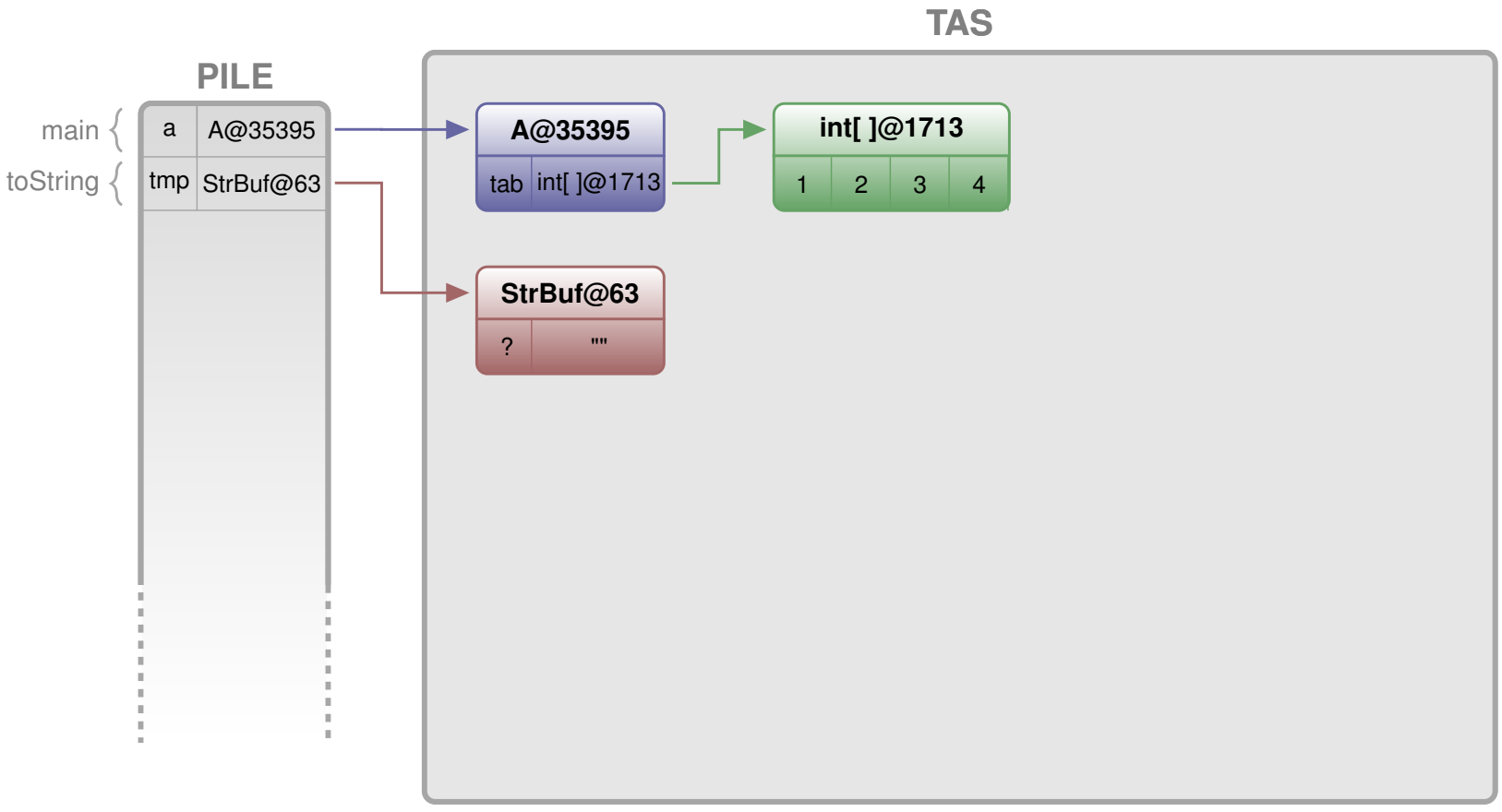
De l'intérêt du StringBuffer

```
public class A {
  int [] tab ;
  public A(int x) {
    tab = new int[x];
    for (int i = 0; i < tab.length; i++) {
      tab[i]=i+1;
    }
  }
  @Override
  public String toString() {
    StringBuffer tmp = new StringBuffer();
    if (tab.length != 0){
      tmp.append("[" + tab[0]);
      for (int i = 1; i < tab.length; i++) {
        tmp.append("," + tab[i]);
      }
      tmp.append("]");
    }
    return tmp.toString();
  }
}
```

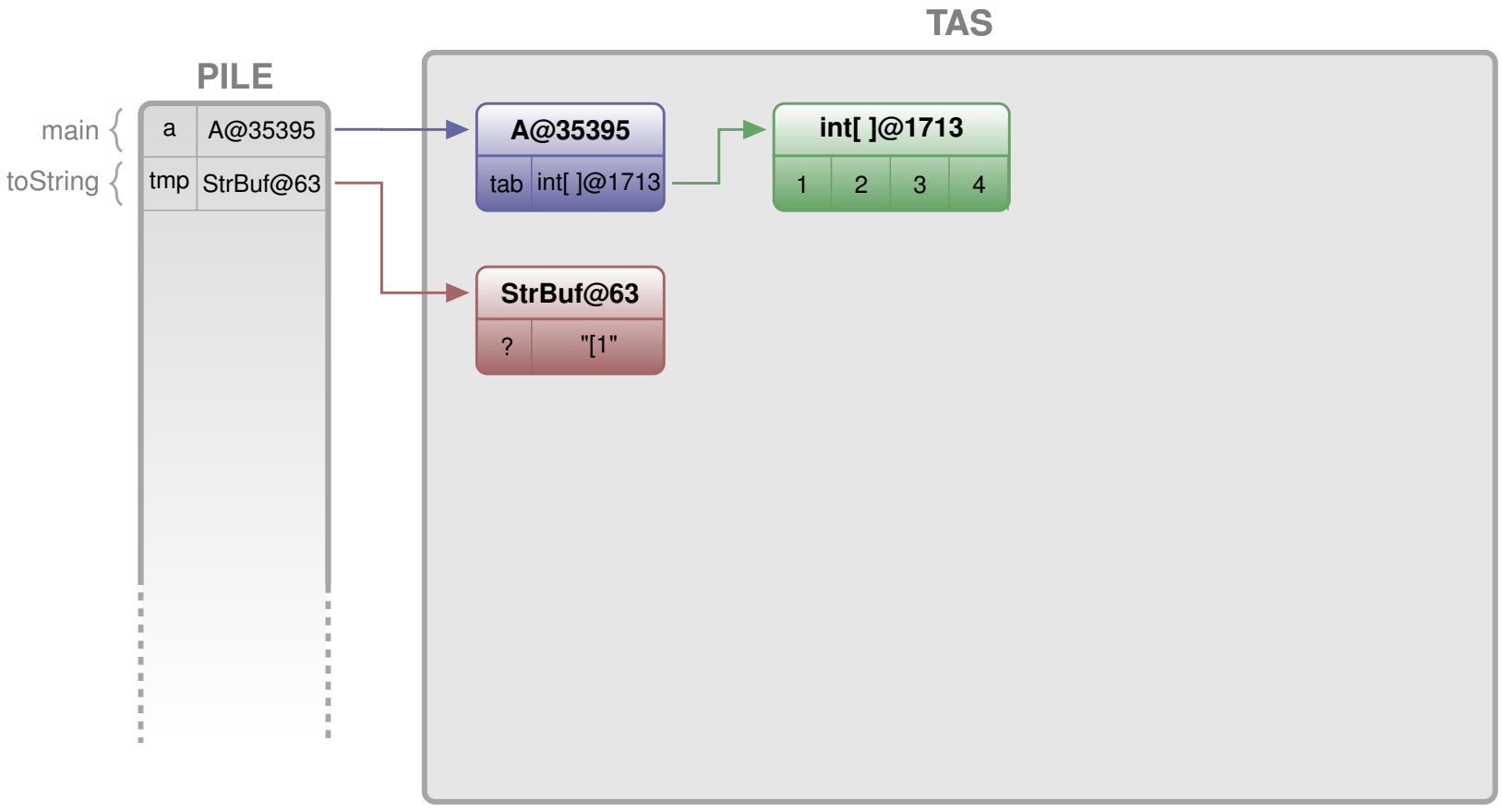
```
public class Main {
  public static void main(String[] args) {
    A a = new A(4);
    System.out.println("tab -> " + a);
  }
}
```

```
javac A.java
java A
    [1|2|3|4]
```

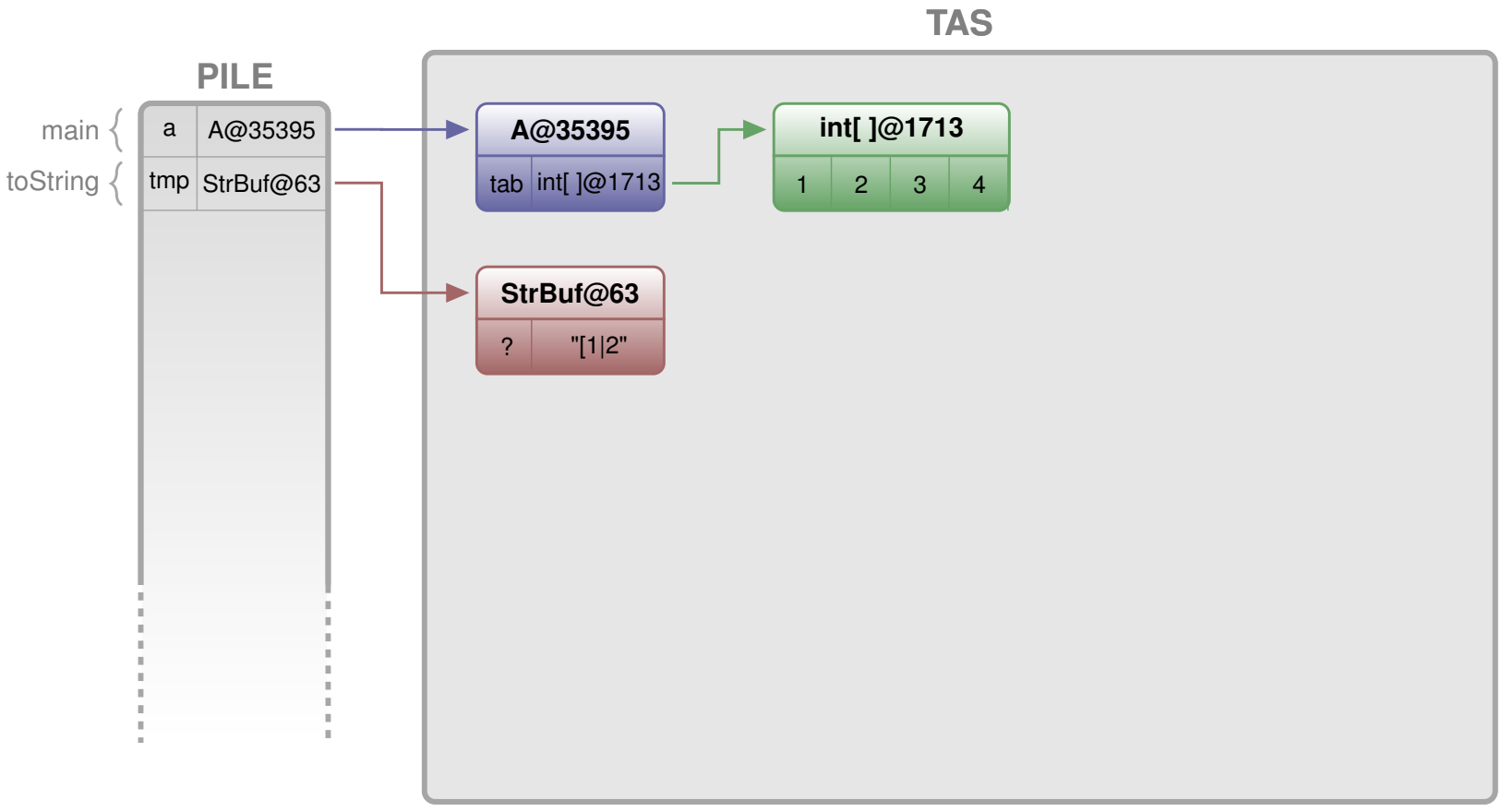
De l'intérêt du StringBuffer



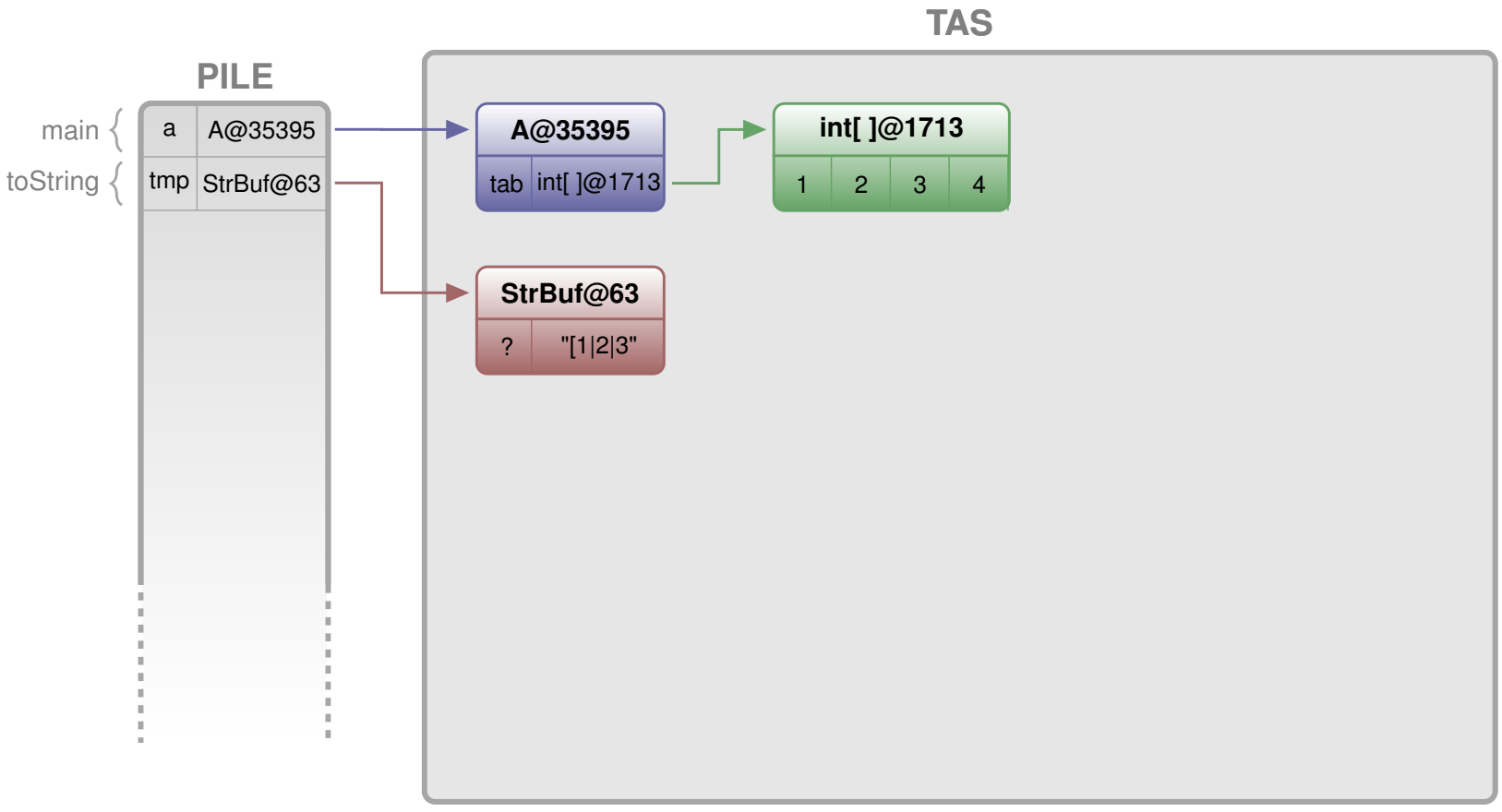
De l'intérêt du StringBuffer



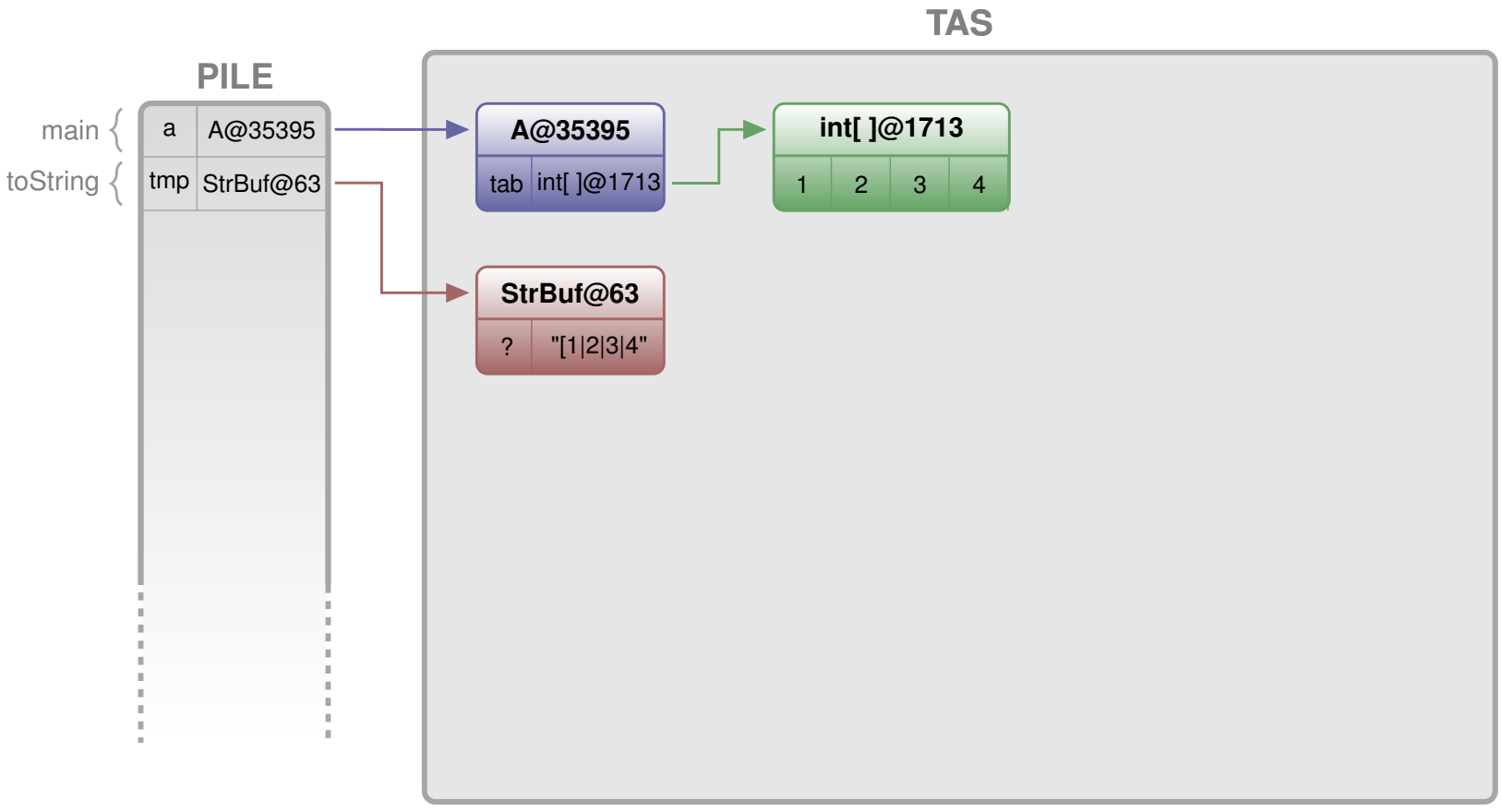
De l'intérêt du StringBuffer



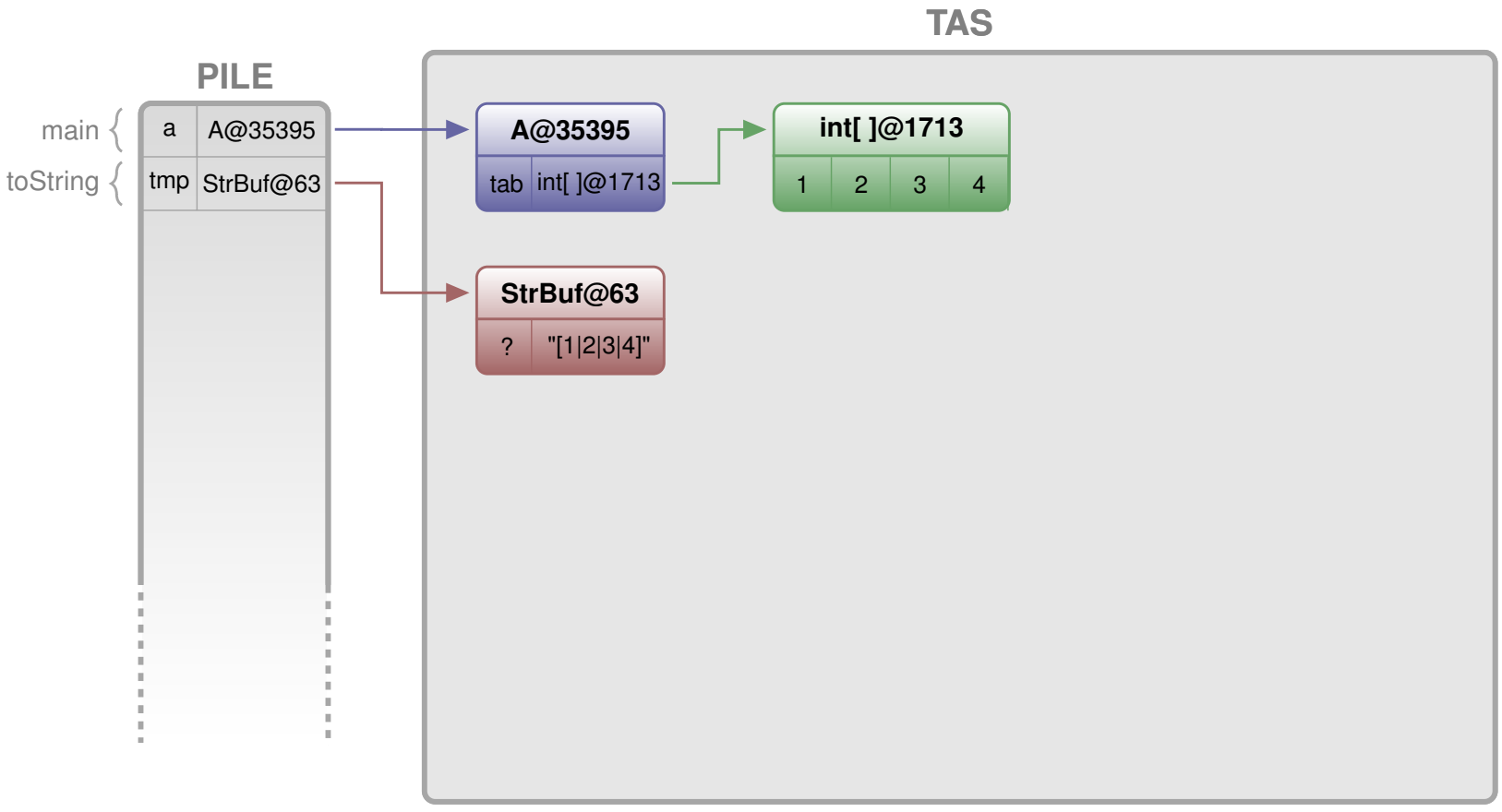
De l'intérêt du StringBuffer



De l'intérêt du StringBuffer



De l'intérêt du StringBuffer



Outline

Outline

Méthodes et blocs static

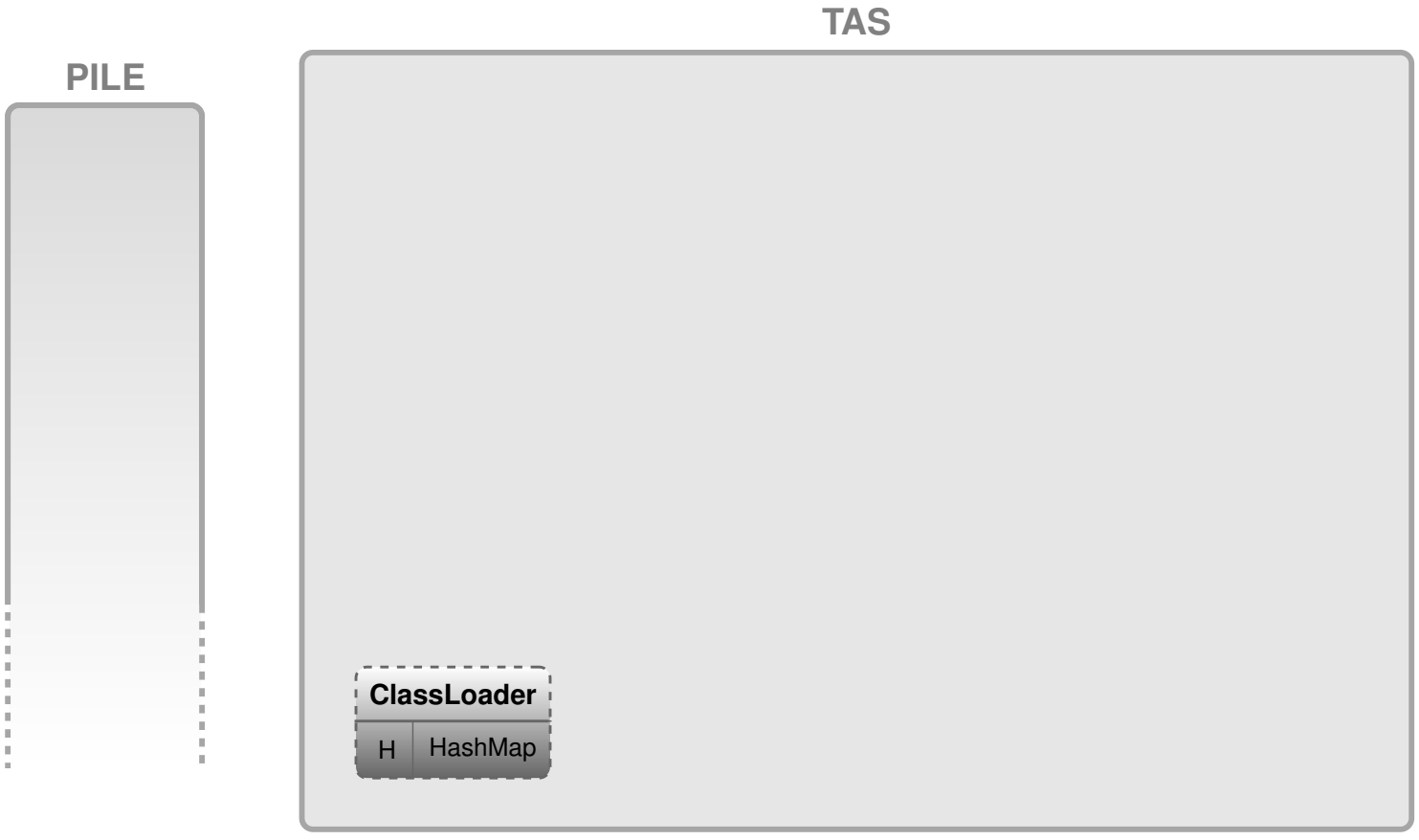
```
public class A {
  static B b = new B(0) ;
  static int i;
  int j;
  B b;
  public A(int j) {
    this.j=j ;
    b = A.b ;
    A.b = new B(j);
    i++;
  }
}
```

```
public class B {
  static int[] tab = new int[3];
  int i;
  static {
    for (int x = 0; x < tab.length; x++) {
      tab[x] = x;
    }
  }
  public B(int i){
    this.i = i;
  }
}
```

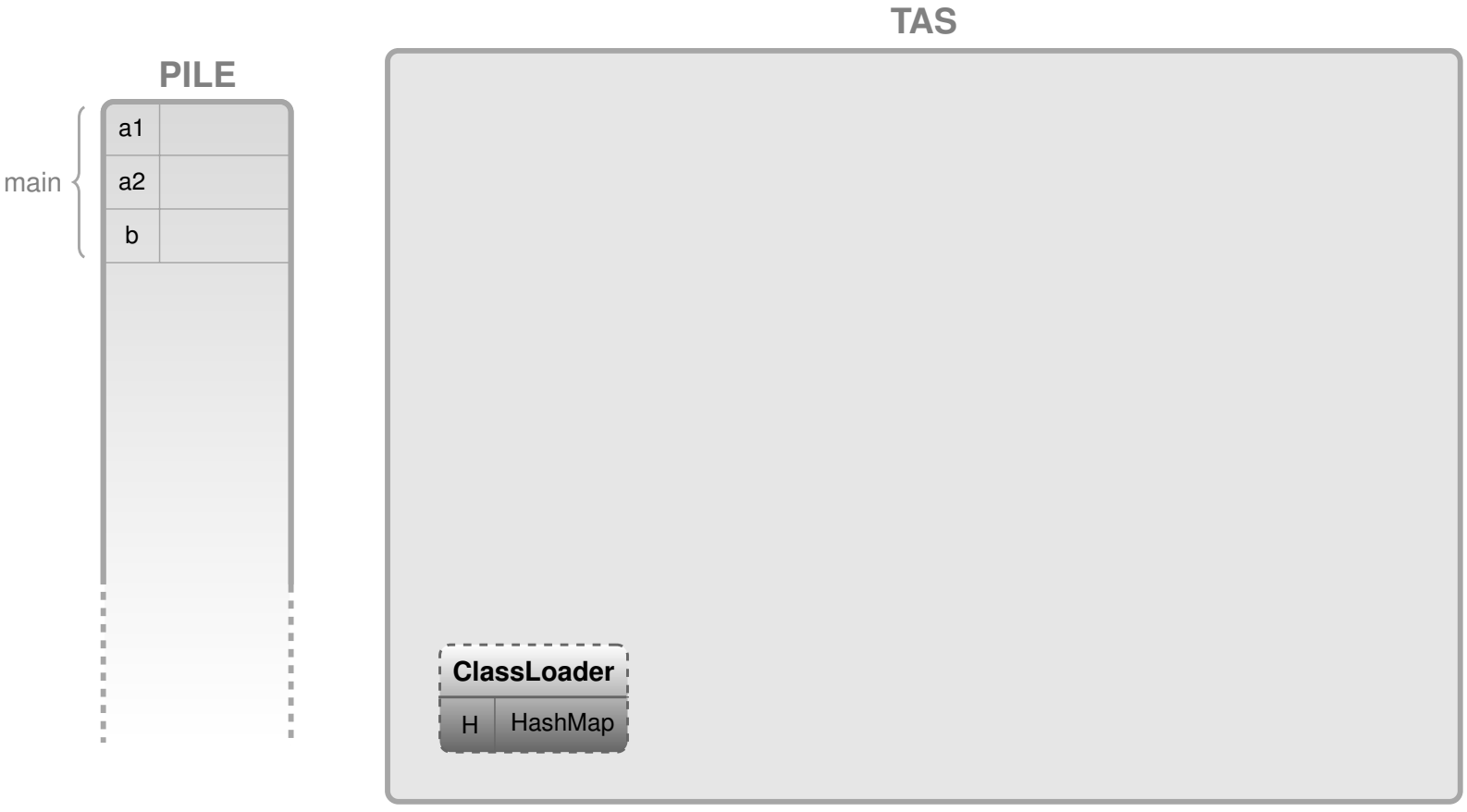
```
public class Main {
  public static void main(String[] args) {
    A a1 = new A(3);
    A a2 = new A(5);
    B b = new B(2);
  }
}
```

Méthodes et blocs static

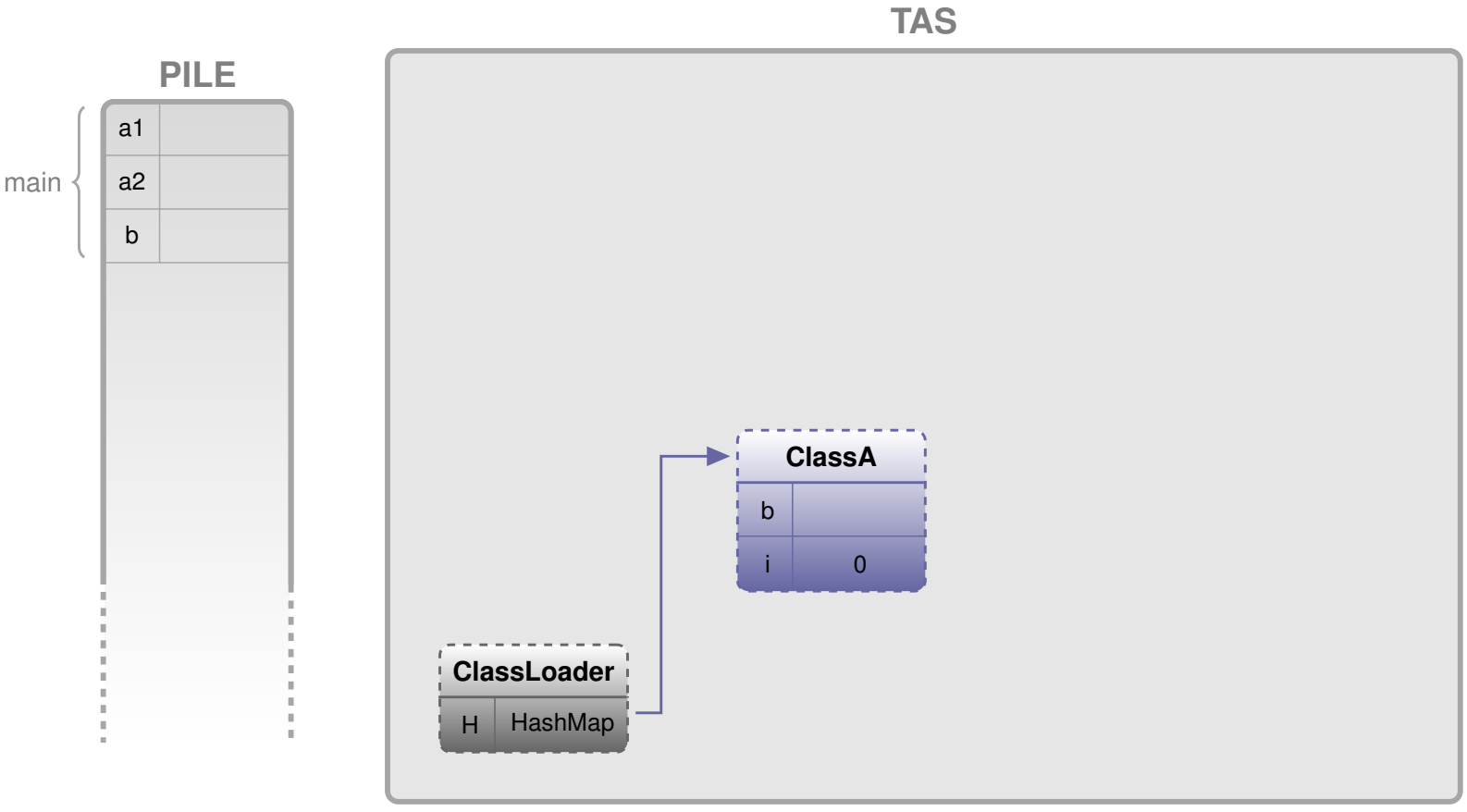
Méthodes et blocs static



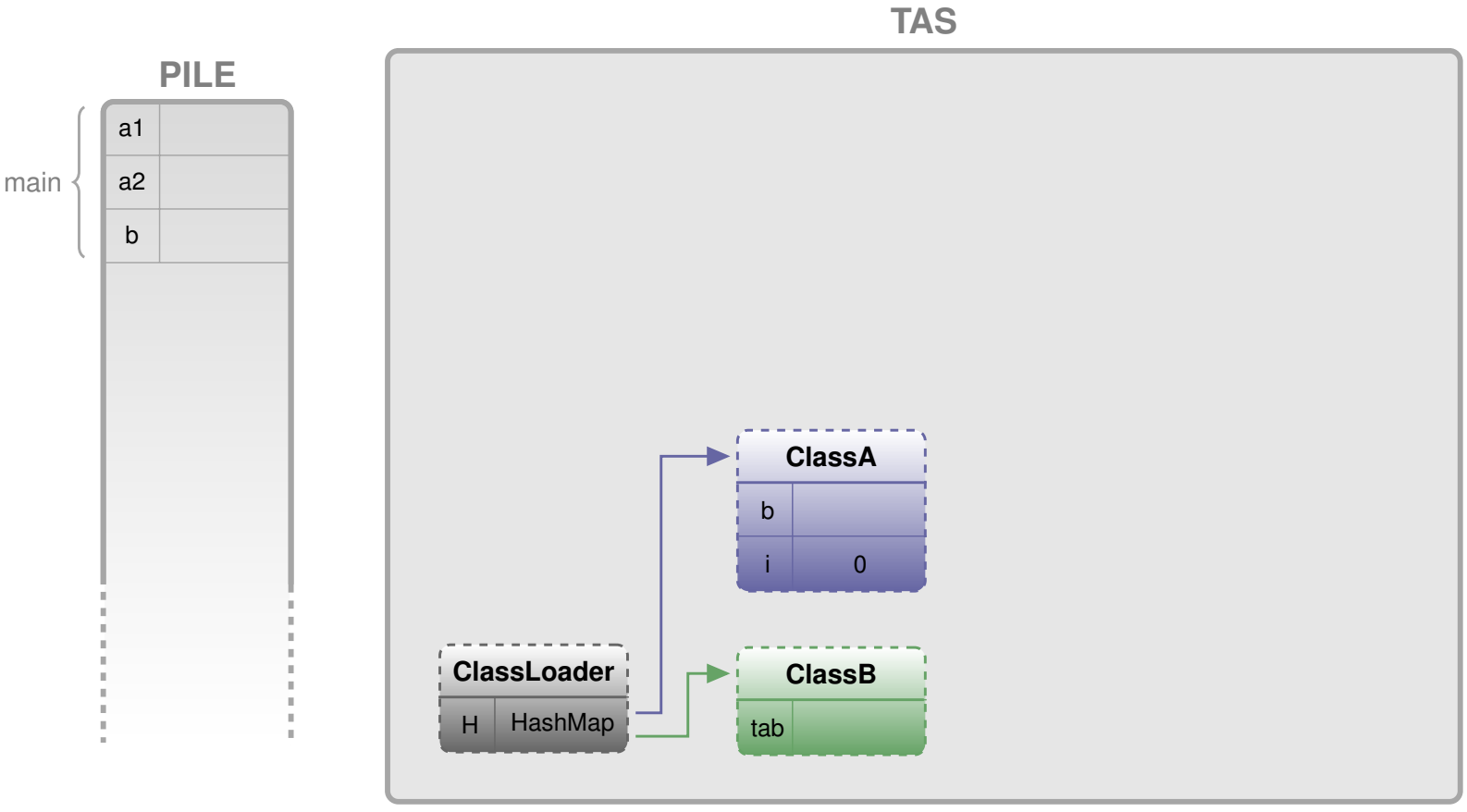
Méthodes et blocs static



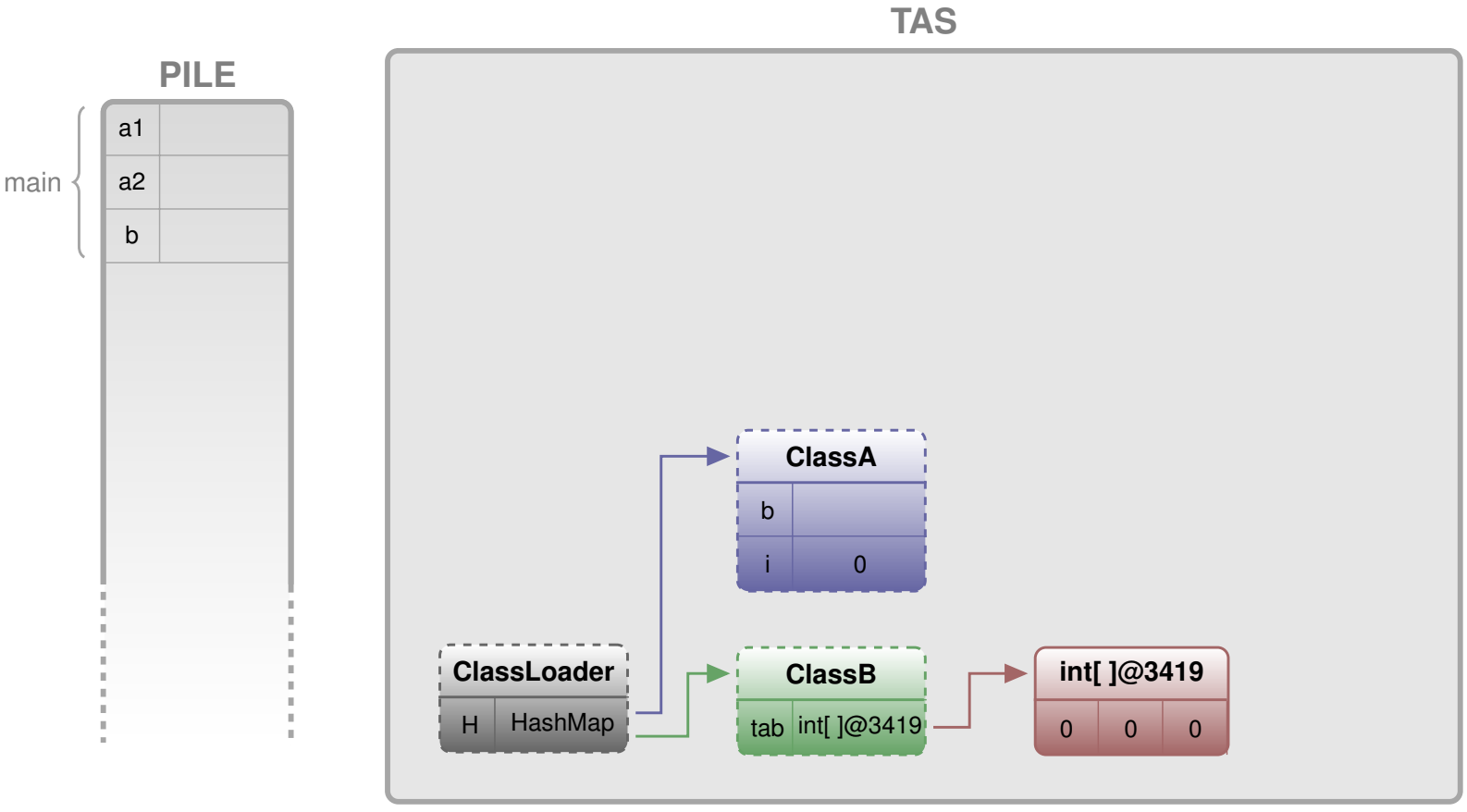
Méthodes et blocs static



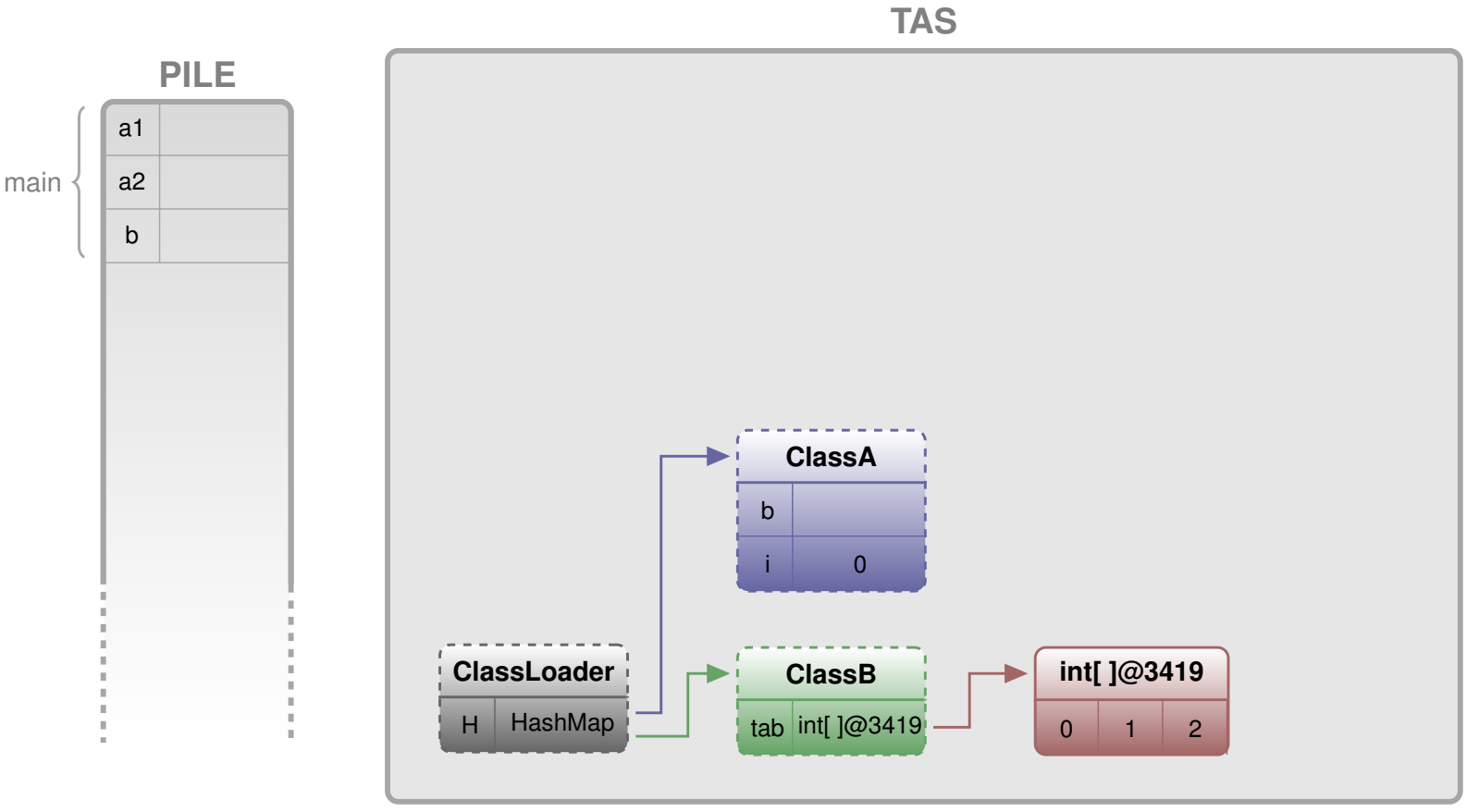
Méthodes et blocs static



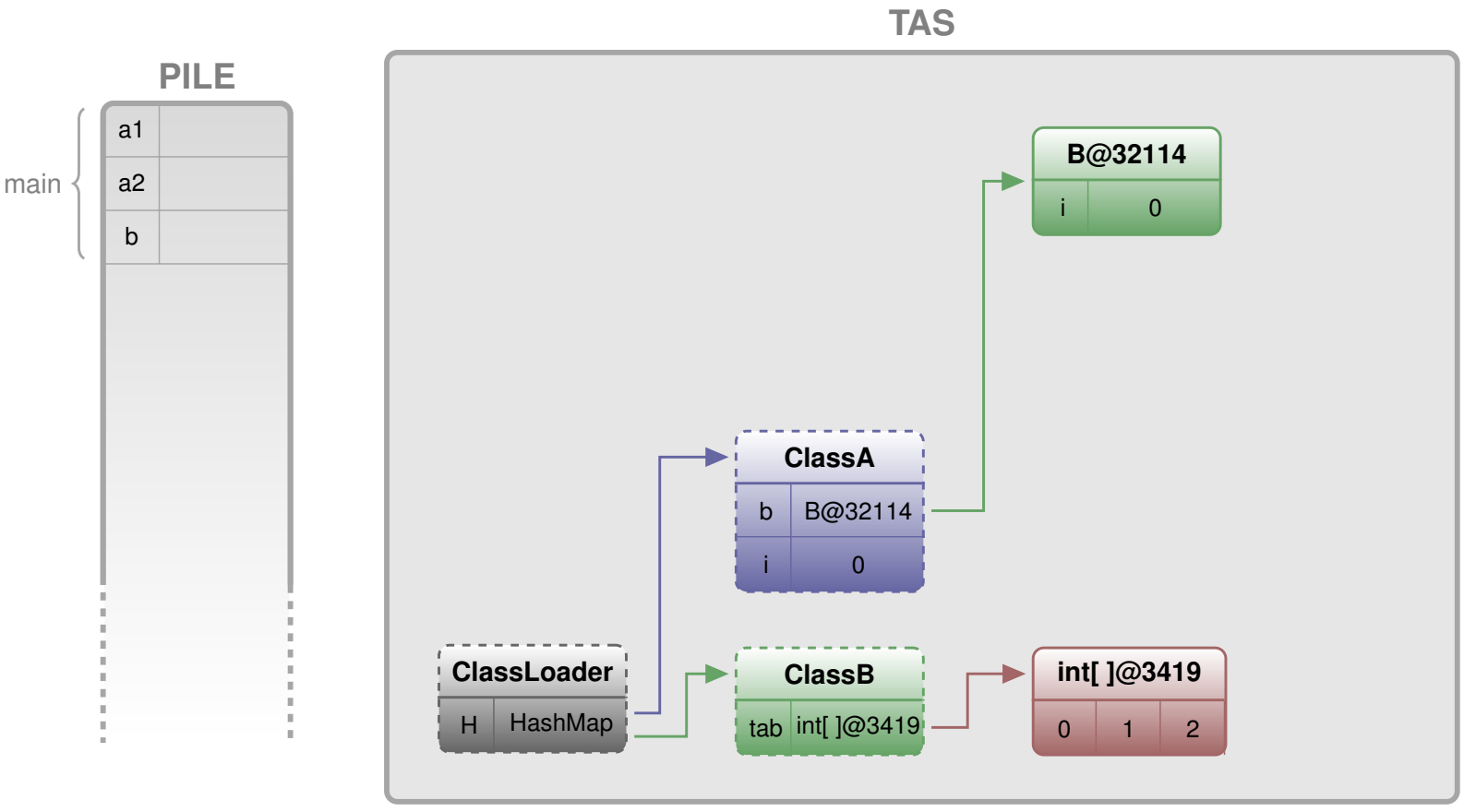
Méthodes et blocs static



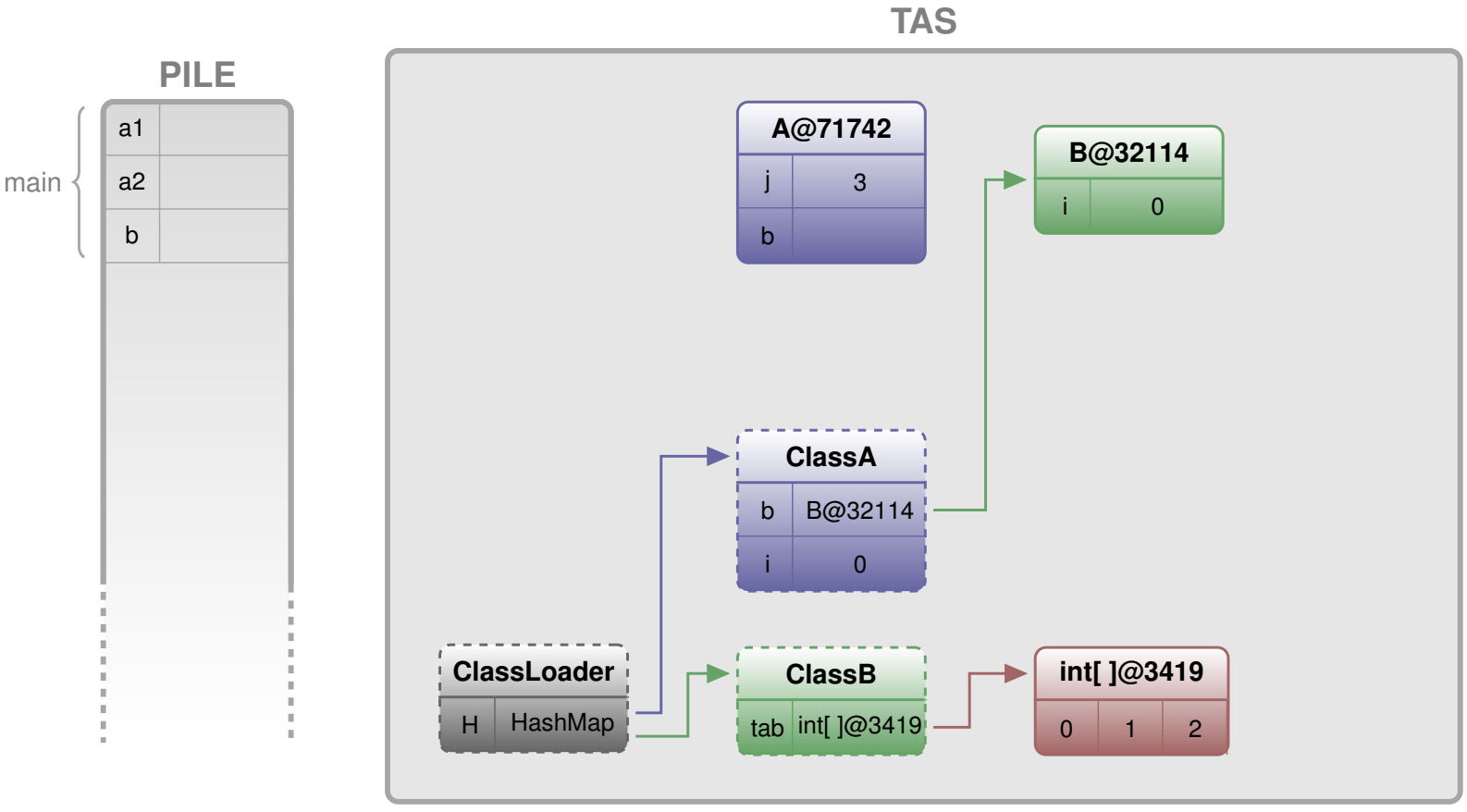
Méthodes et blocs static



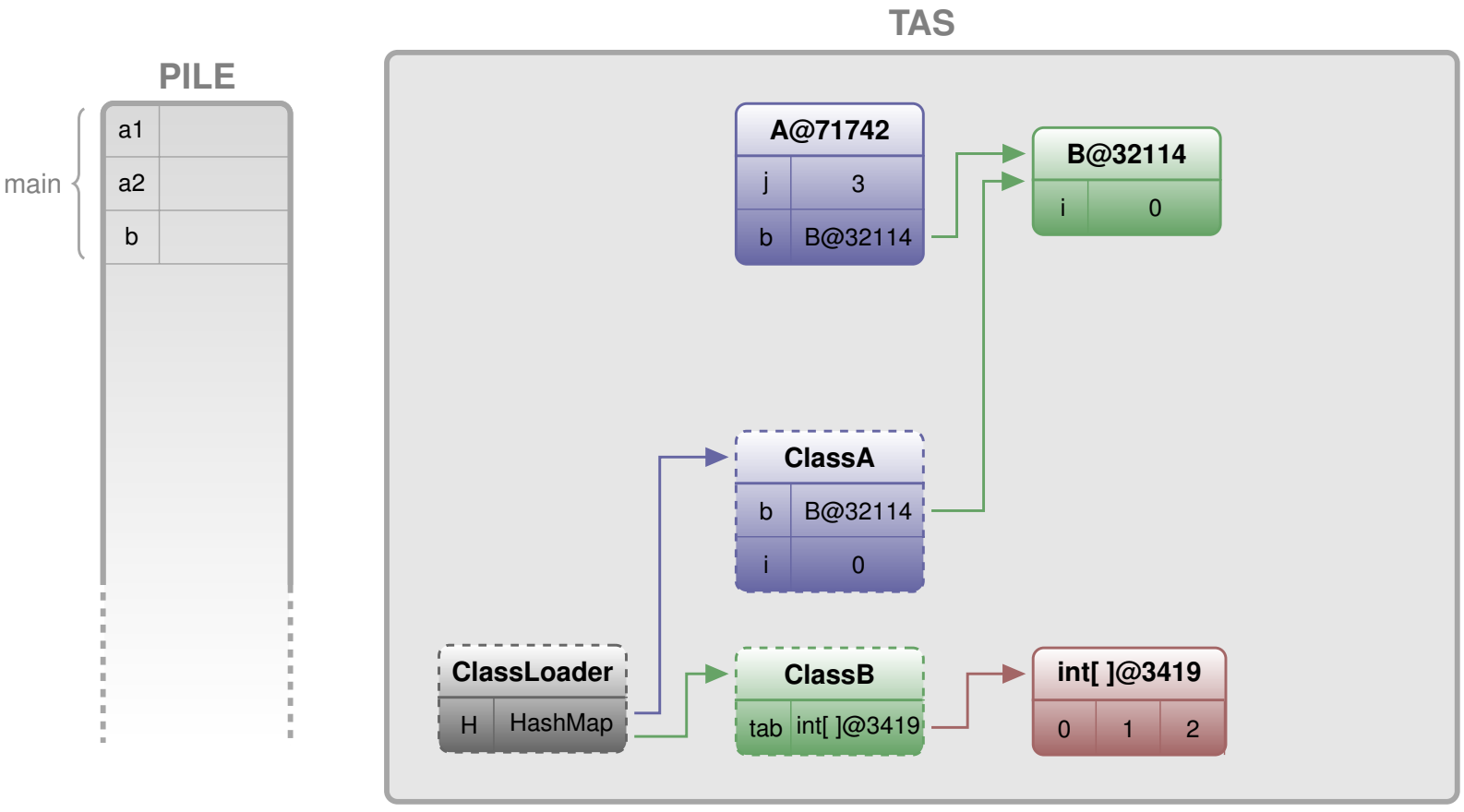
Méthodes et blocs static



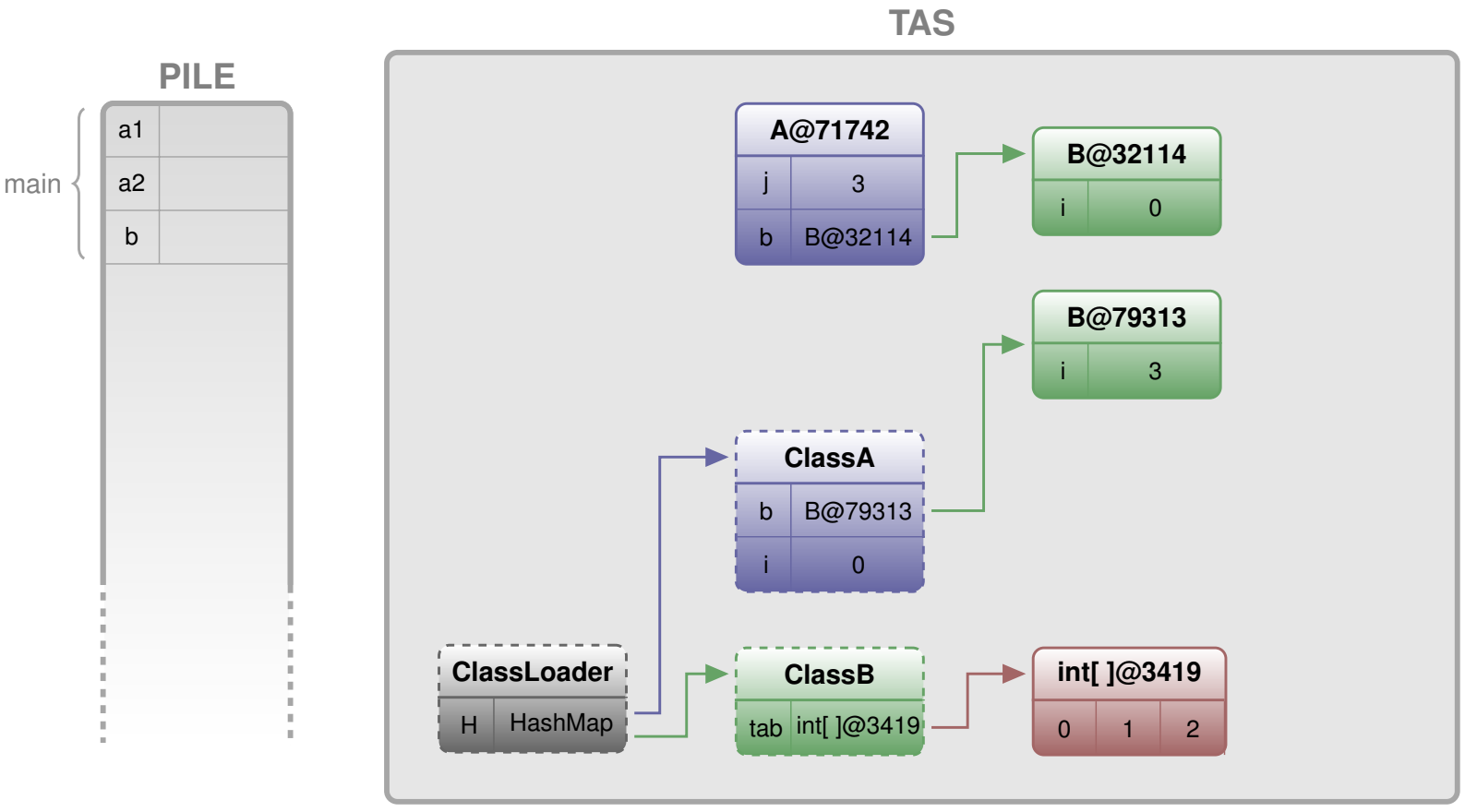
Méthodes et blocs static



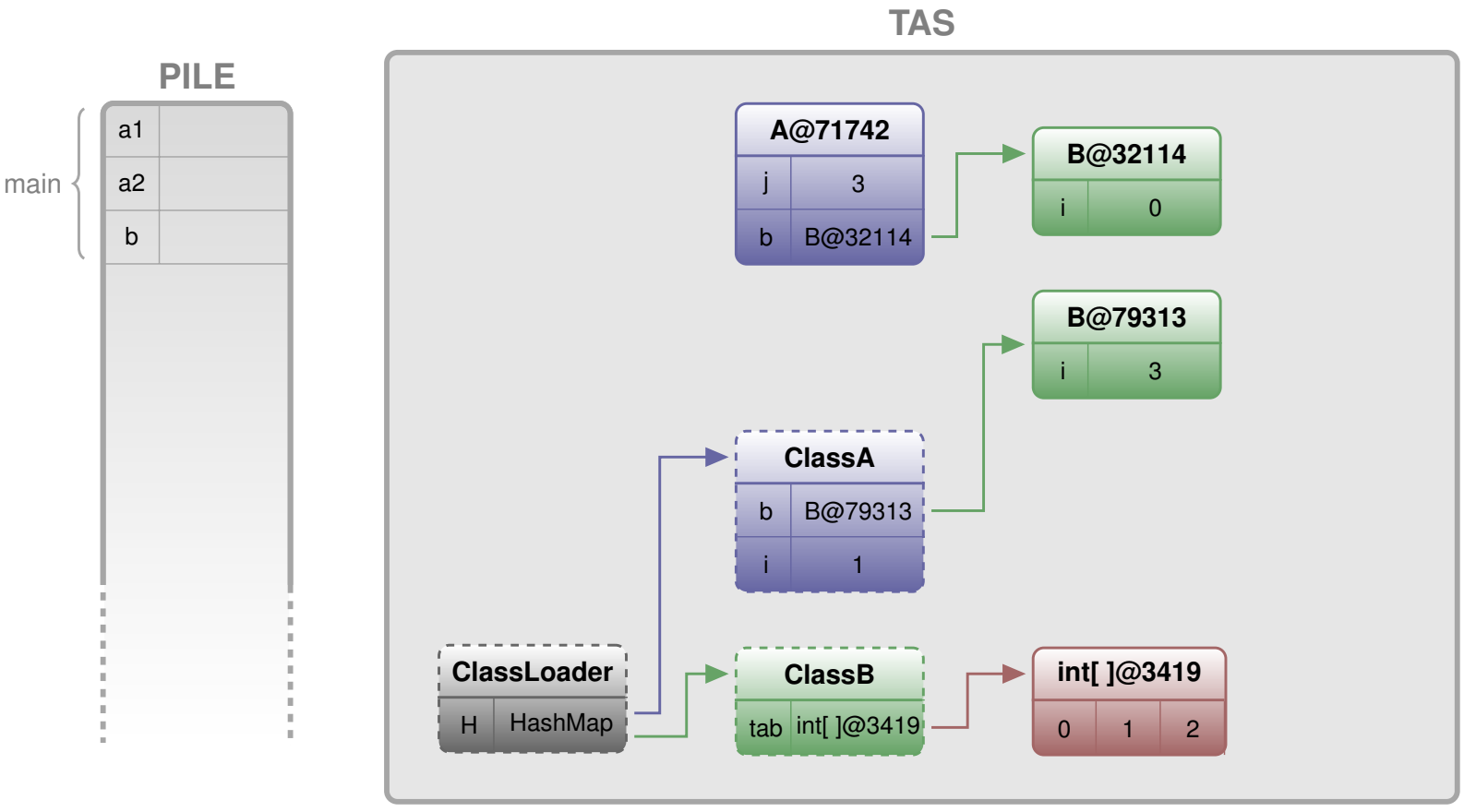
Méthodes et blocs static



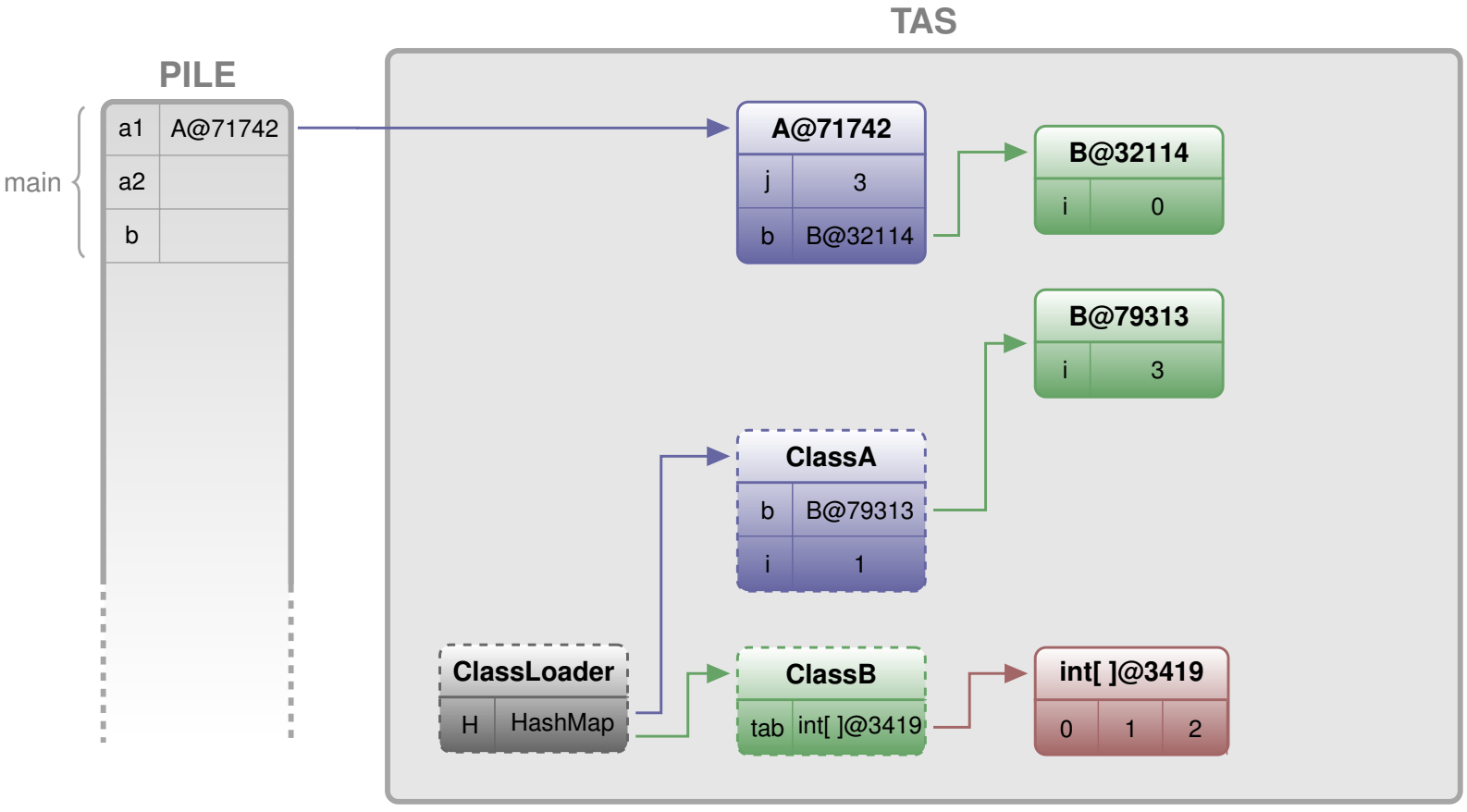
Méthodes et blocs static



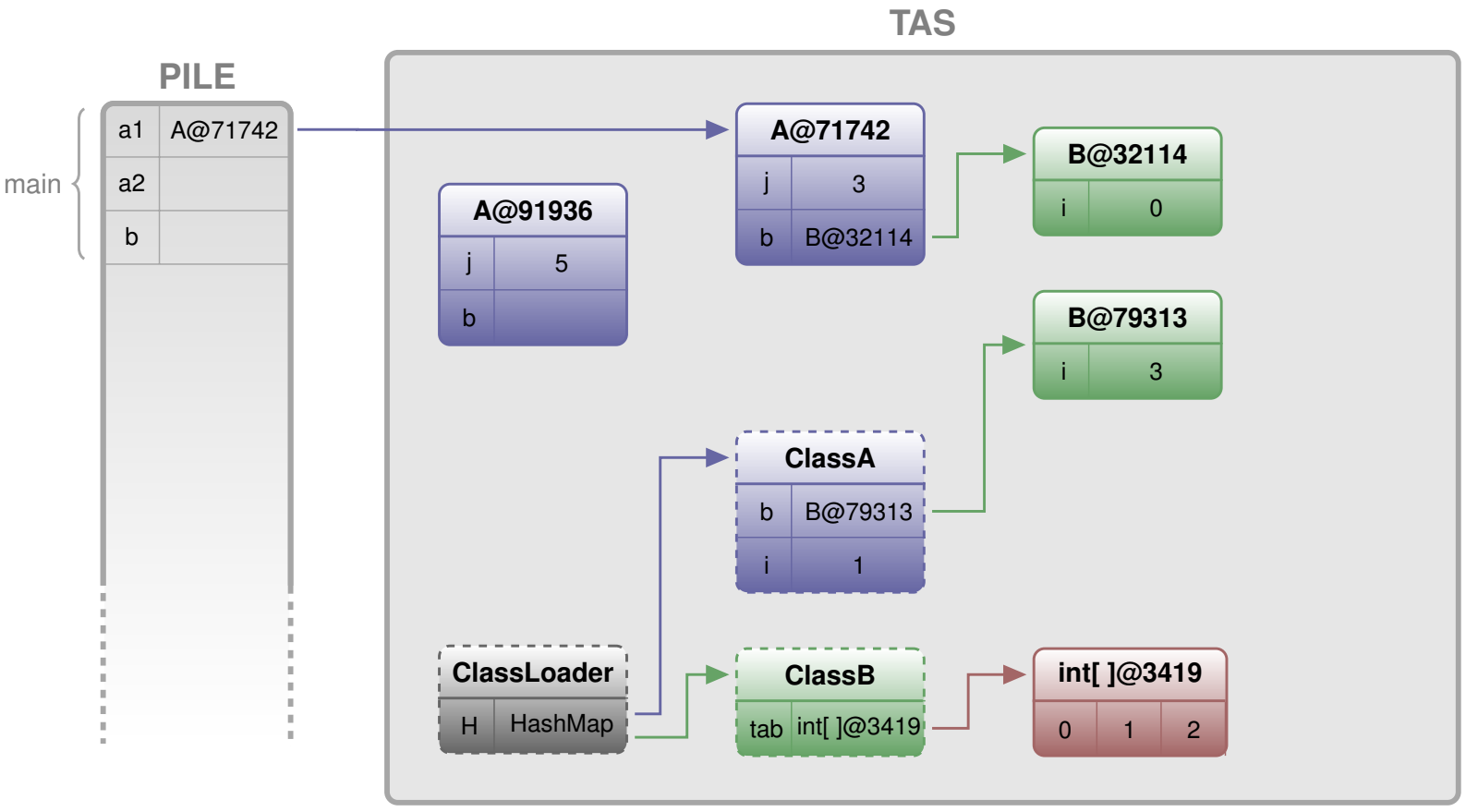
Méthodes et blocs static



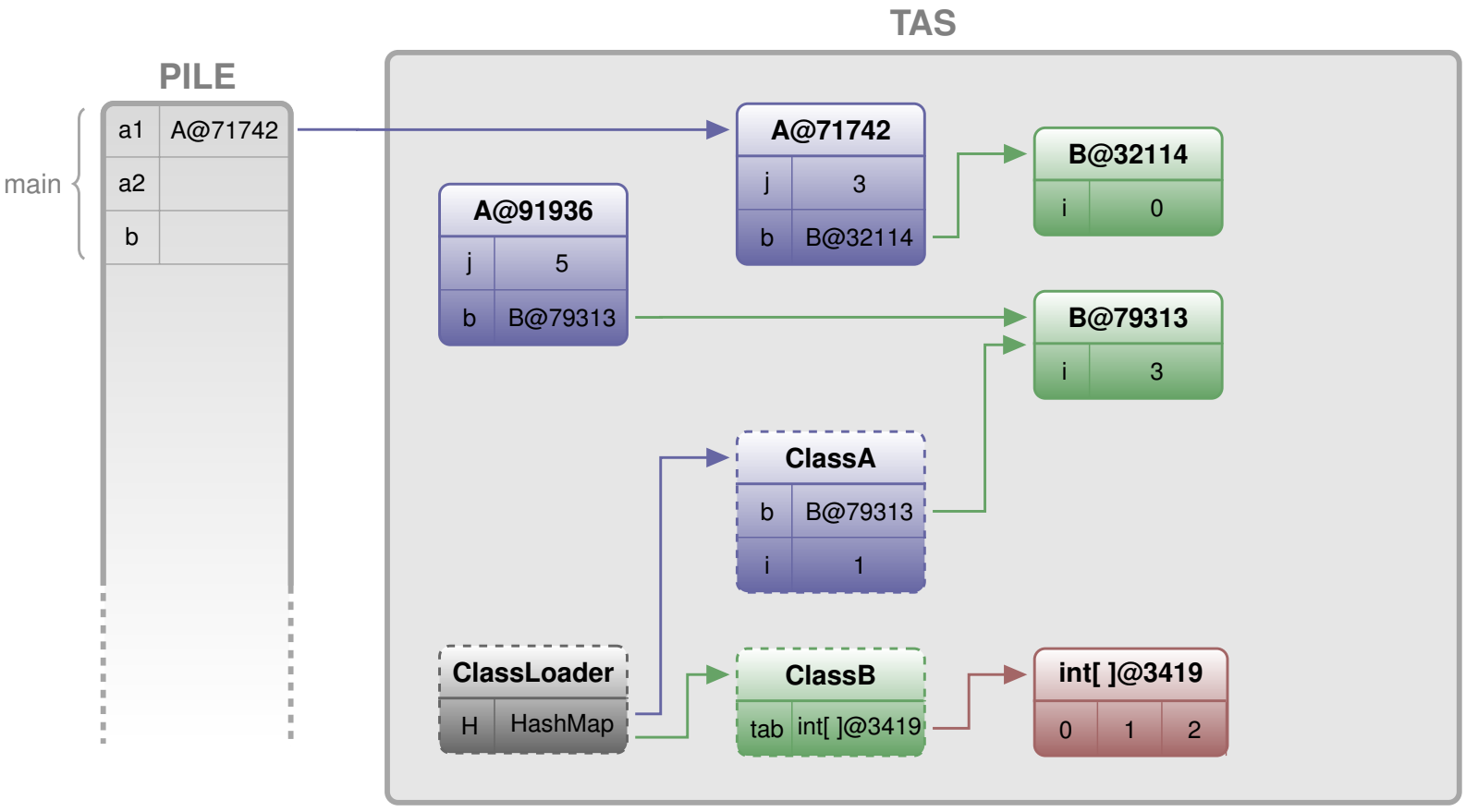
Méthodes et blocs static



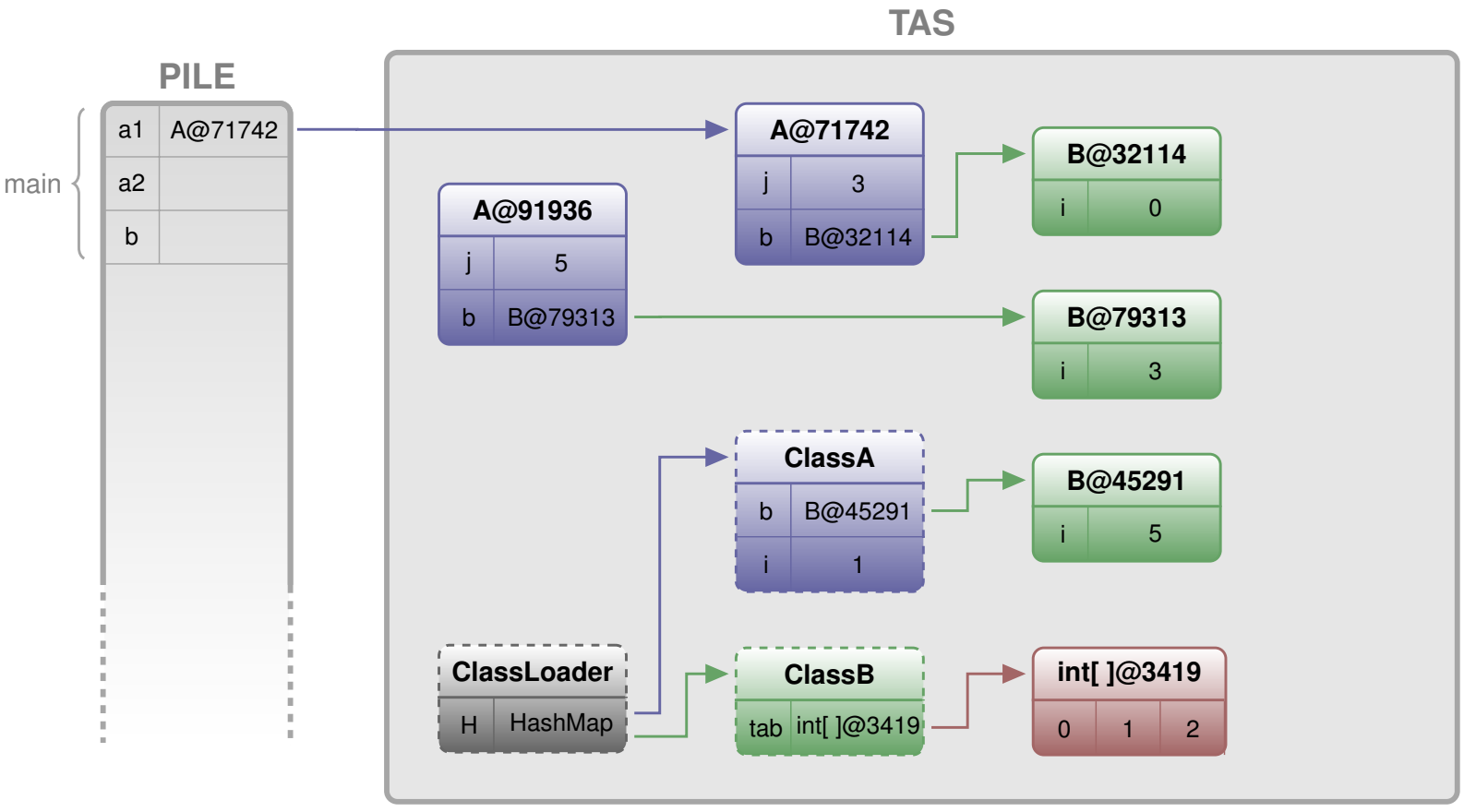
Méthodes et blocs static



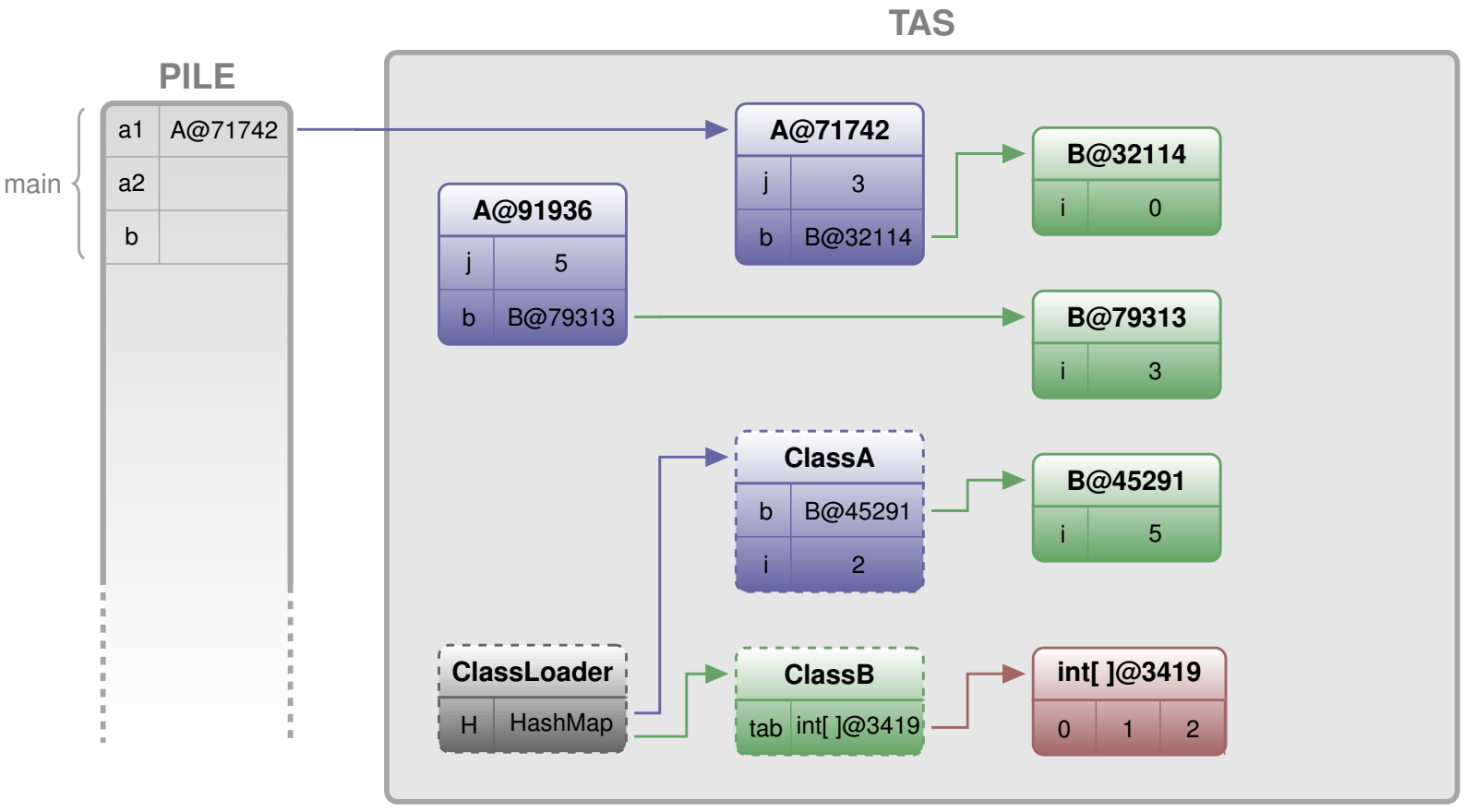
Méthodes et blocs static



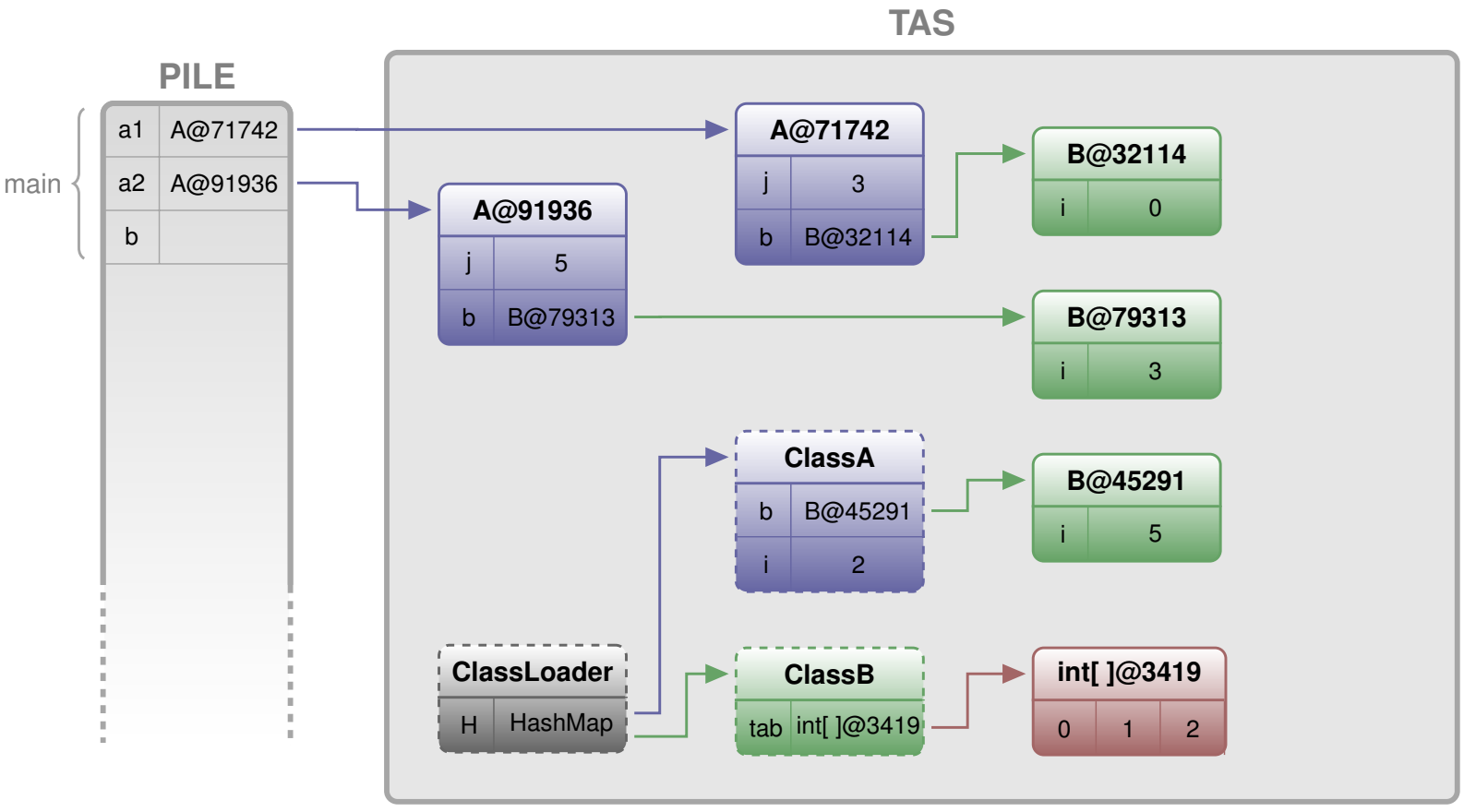
Méthodes et blocs static



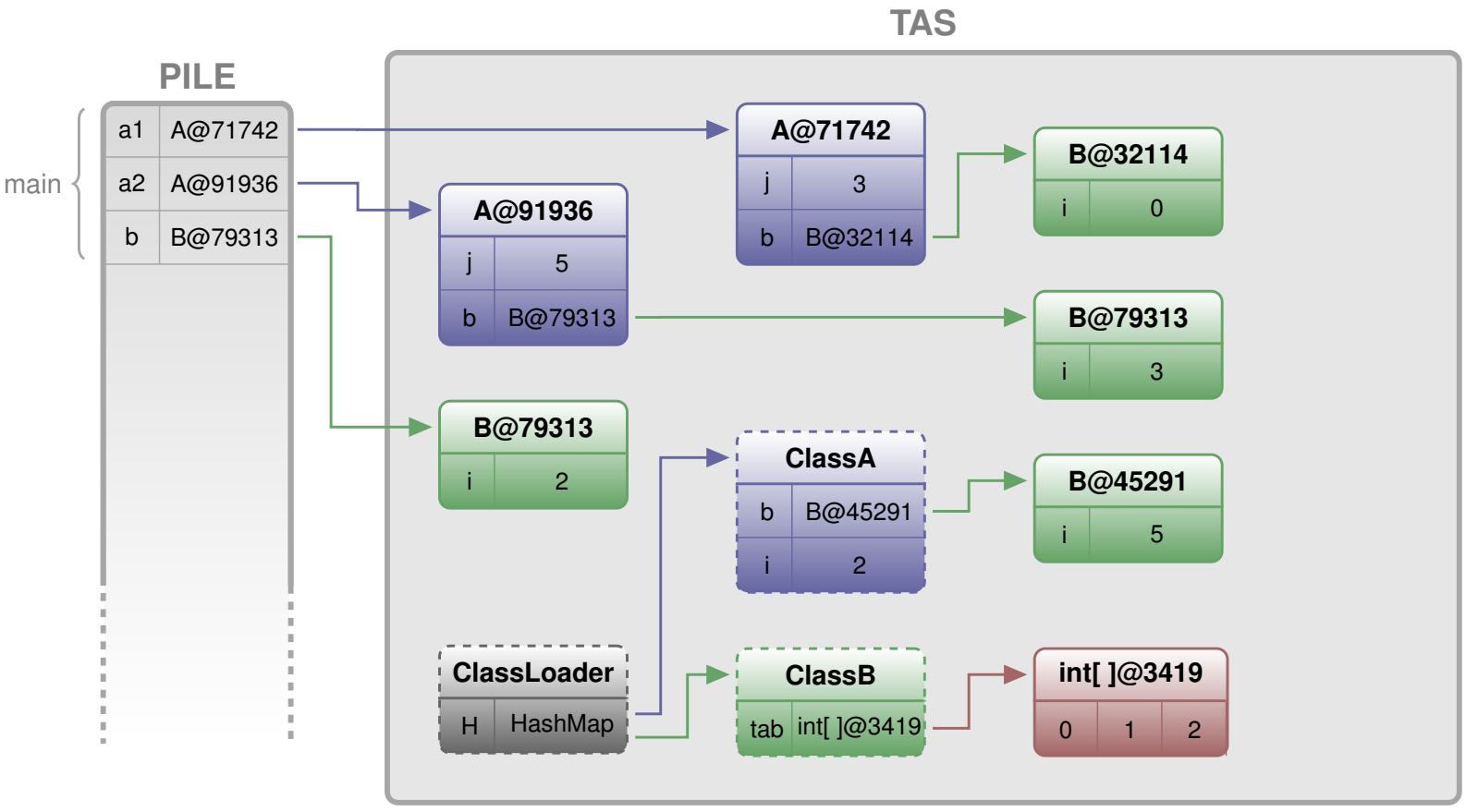
Méthodes et blocs static



Méthodes et blocs static



Méthodes et blocs static



Et dans la vrai vie ...

```java
public class Hello {
  public static void main (String[] args){
    System.out.println("Hello world !");
  }
}
```

```
java -verbose:class Hello
```

Et dans la vrai vie ...

```java
public class Hello {
  public static void main (String[] args){
    System.out.println("Hello world !");
  }
}
```

```
java -verbose:class Hello
    [Opened /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Object from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.io.Serializable from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Comparable from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.String from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Class from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Cloneable from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.ClassLoader from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.System from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Throwable from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Error from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.ThreadDeath from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Exception from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.RuntimeException from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    ... ≈ 400 Lignes ⟹ ≈ 400 Classes
    [Loaded java.security.AllPermission from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.security.UnresolvedPermission from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.security.BasicPermissionCollection from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded Hello from file:/tmp/]
    [Loaded java.lang.Void from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    Hello world !
    [Loaded java.lang.Shutdown from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Shutdown$Lock from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
```

Et dans la vrai vie ...

```java
public class Hello {
  public static void main (String[] args){
    System.out.println("Hello world !");
  }
}
```

```
java -verbose:class Hello
    [Opened /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Object from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.io.Serializable from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Comparable from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.String from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Class from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Cloneable from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.ClassLoader from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.System from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Throwable from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Error from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.ThreadDeath from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Exception from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.RuntimeException from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    ... ≈ 400 Lignes ⟹ ≈ 400 Classes
    [Loaded java.security.AllPermission from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.security.UnresolvedPermission from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.security.BasicPermissionCollection from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded Hello from file:/tmp/]
    [Loaded java.lang.Void from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    Hello world !
    [Loaded java.lang.Shutdown from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
    [Loaded java.lang.Shutdown$Lock from /usr/lib/jvm/java-7-openjdk/jre/lib/rt.jar]
```

Outline

L'héritage

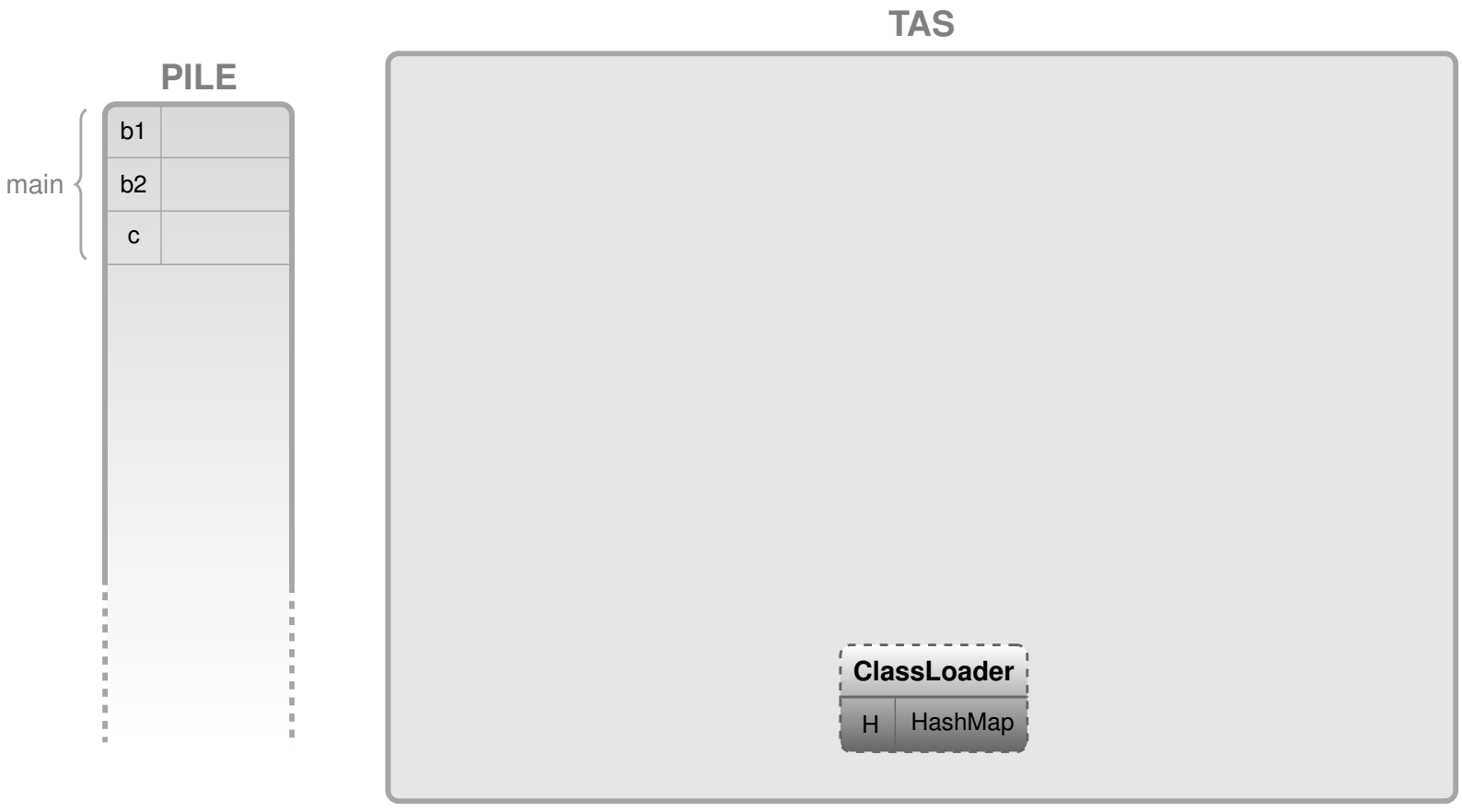
```
public class A {
  static int x;
  int i;
  private int p;
  public A(int i) {
    this.i = i ;
    p=1 ;
  }
  public void f () {
    System.out.print("i="+i+" ");
  }
}
```

```
public class B extends A {
  static int y=10;
  int j;
  public B(){
    super(y);
    y++;
    x++;
    j=x+y;
  }
  public void f () {
    System.out.print("j="+j+" ");
  }
}
```

```
public class C extends B {
  static int x=20;
  int k;
  public C(){
    y++;
    x++;
    k=x+y;
  }
  public void f () {
    System.out.print("k="+k+" ");
  }
}
```

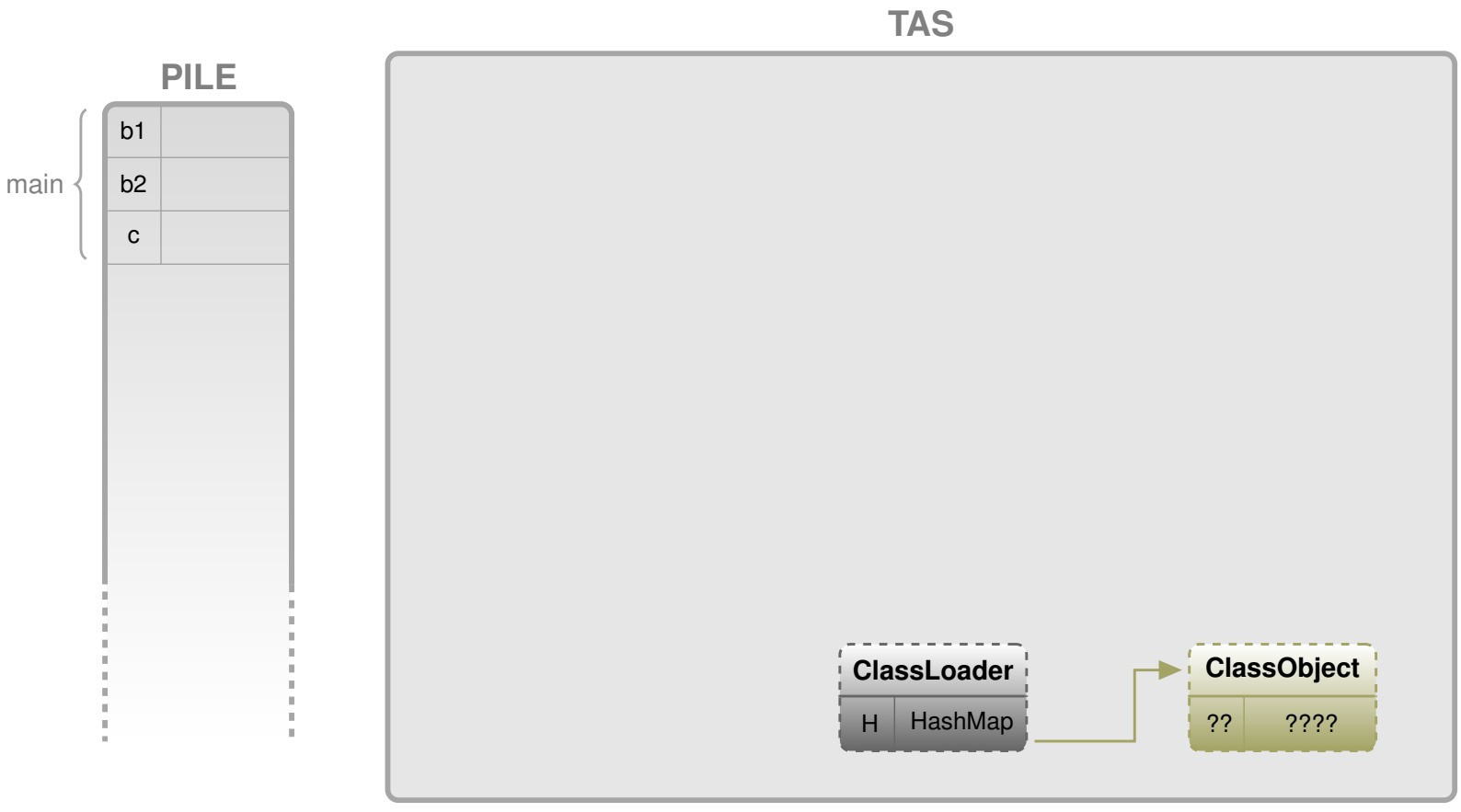
```
public class Main {
  public static B f () {
    C cLoc = new C() ;
    return cLoc ;
  }
  public static void main(String[] args) {
    B b1 = new B();
    B b2 = f() ;
    C c = (C) b2 ;
    b1.f() ;
    b2.f() ;
    c.f() ;
  }
}
```

L'héritage



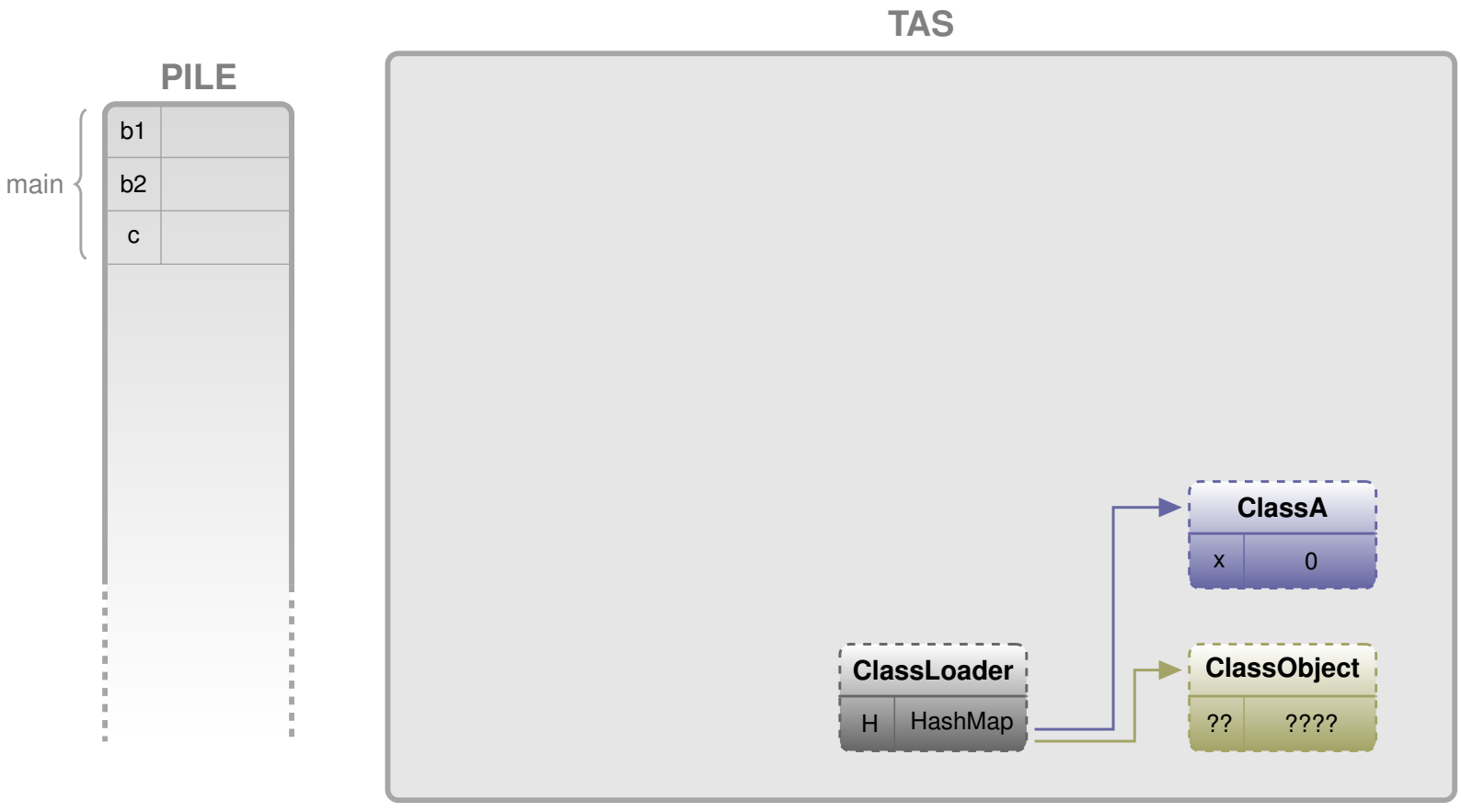

```
java Main
```

L'héritage



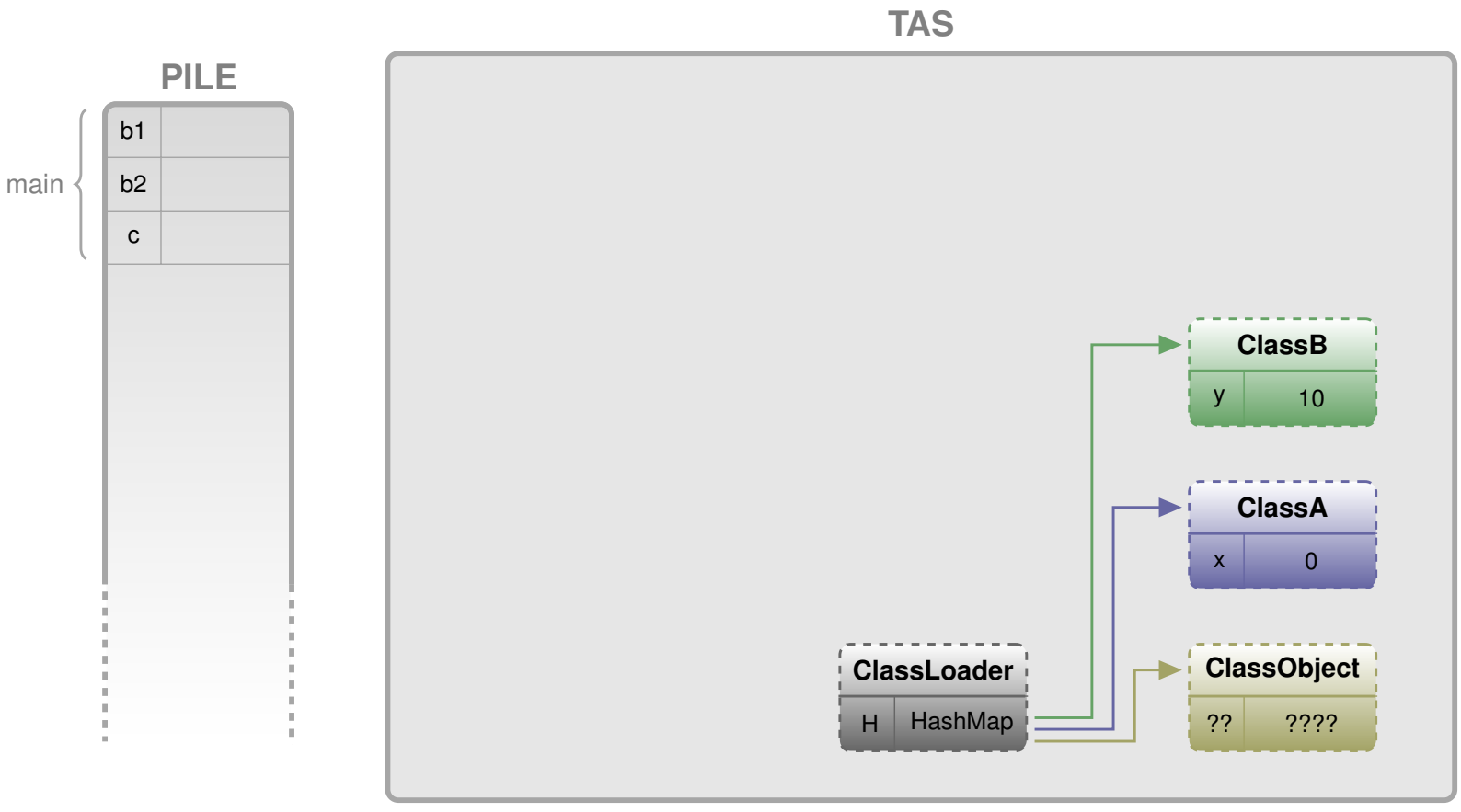

```
java Main
```

L'héritage



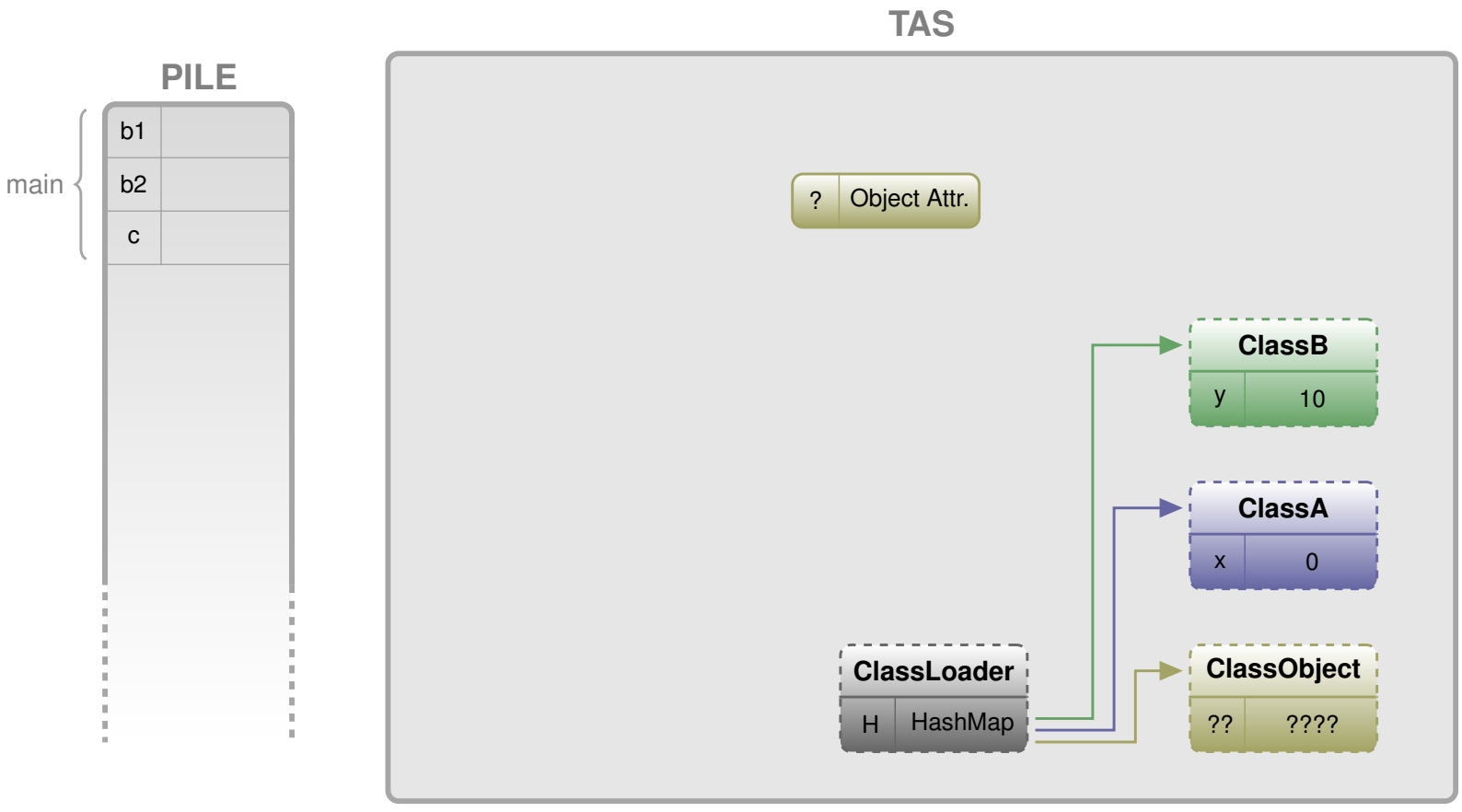

```
java Main
```

L'héritage



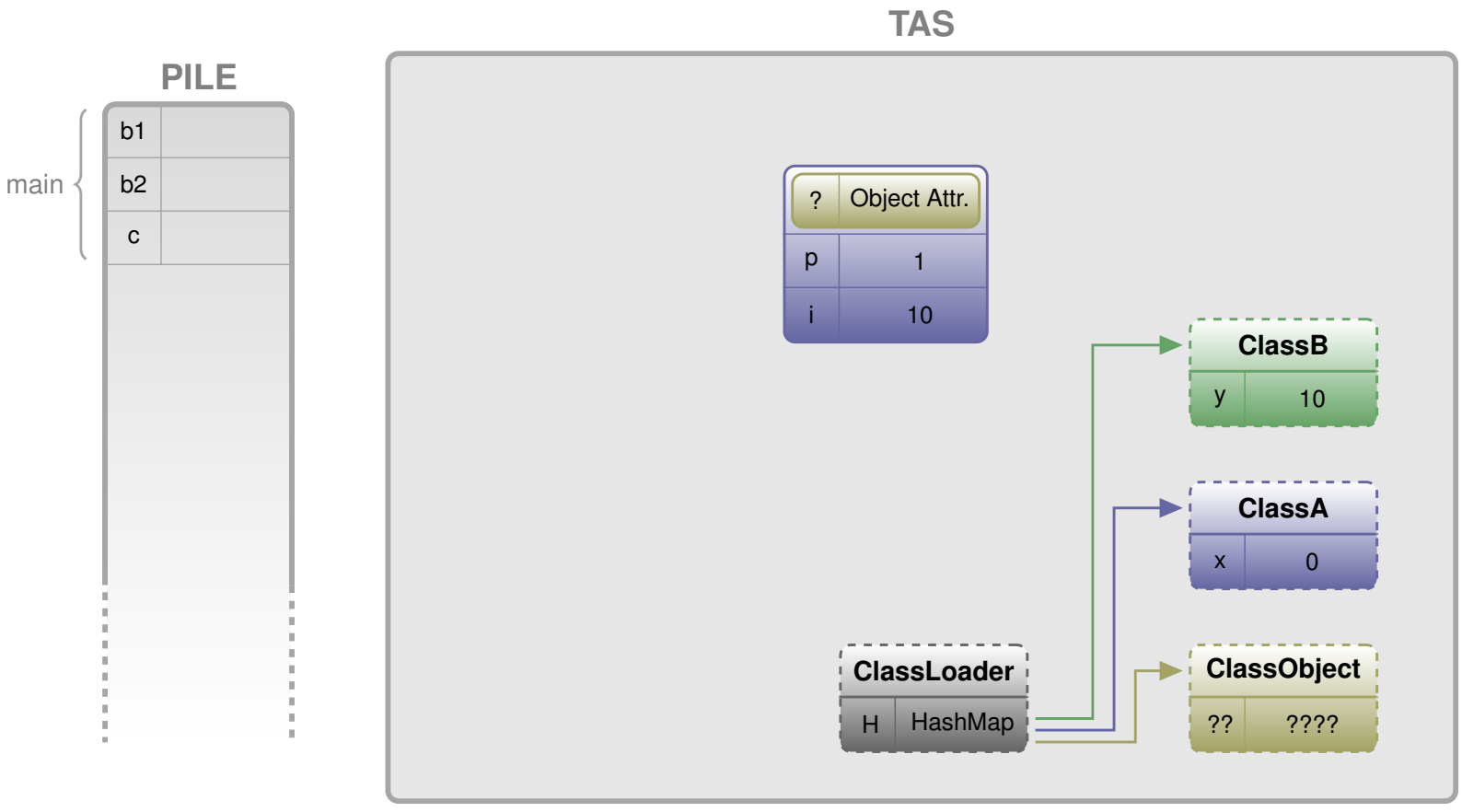

```
java Main
```

L'héritage



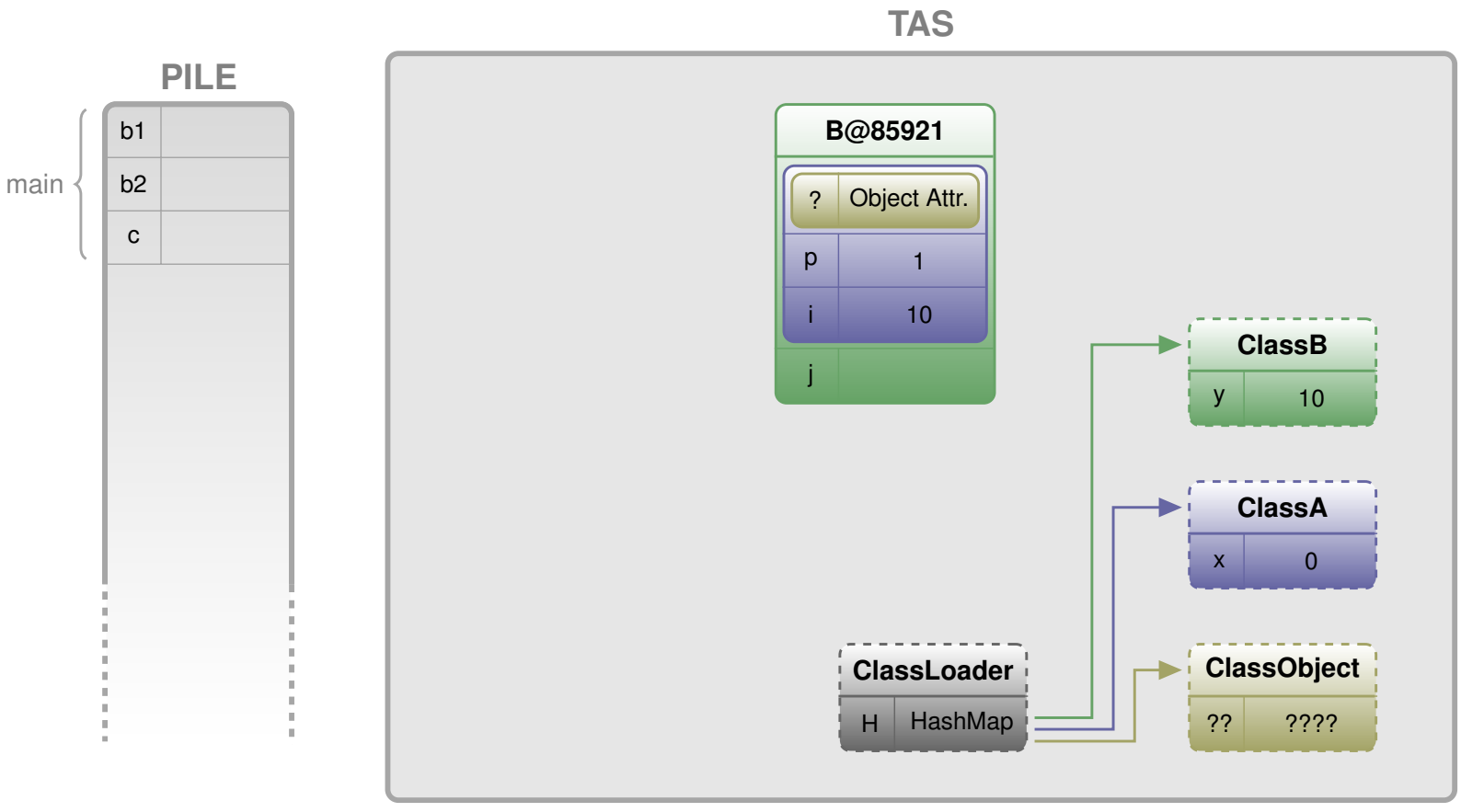

```
java Main
```

L'héritage



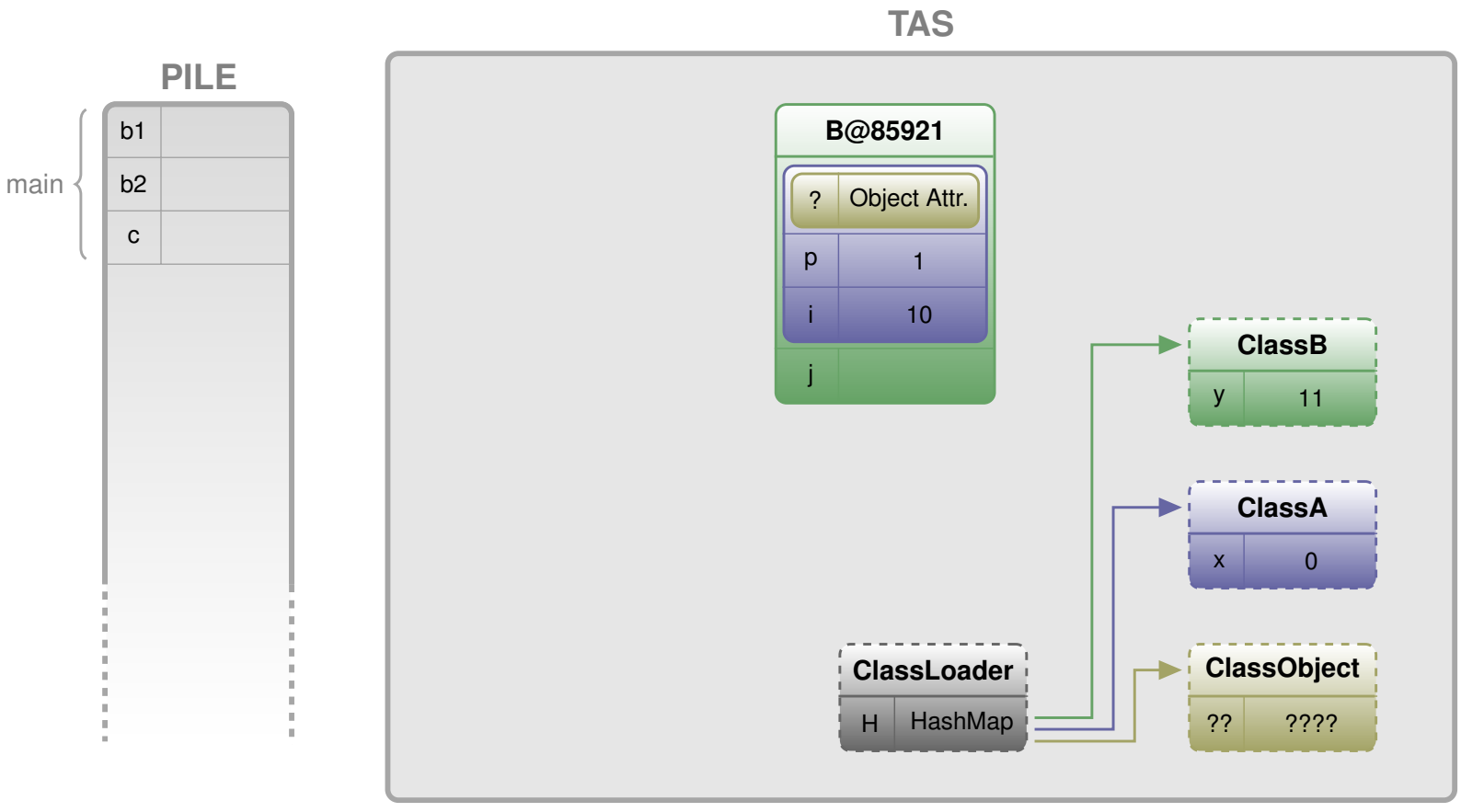

```
java Main
```

L'héritage



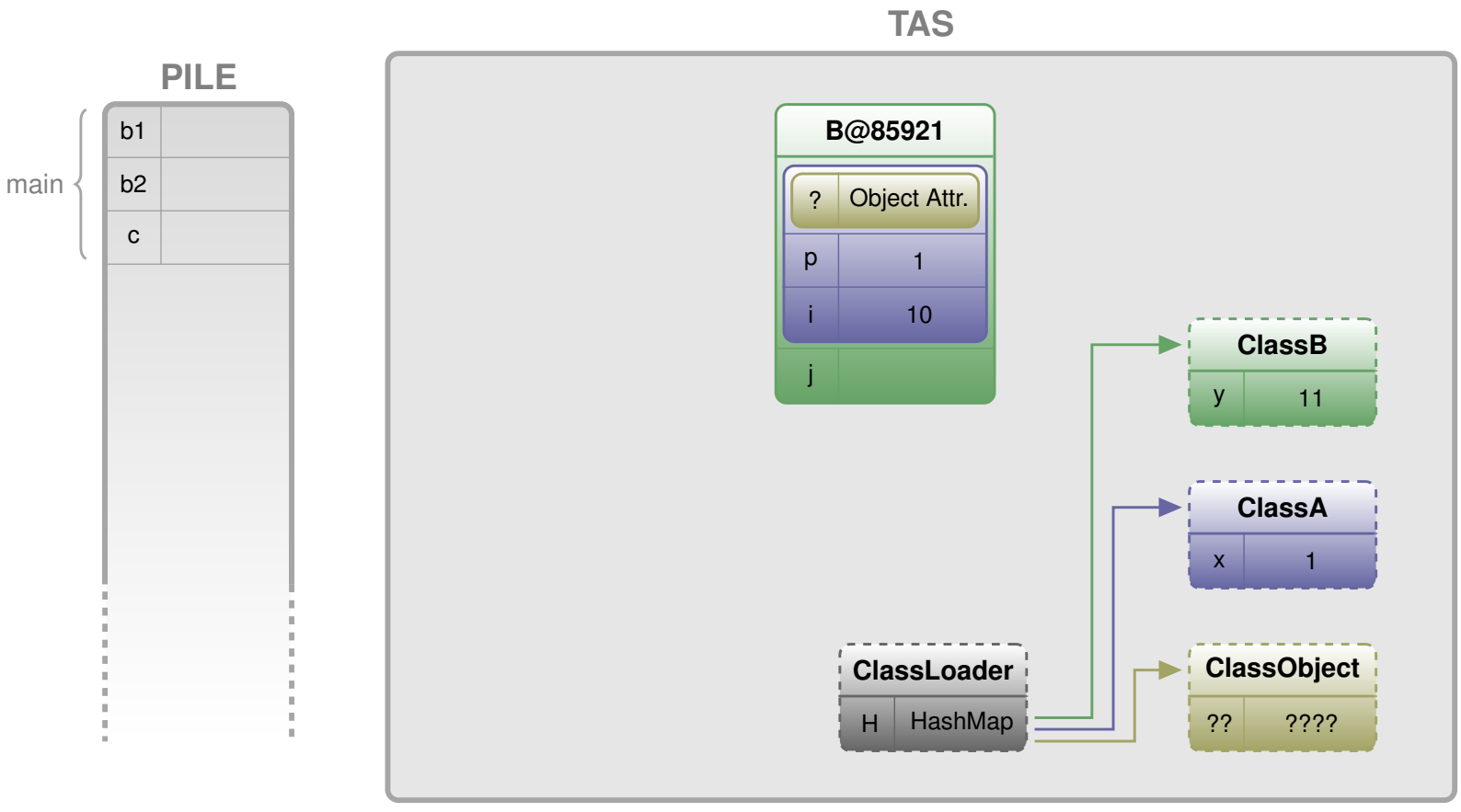

```
java Main
```

L'héritage



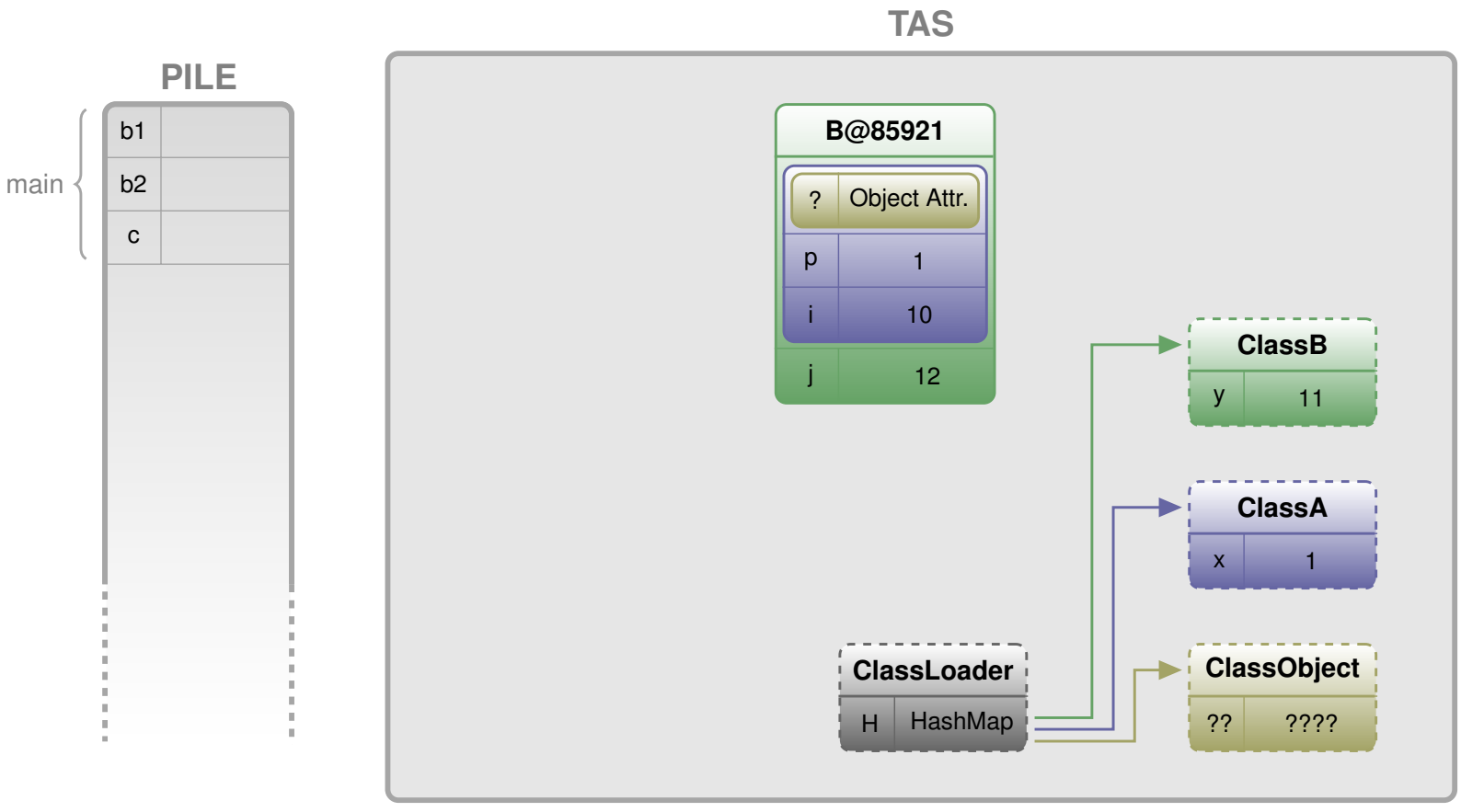

```
java Main
```

L'héritage



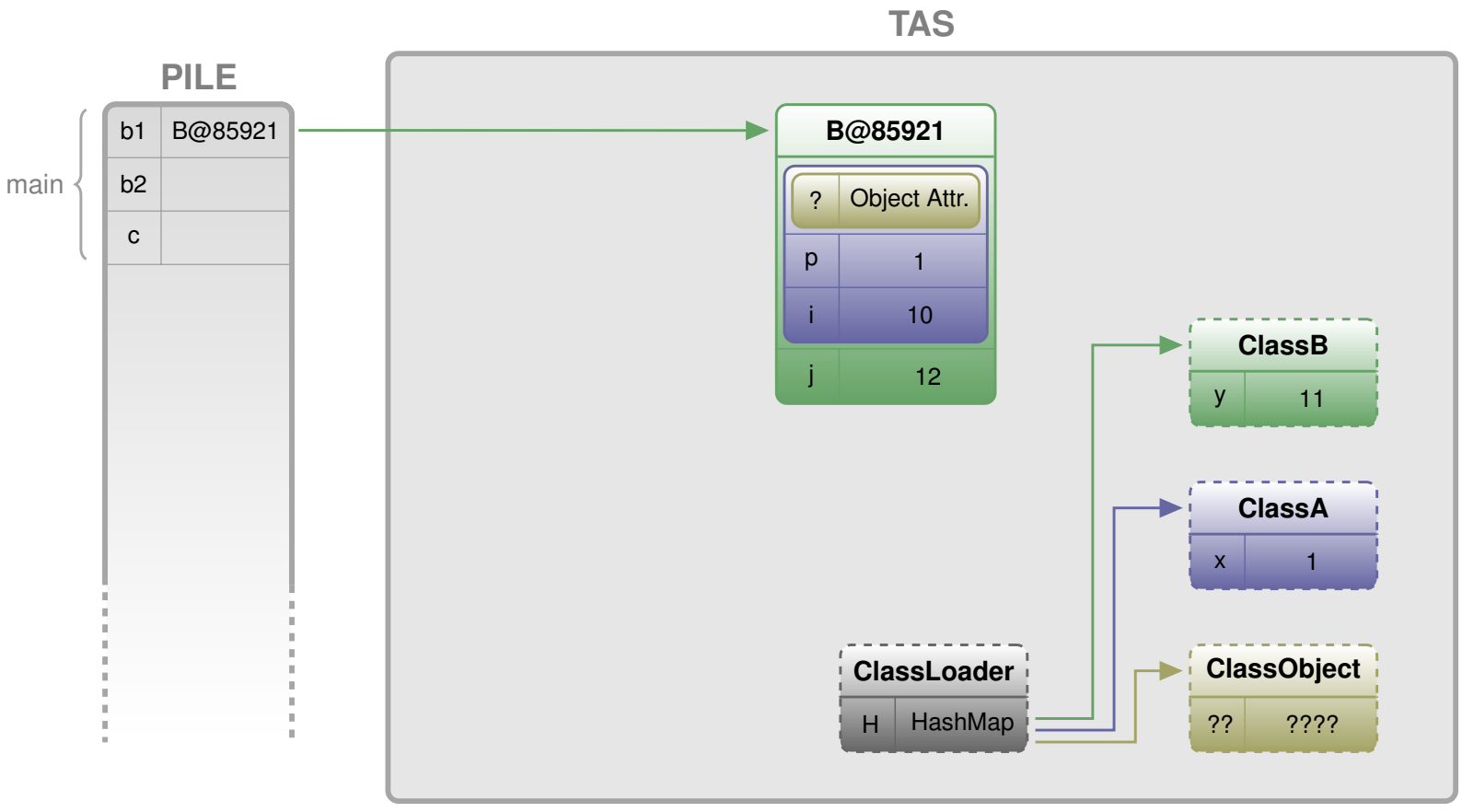

```
java Main
```

L'héritage



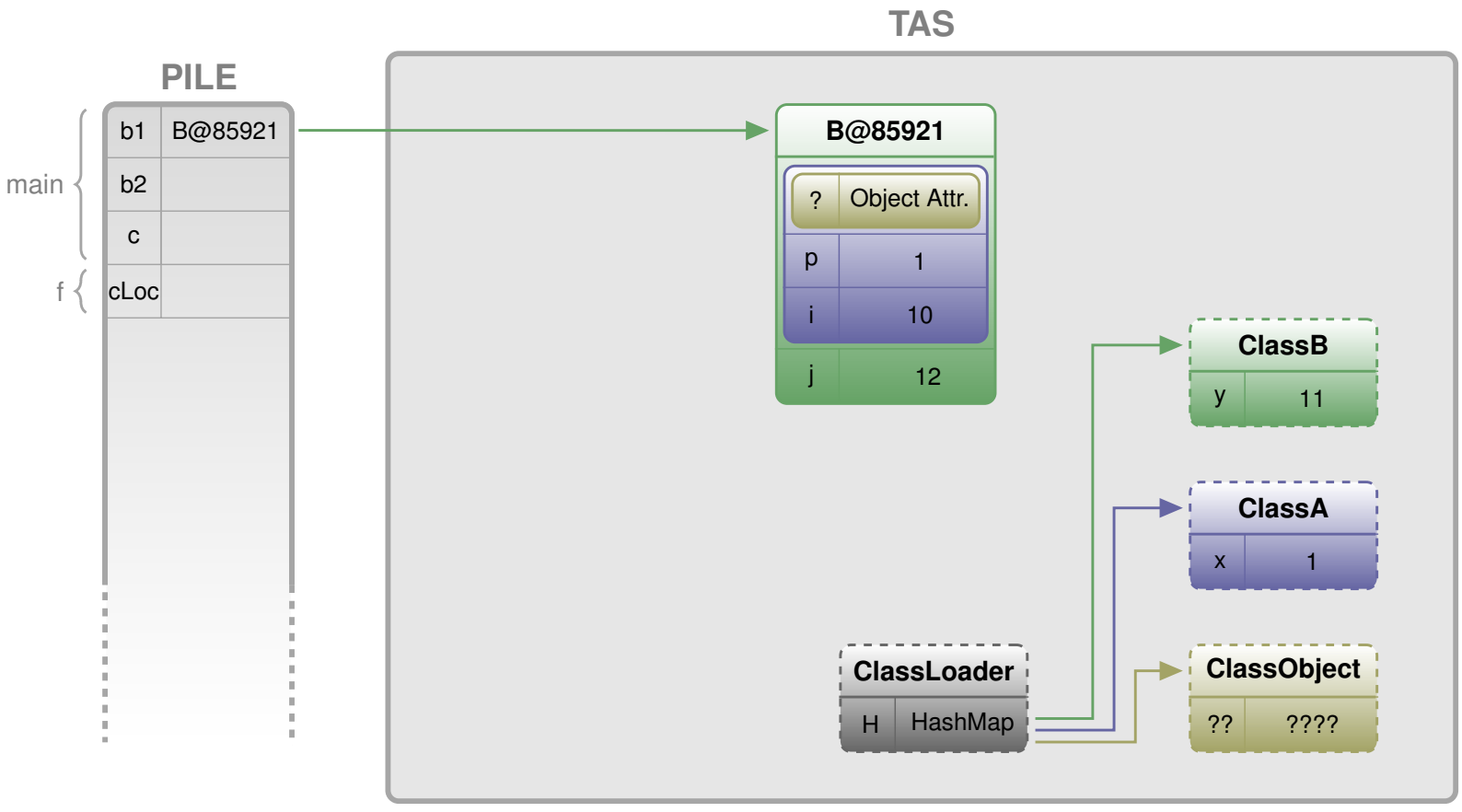

```
java Main
```

L'héritage



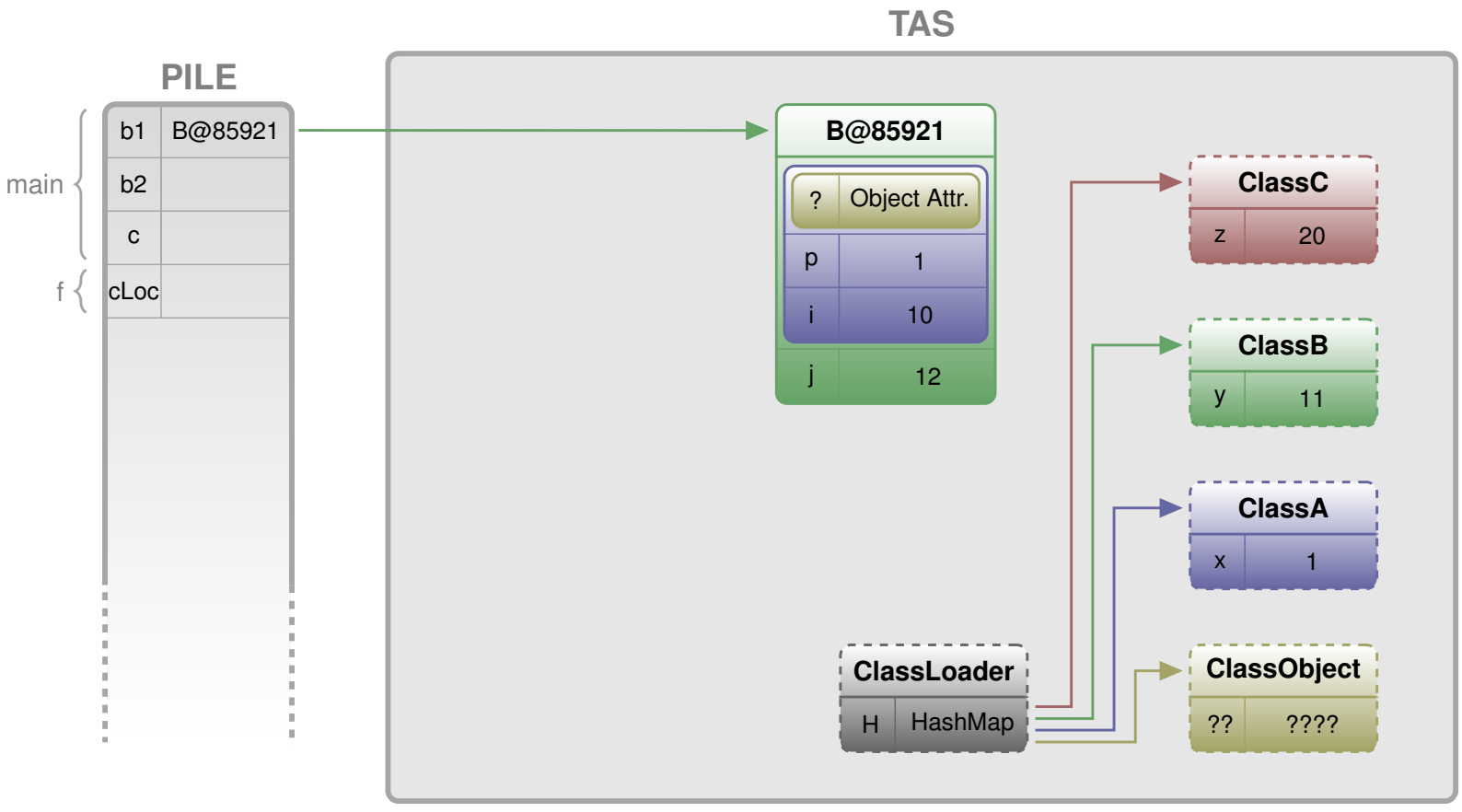

```
java Main
```

L'héritage



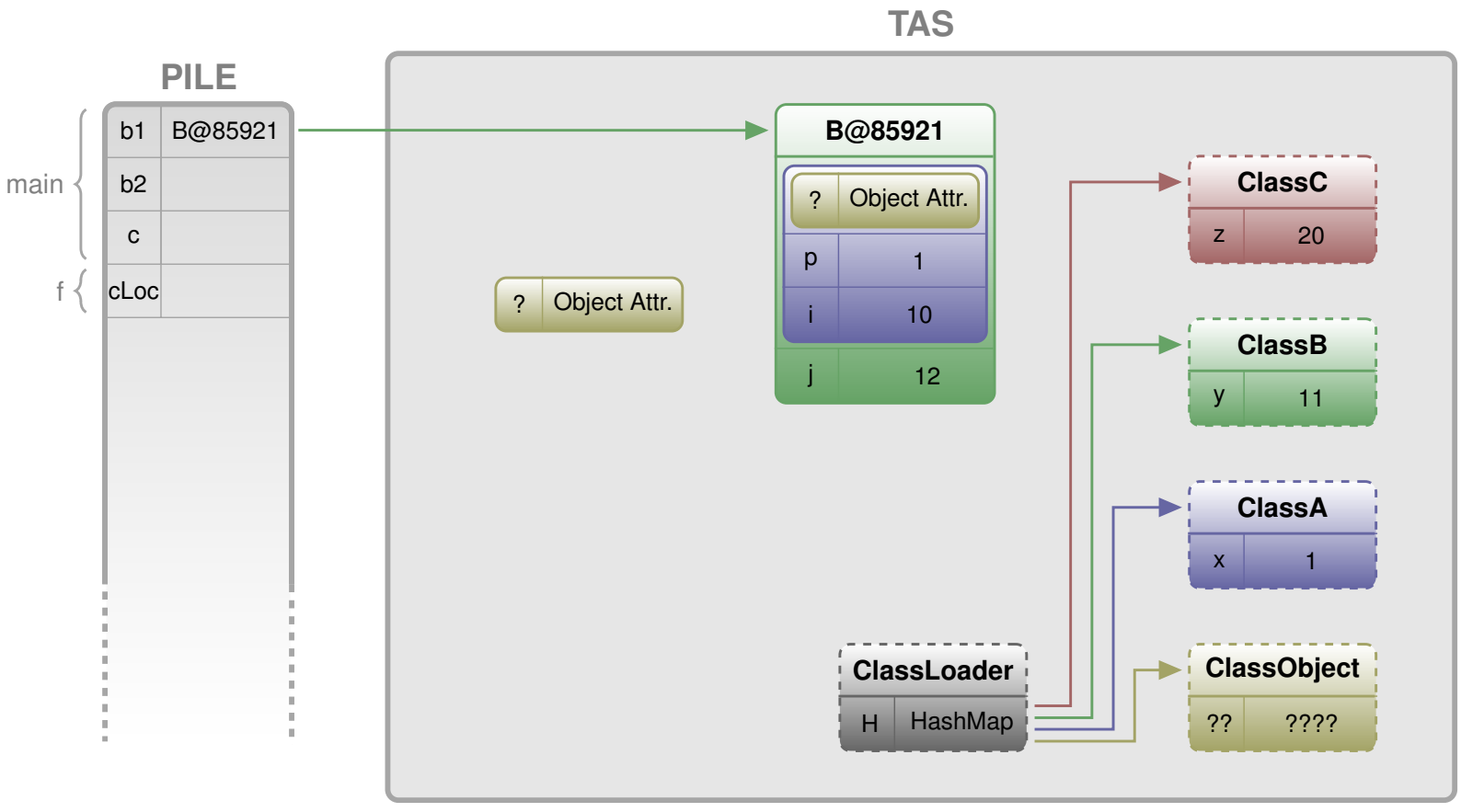

```
java Main
```

L'héritage



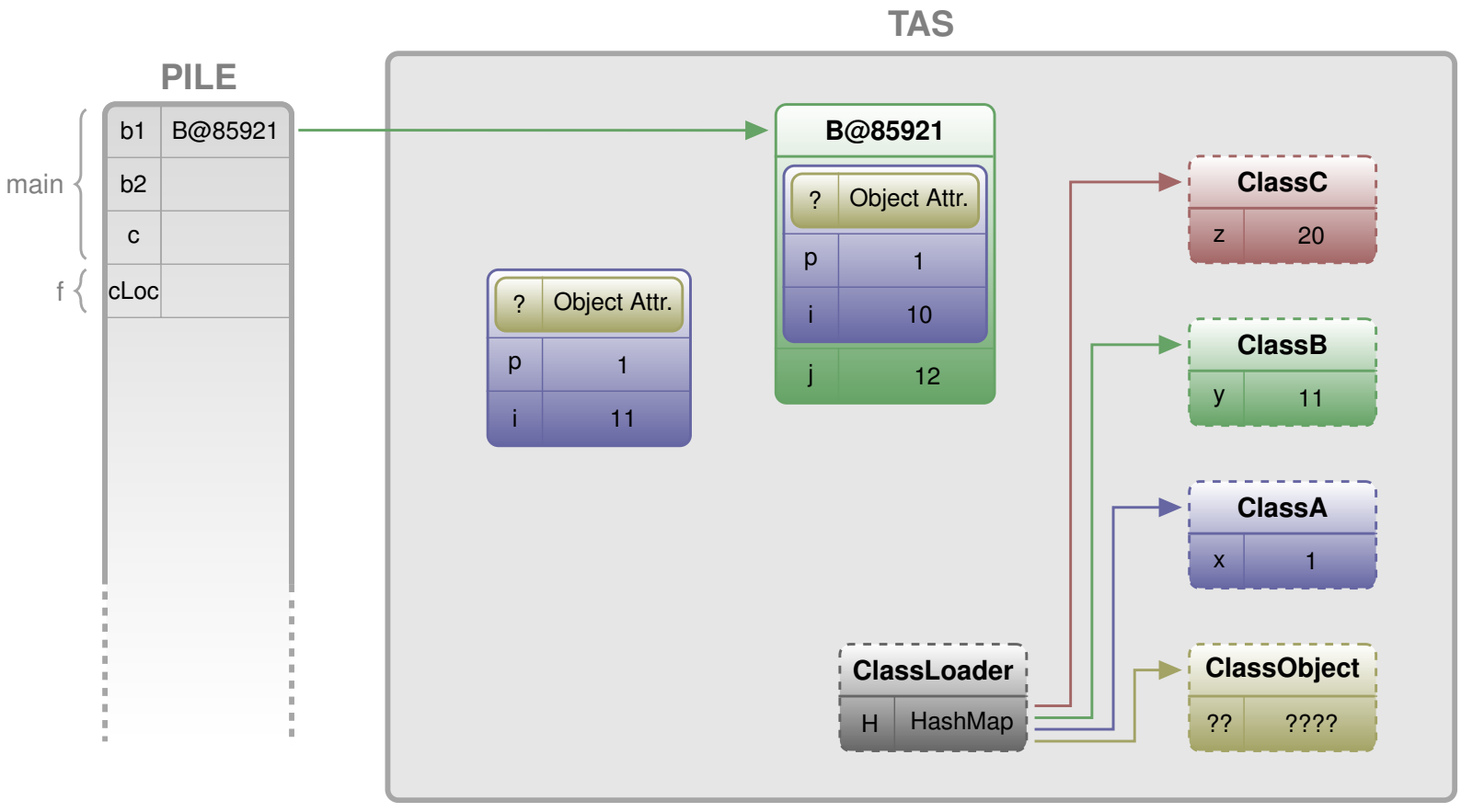

```
java Main
```

L'héritage



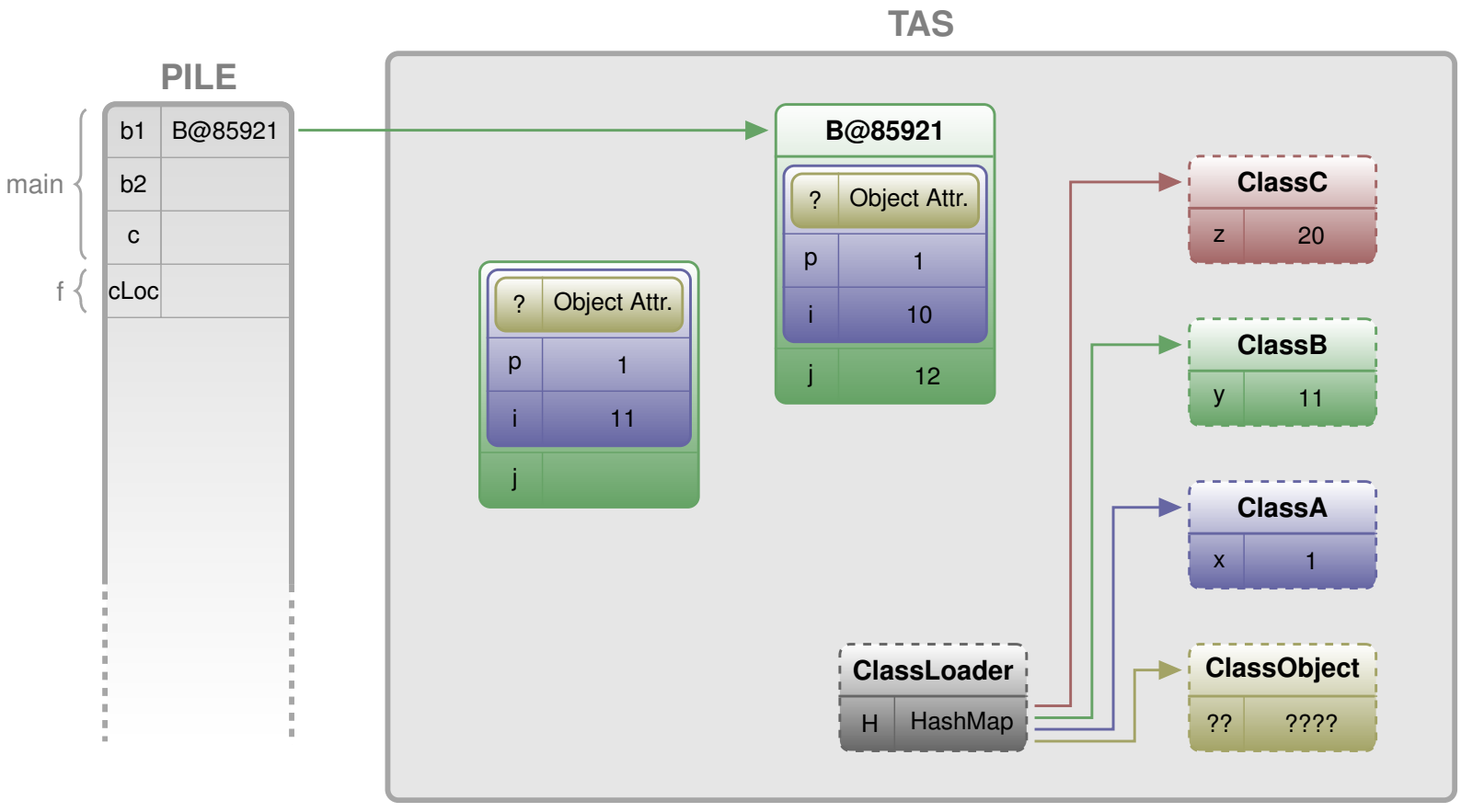

```
java Main
```

L'héritage



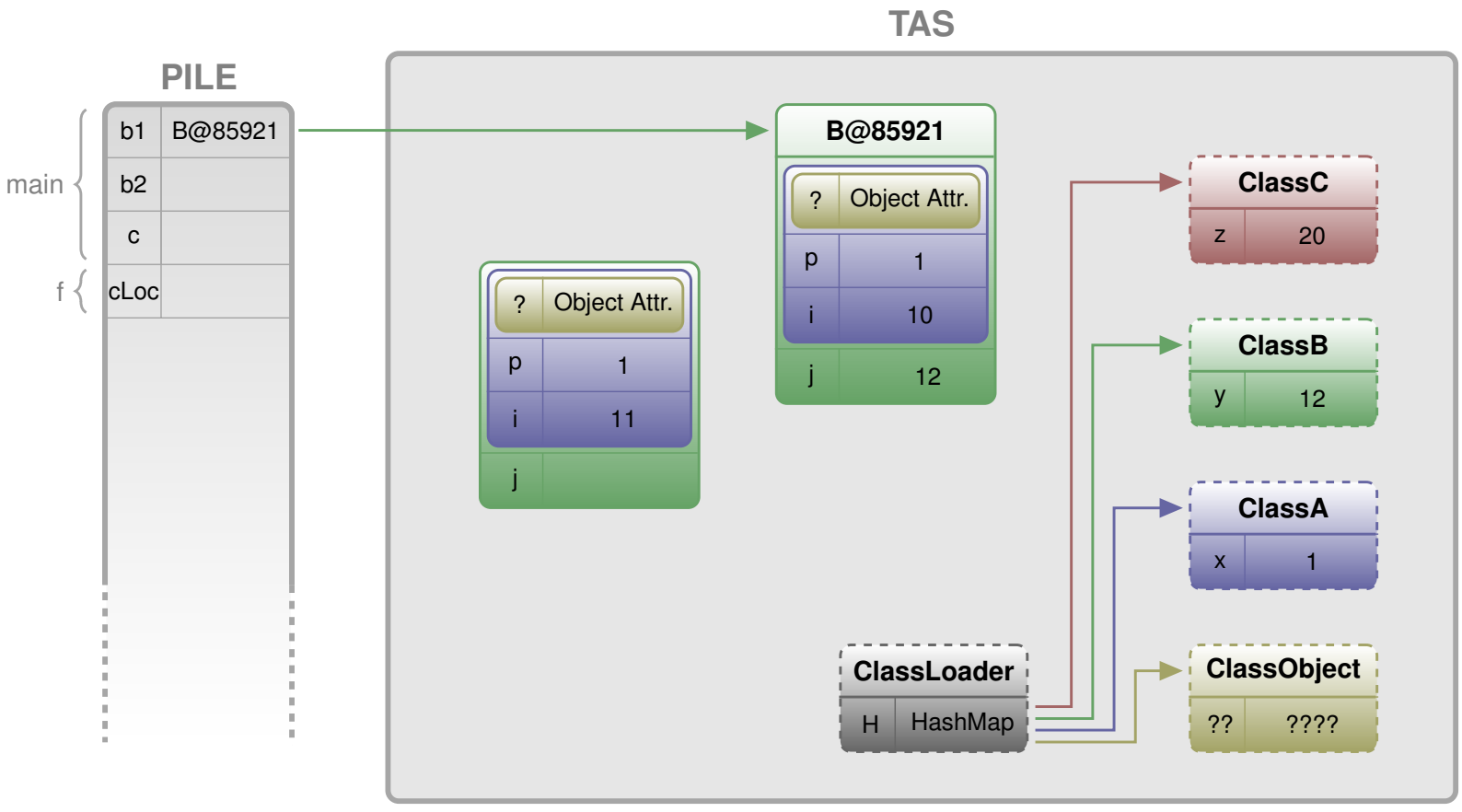

```
java Main
```

L'héritage



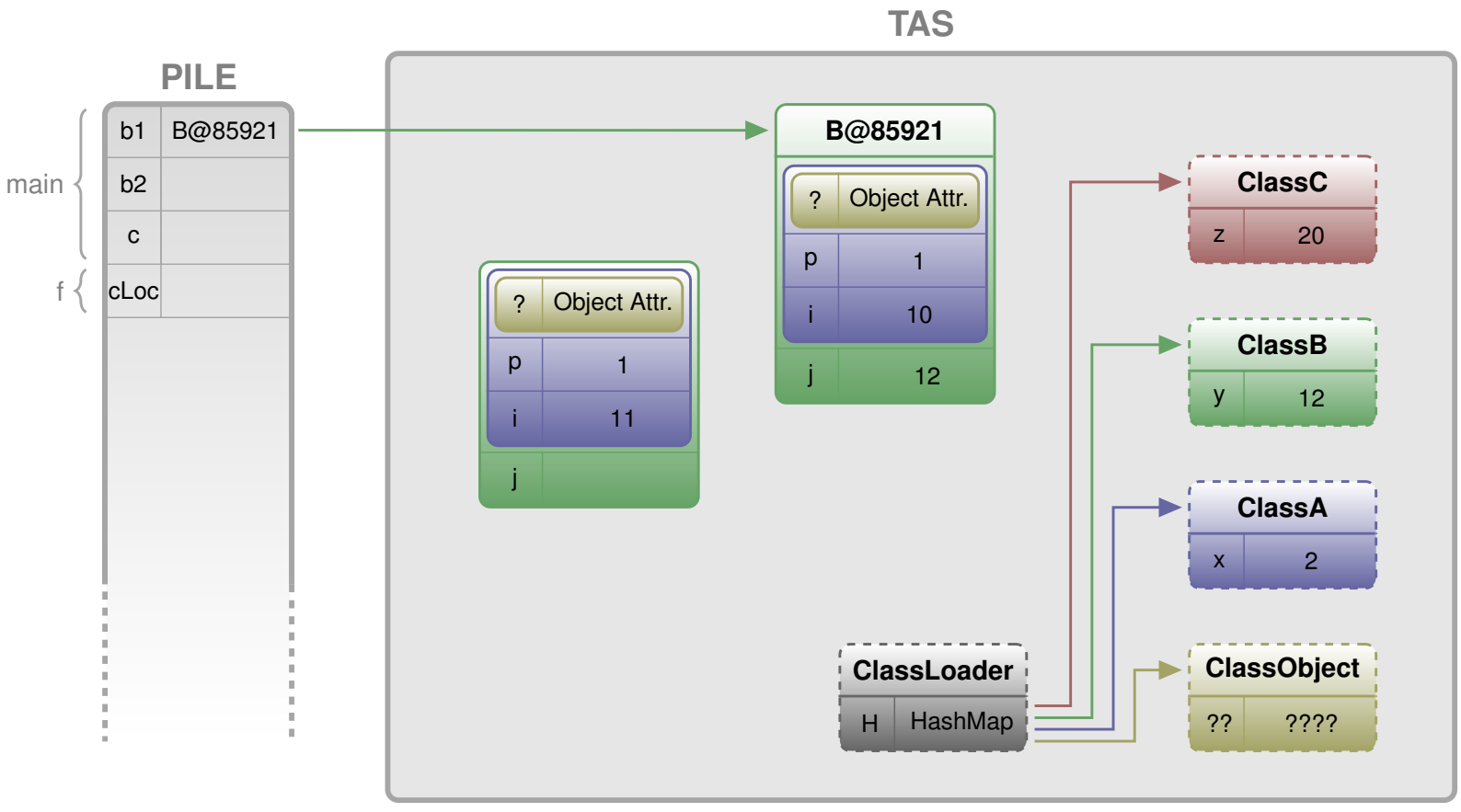

```
java Main
```

L'héritage



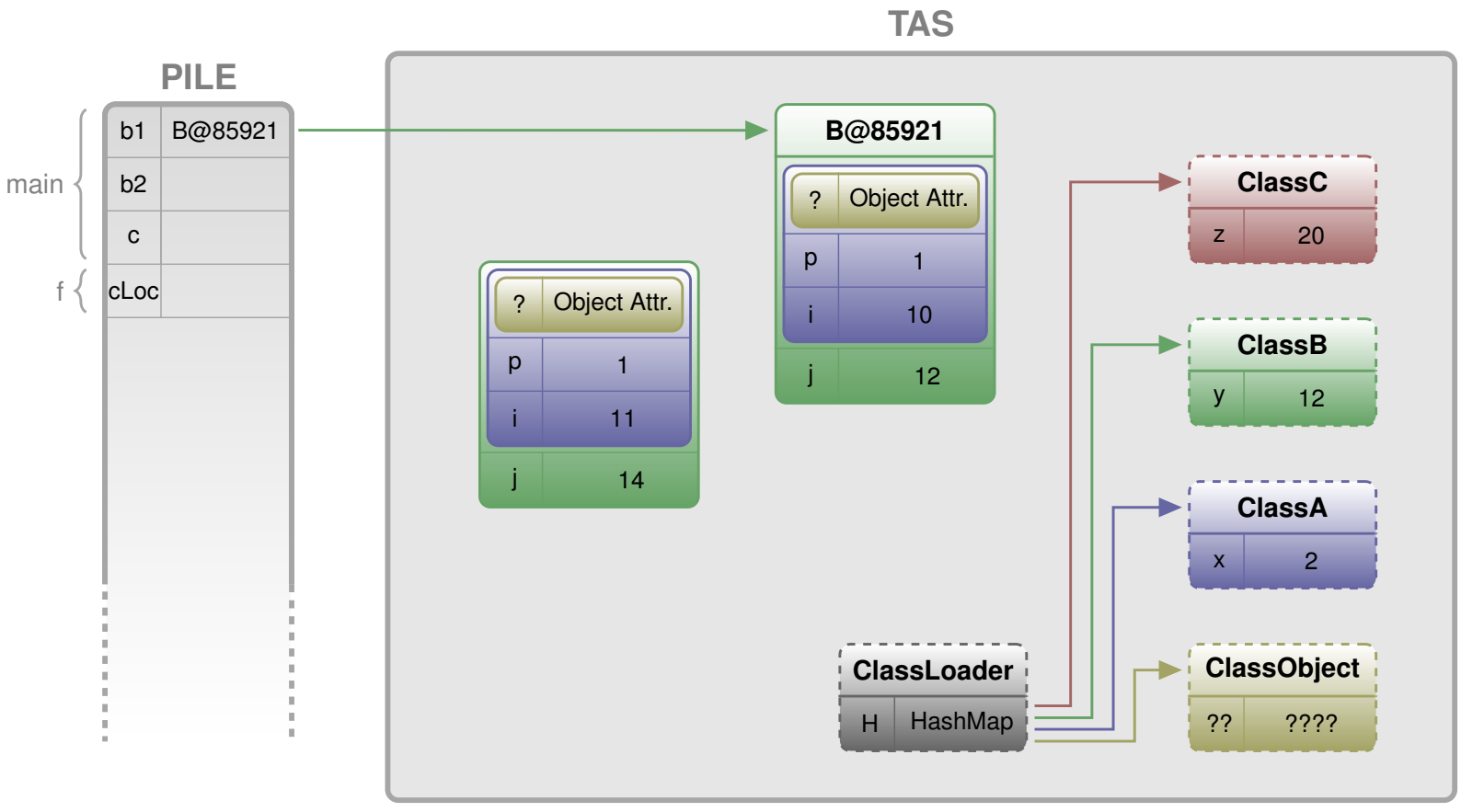

```
java Main
```

L'héritage



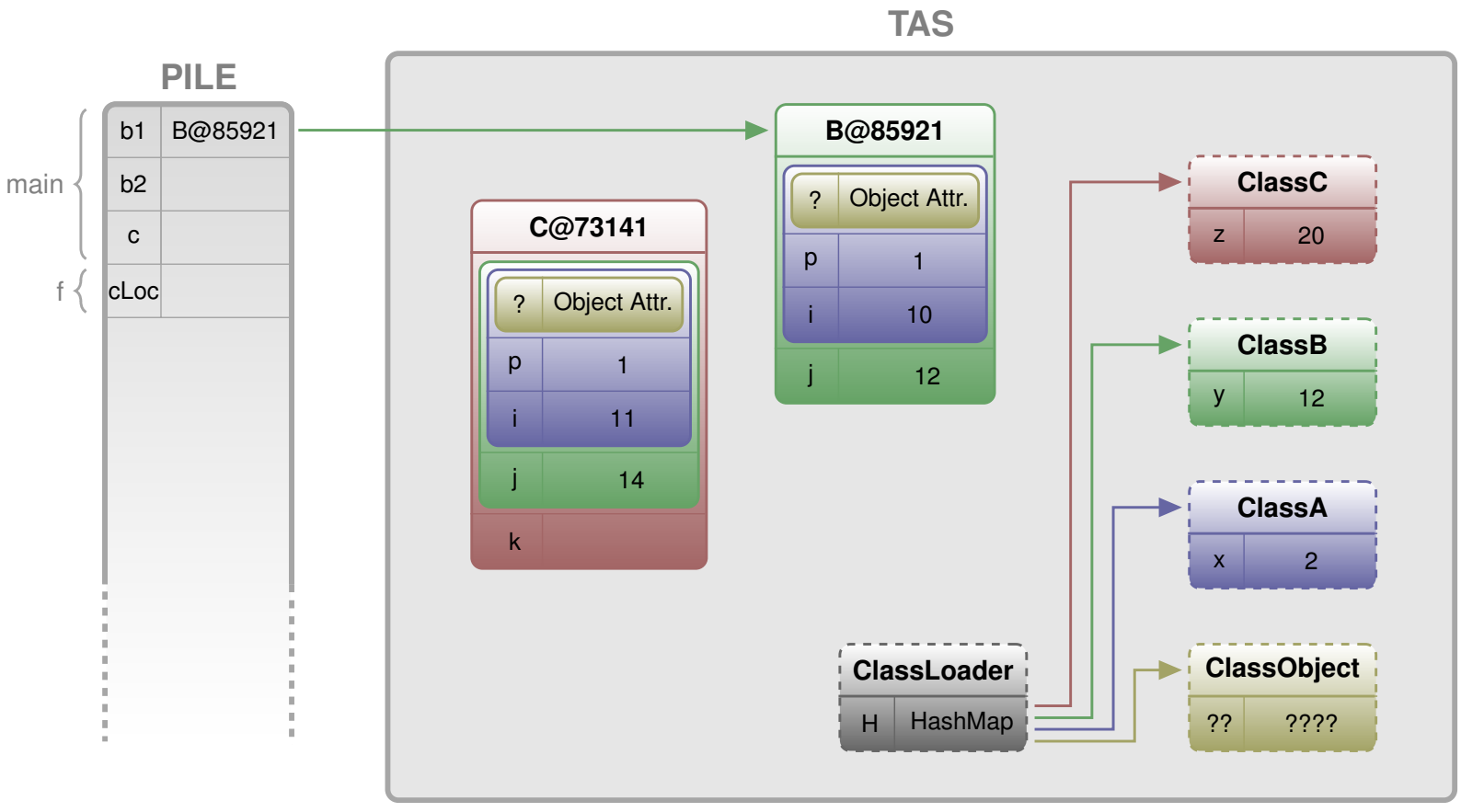

```
java Main
```

L'héritage



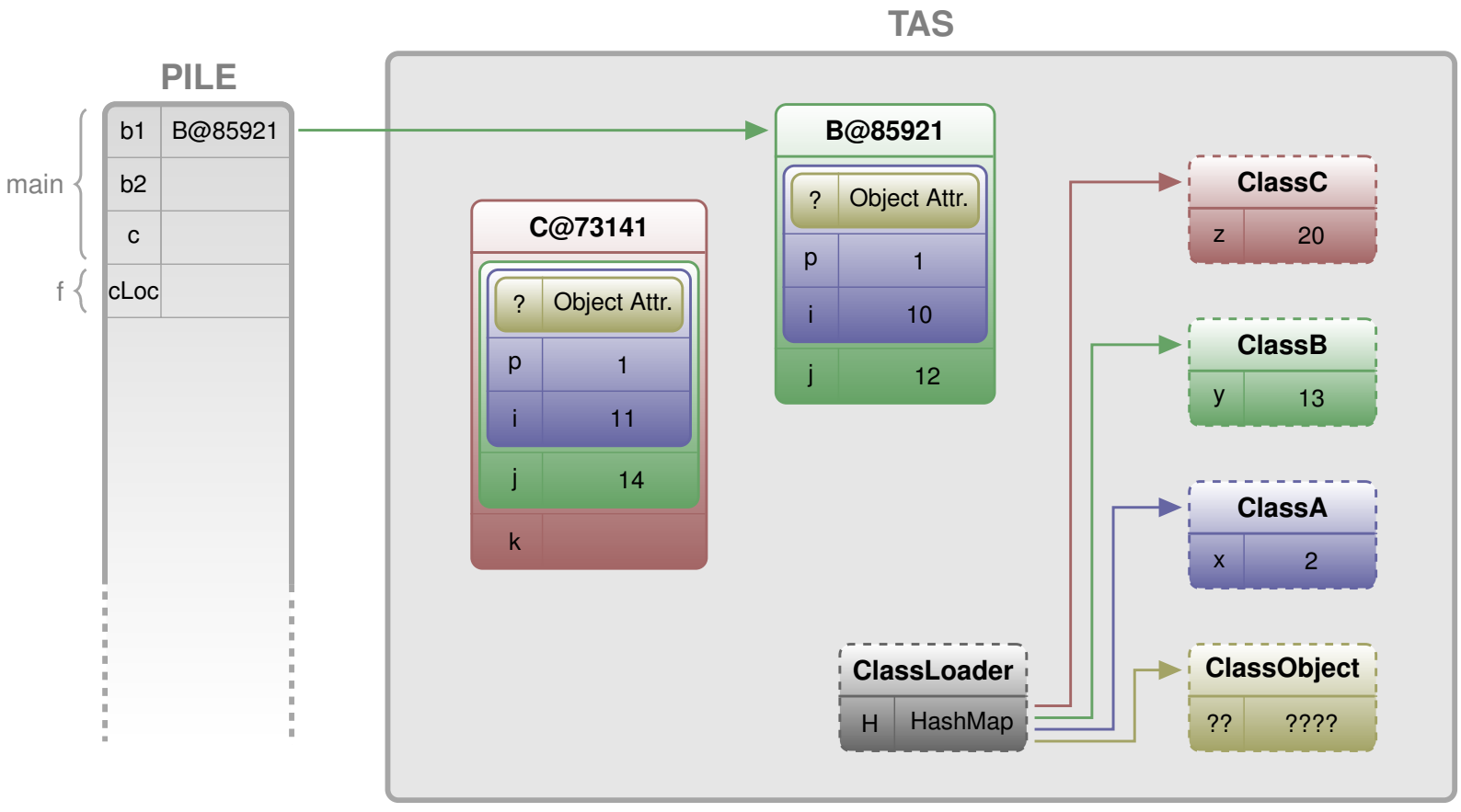

```
java Main
```

L'héritage



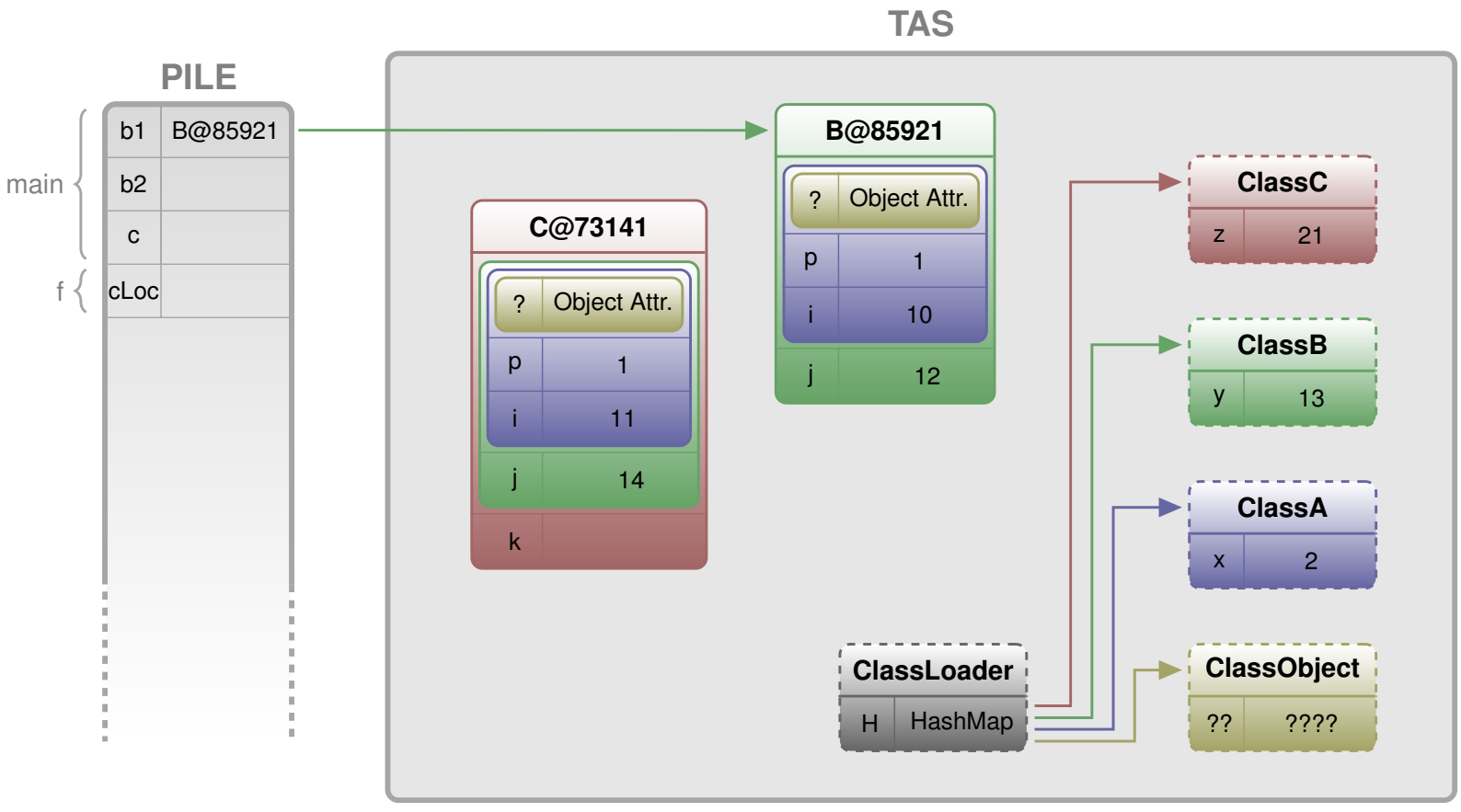

```
java Main
```

L'héritage



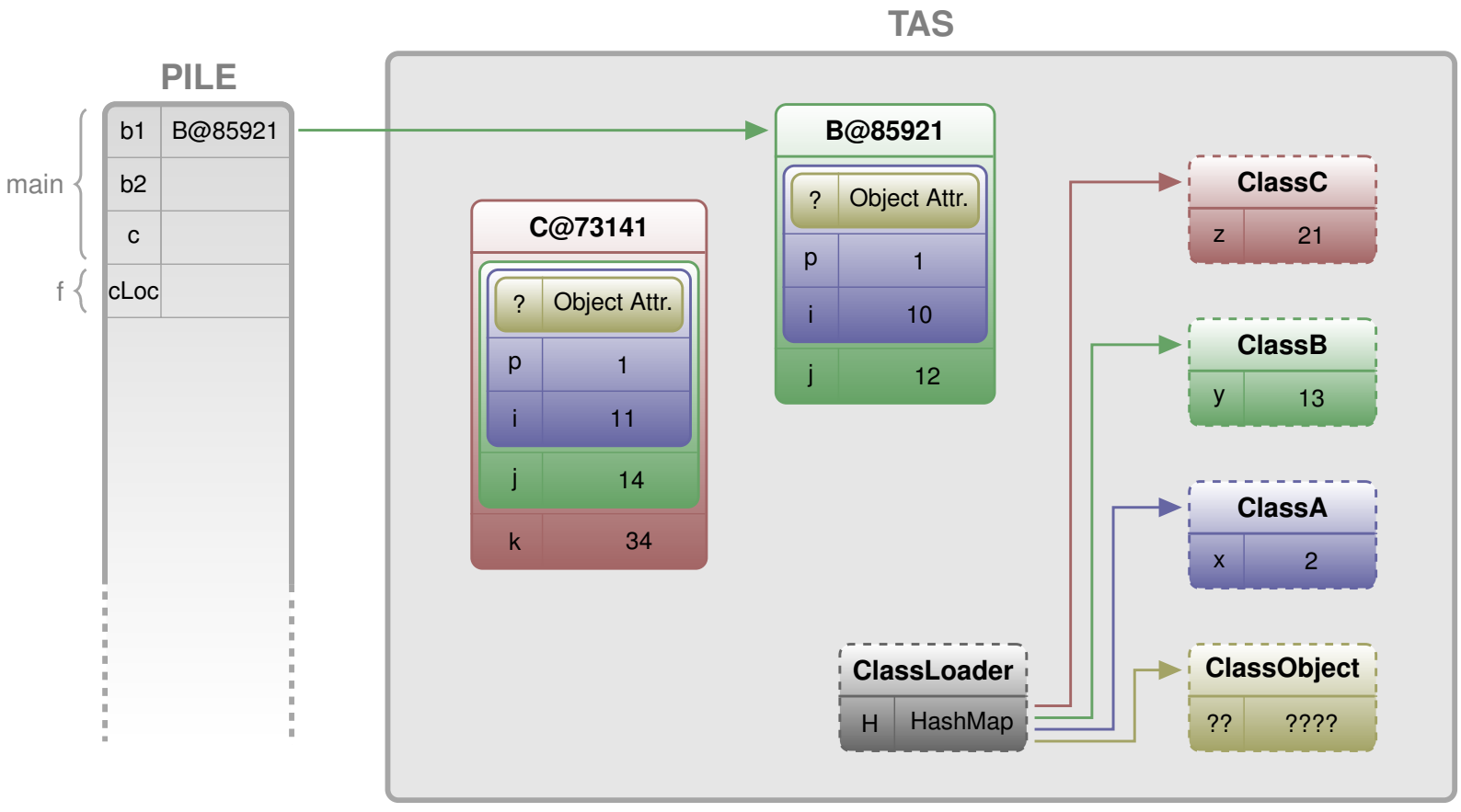

```
java Main
```

L'héritage



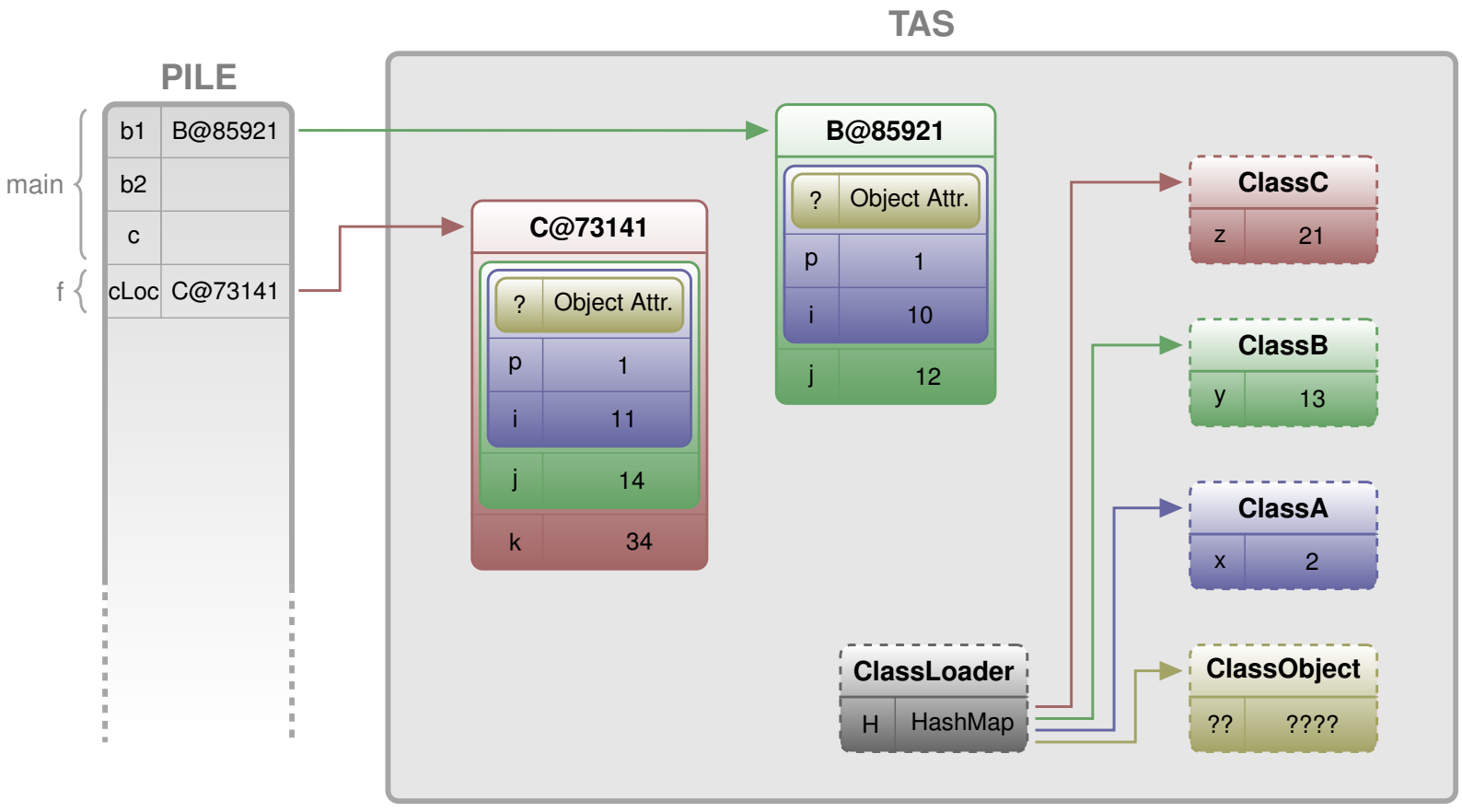

```
java Main
```

L'héritage



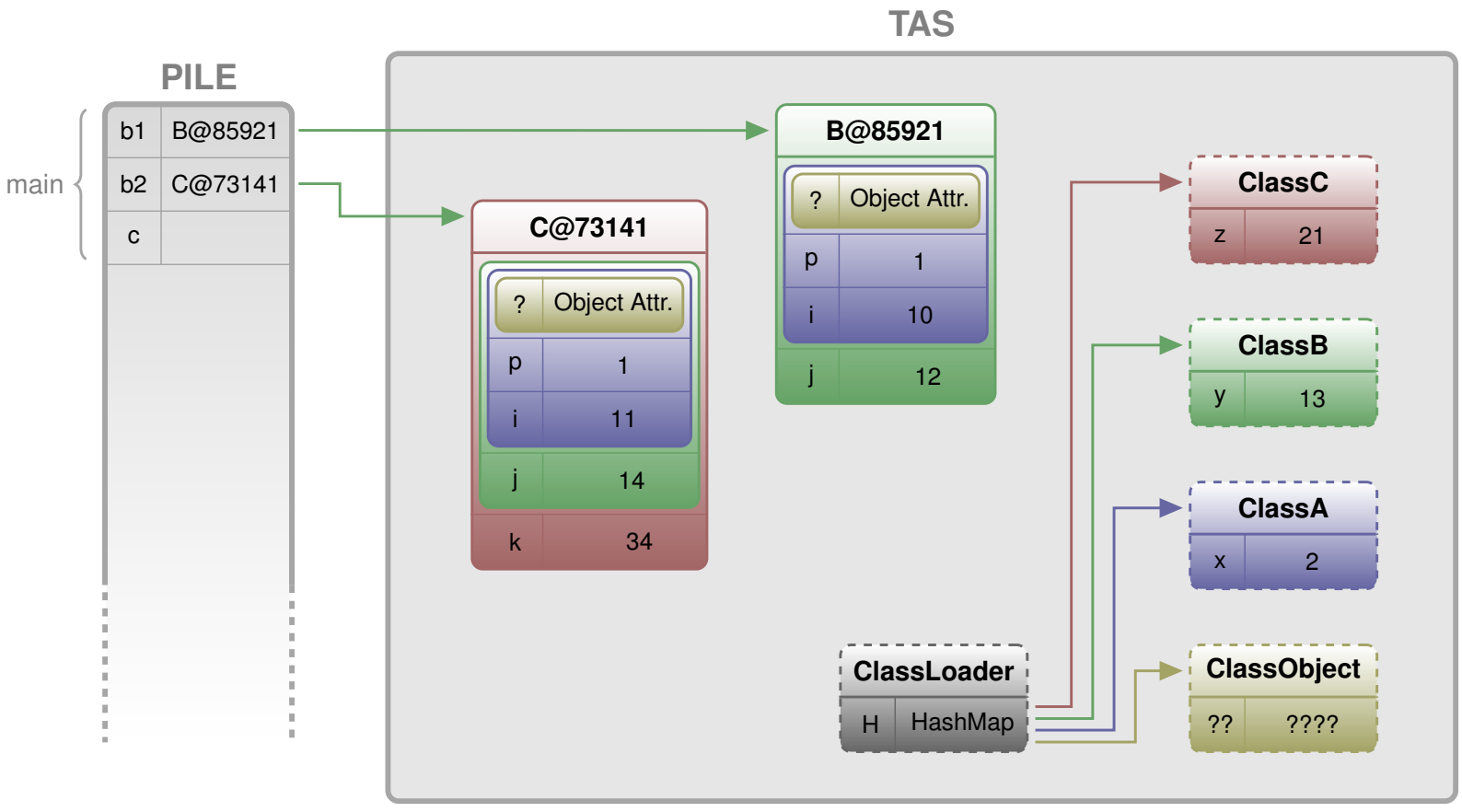

```
java Main
```

L'héritage



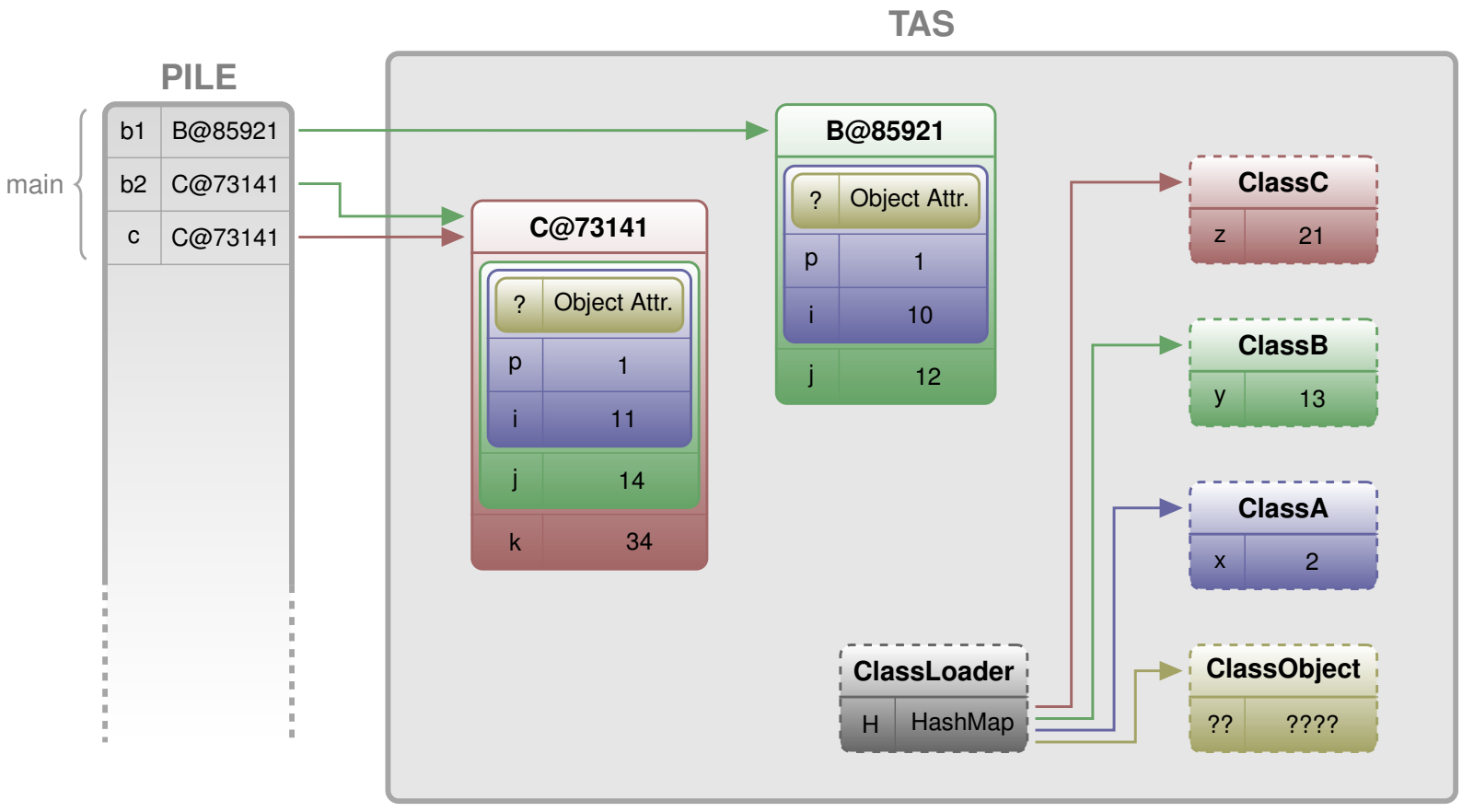

```
java Main
```

L'héritage



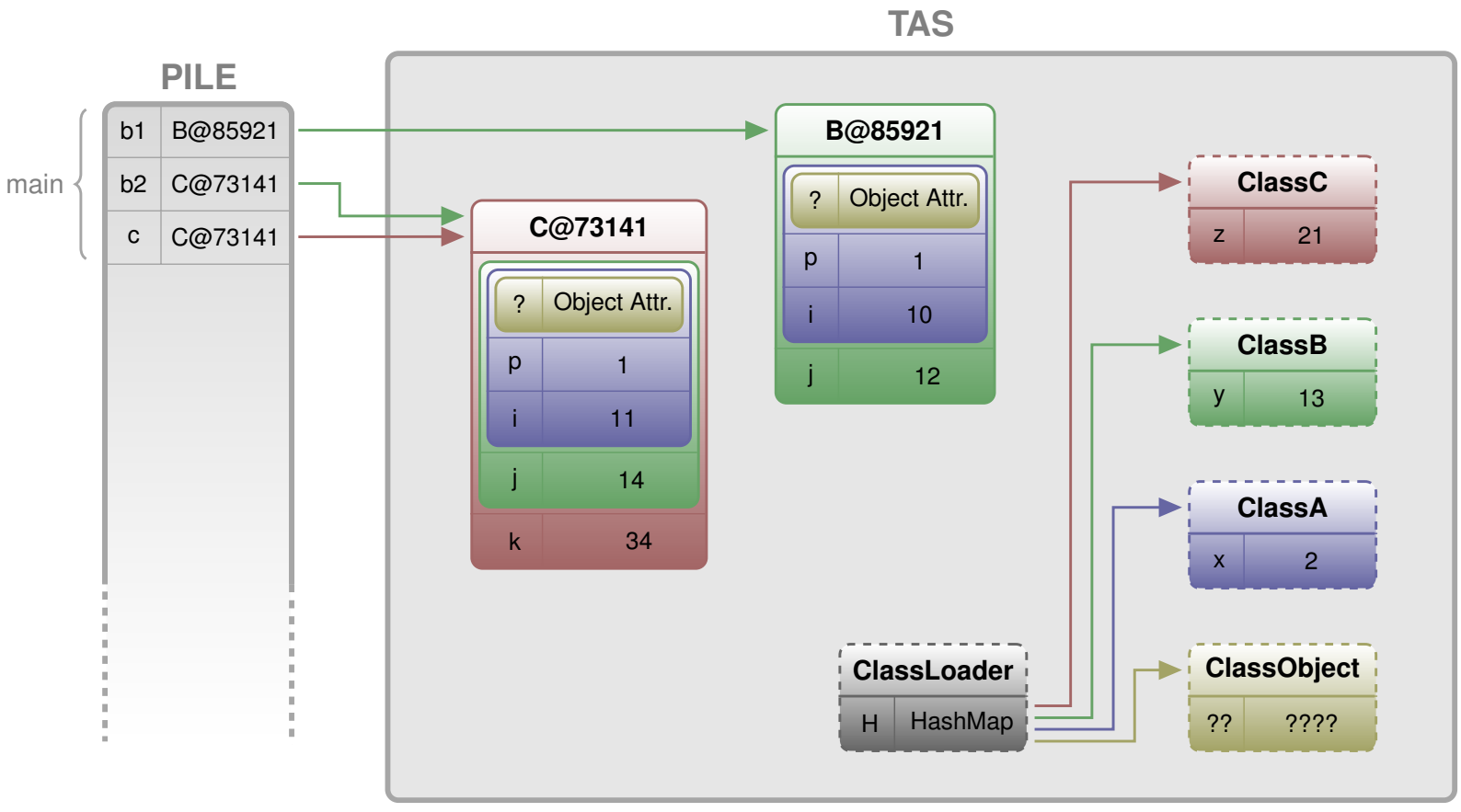

```
java Main
```

L'héritage



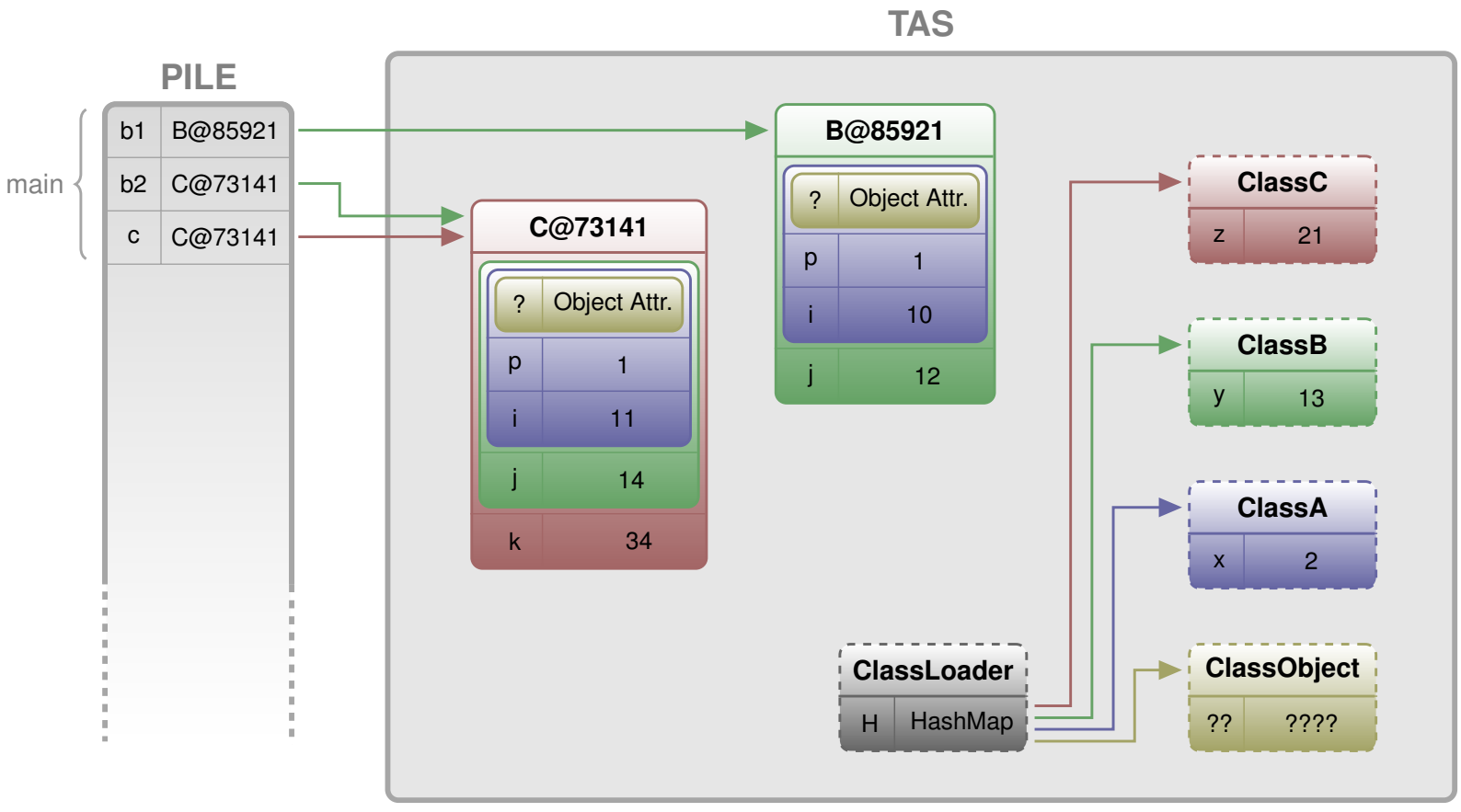

```
java Main
```

L'héritage



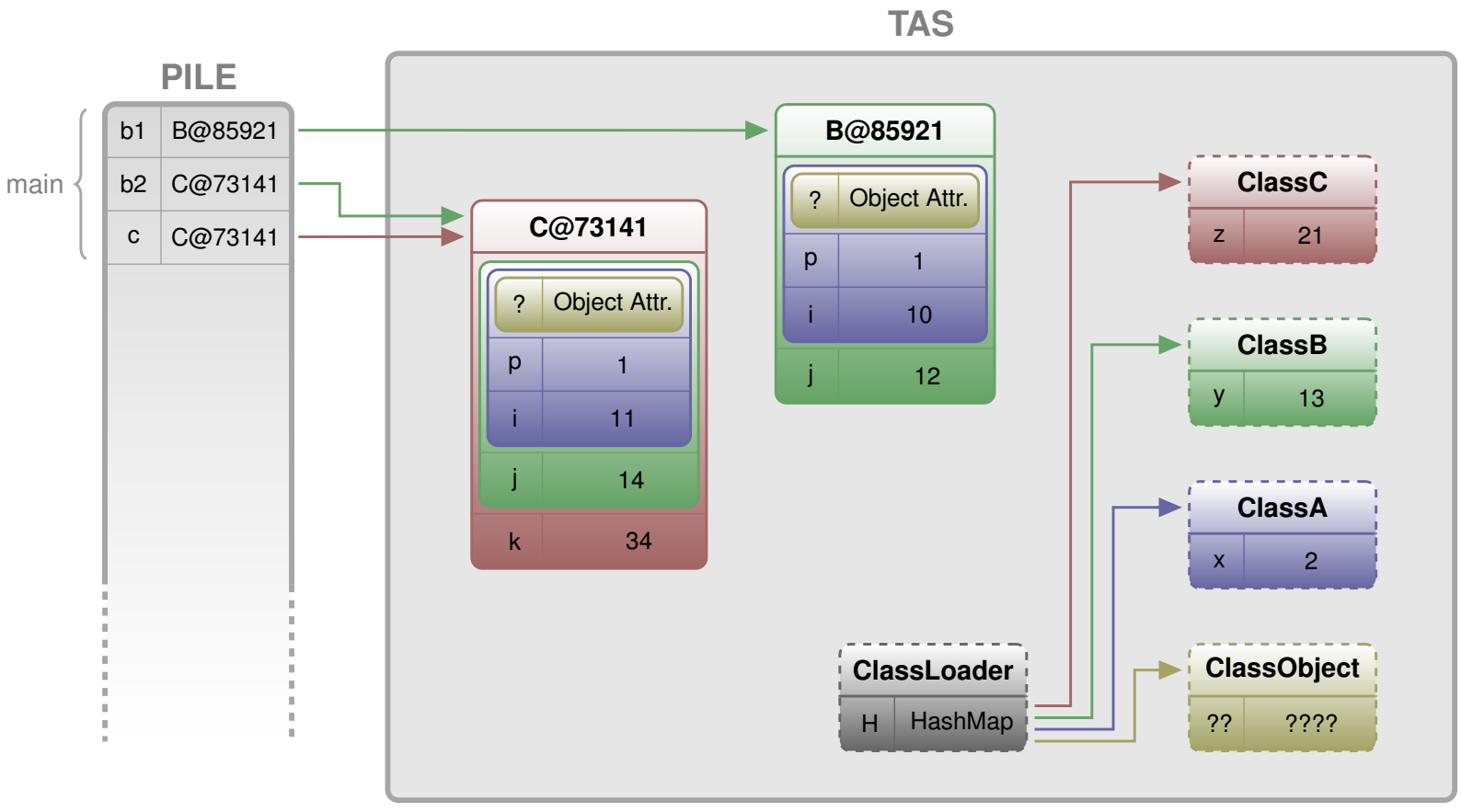

```
java Main
    j=12
```

L'héritage




```
java Main
    j=12 k=34
```

L'héritage




```
java Main
    j=12 k=34 k=34
```