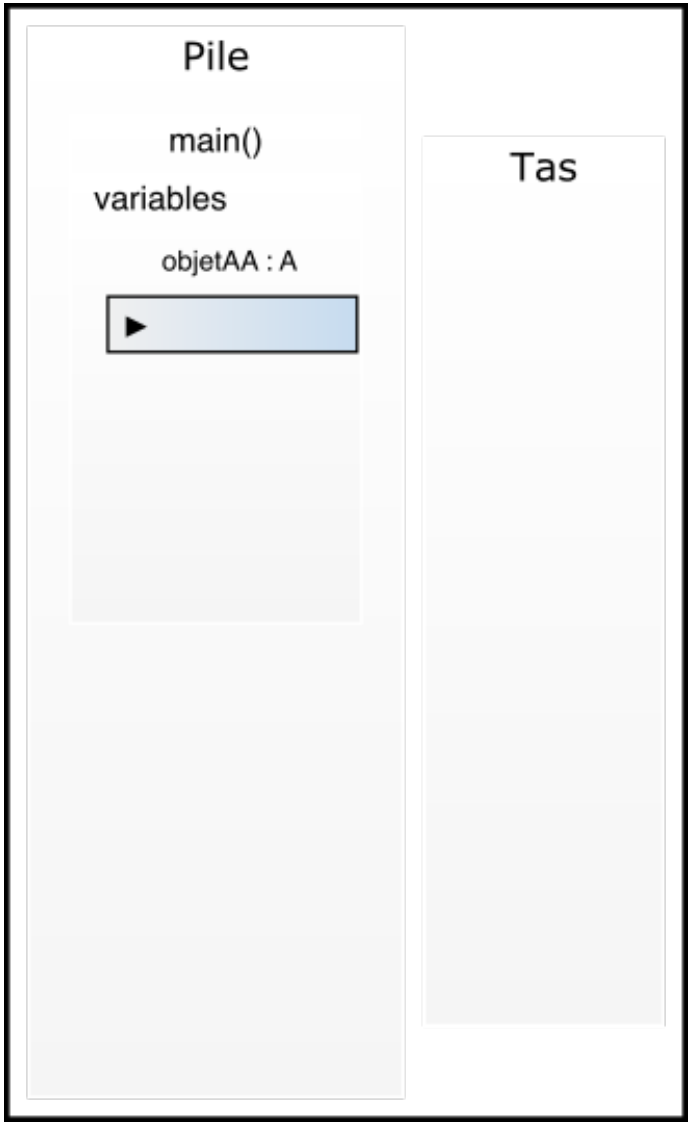
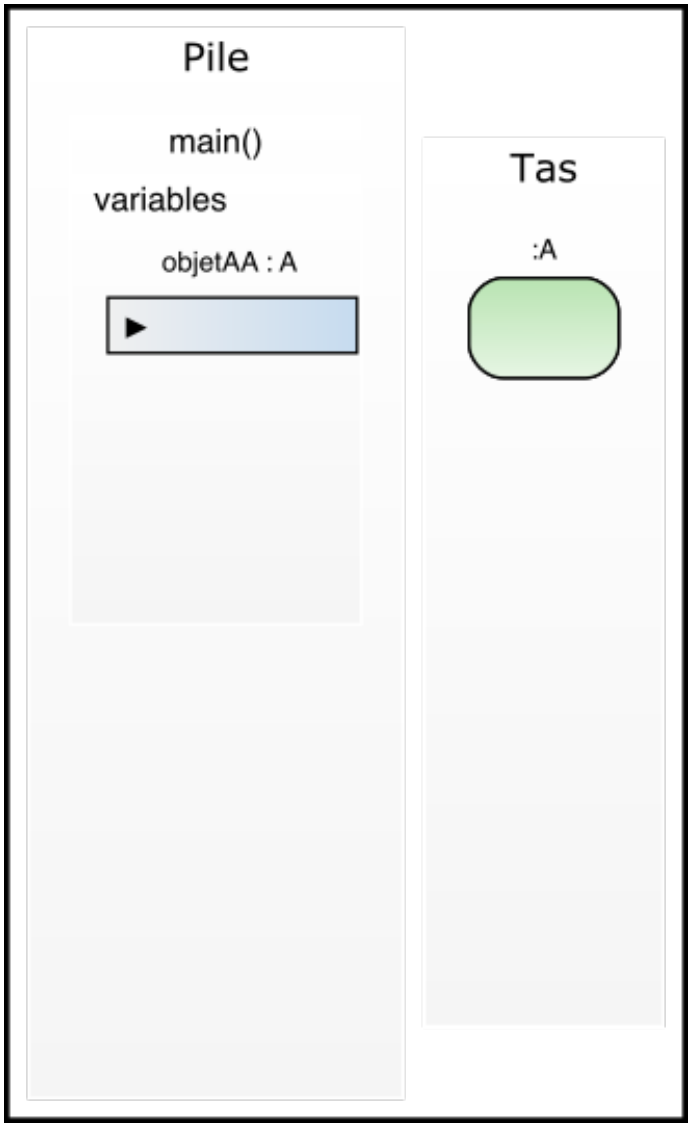



```
1  public class Test {
2      public static void main(String[] args) {
3          System.out.println("A objetAA = new A();");
4           A objetAA = new A();
5          System.out.println("A objetAB = new B();");
6          A objetAB = new B();
7          System.out.println("B objetBA = new A();");
8          // B objetBA = new A();
9      }
10 }
```

– Déclaration de la variable –

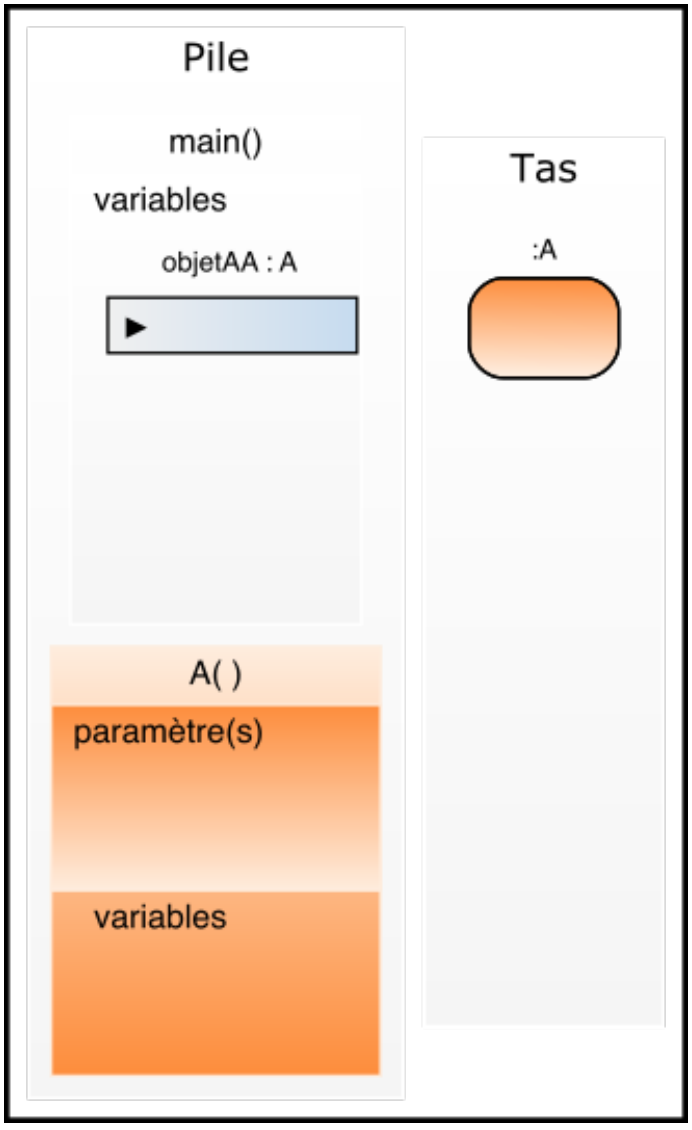
- nom: `objetAA`
- type: référence vers `A`




```
1  public class Test {
2      public static void main(String[] args) {
3          System.out.println("A objetAA = new A();");
4           A objetAA = new A();
5          System.out.println("A objetAB = new B();");
6          A objetAB = new B();
7          System.out.println("B objetBA = new A();");
8          // B objetBA = new A();
9      }
10 }
```

– Etape 1 de l'instanciation –

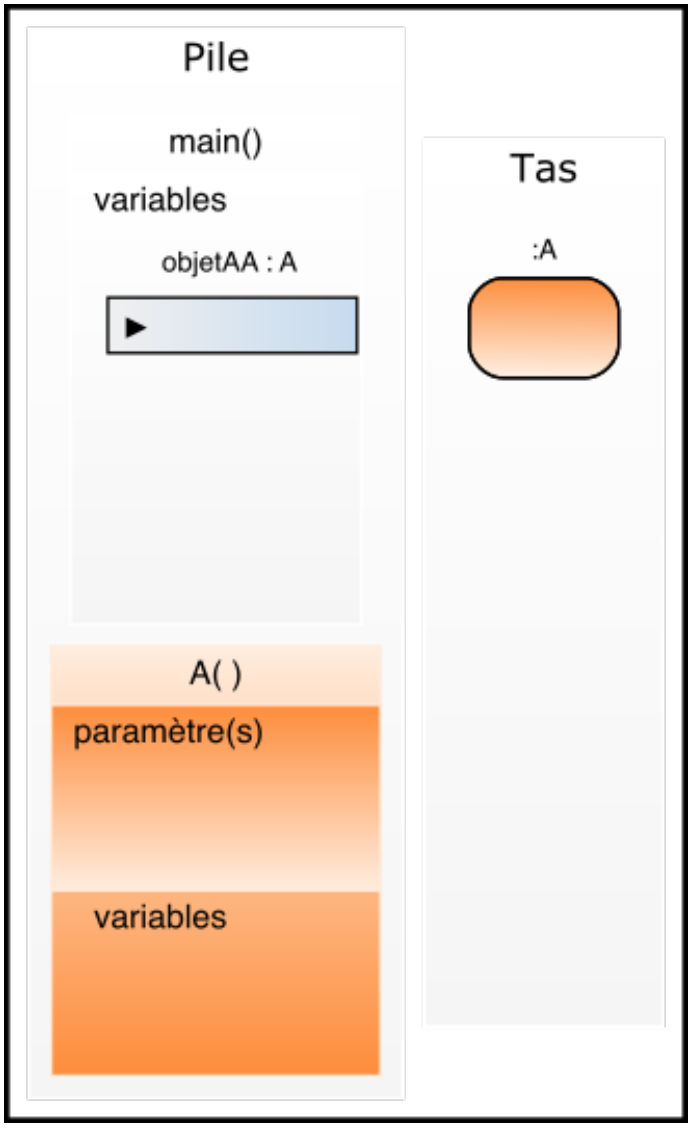
1. Appel à l'opérateur `new`




```
1  public class A {
2        public A()
3            System.out.println("> Constructeur de A()");
4            afficher();
5        }
6
7        public void afficher() {
8            System.out.println("> Affichage de A");
9        }
10 }
```

– Etape 2 de l'instanciation –

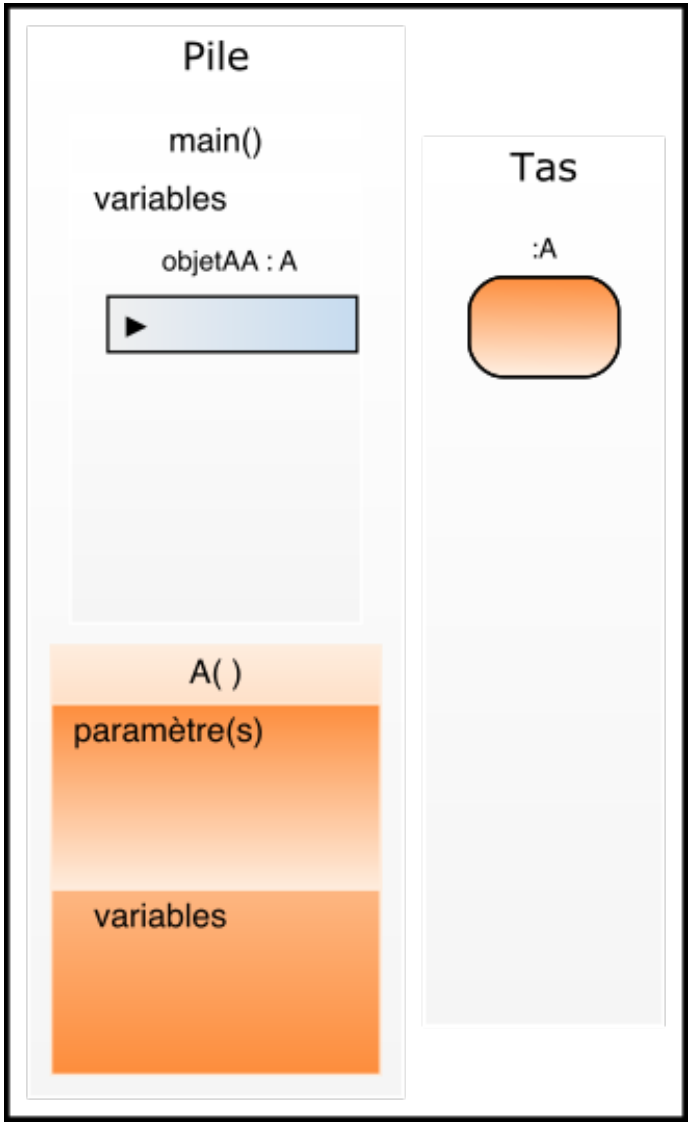
1. Appel à l'opérateur `new`
2. Appel au constructeur du type `A`




```
1  public class A {
2      public A() {
3          System.out.println("> Constructeur de A()");
4           afficher();
5      }
6
7      public void afficher() {
8          System.out.println(">␣Affichage␣de␣A");
9      }
10 }
```

– On affiche –

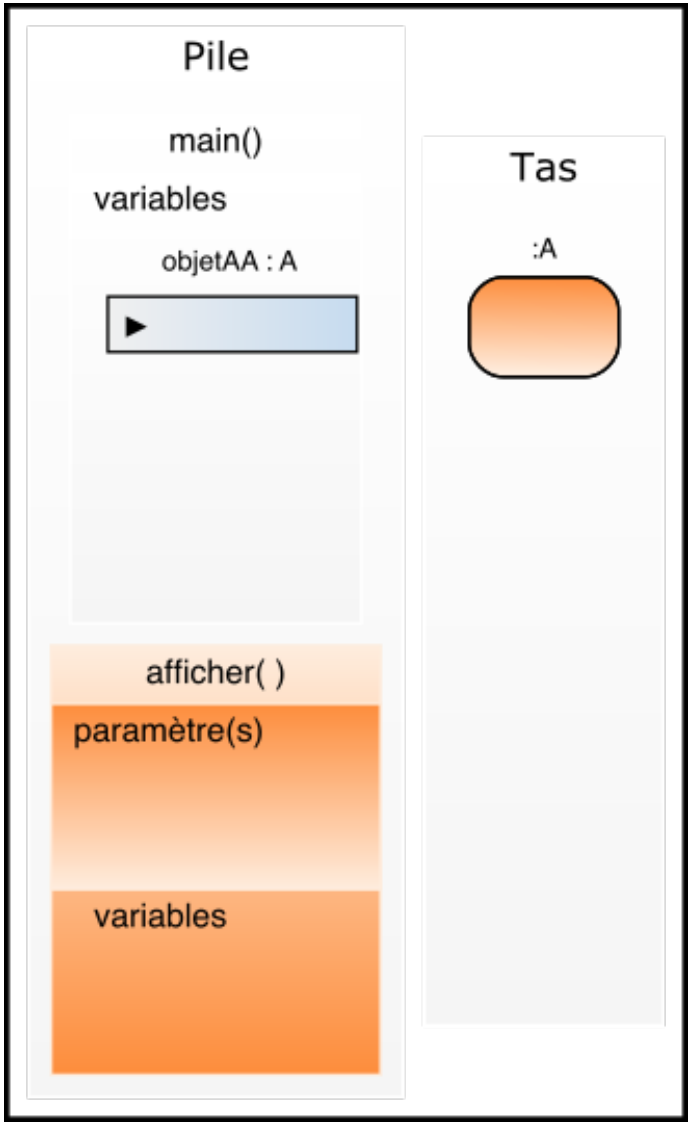
```
> A objet = new A();
> Constructeur de A()
```




```
1  public class A {
2      public A() {
3          System.out.println("> Constructeur de A()");
4          afficher();
5      }
6
7      public void afficher() {
8          System.out.println("> Affichage de A");
9      }
10 }
```

– Appel à la méthode –

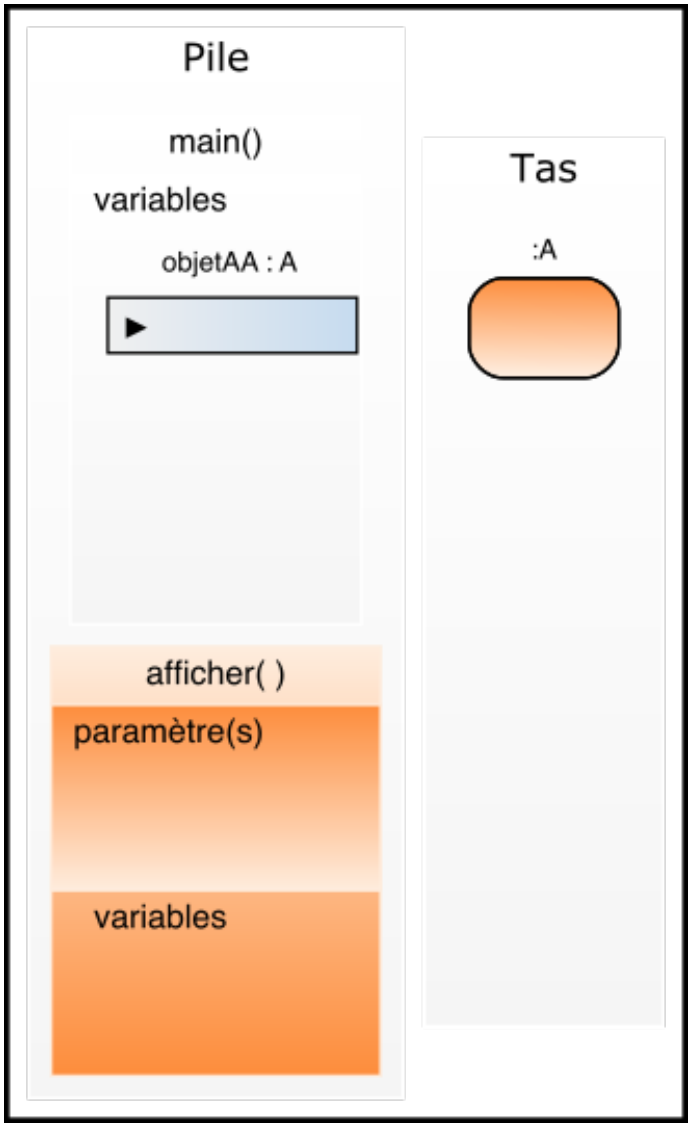
- `afficher()`
- sur l'instance courante




```
1  public class A {
2      public A() {
3          System.out.println("> Constructeur de A()");
4          afficher();
5      }
6
7       public void afficher()
8          System.out.println("> Affichage de A");
9      }
10 }
```

– Appel à la méthode –

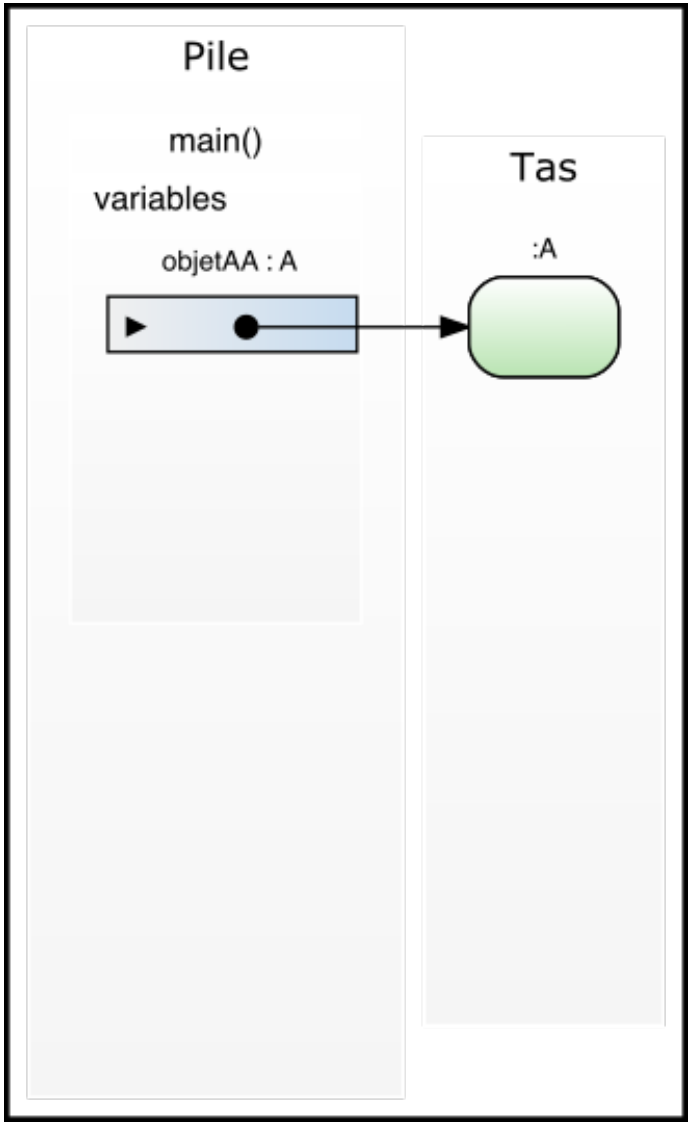
- `afficher()`
- sur l'instance courante




```
1  public class A {
2      public A() {
3          System.out.println("> Constructeur de A()");
4          afficher();
5      }
6
7      public void afficher() {
8          System.out.println("> Affichage de A");
9      }
10 }
```

– On affiche –

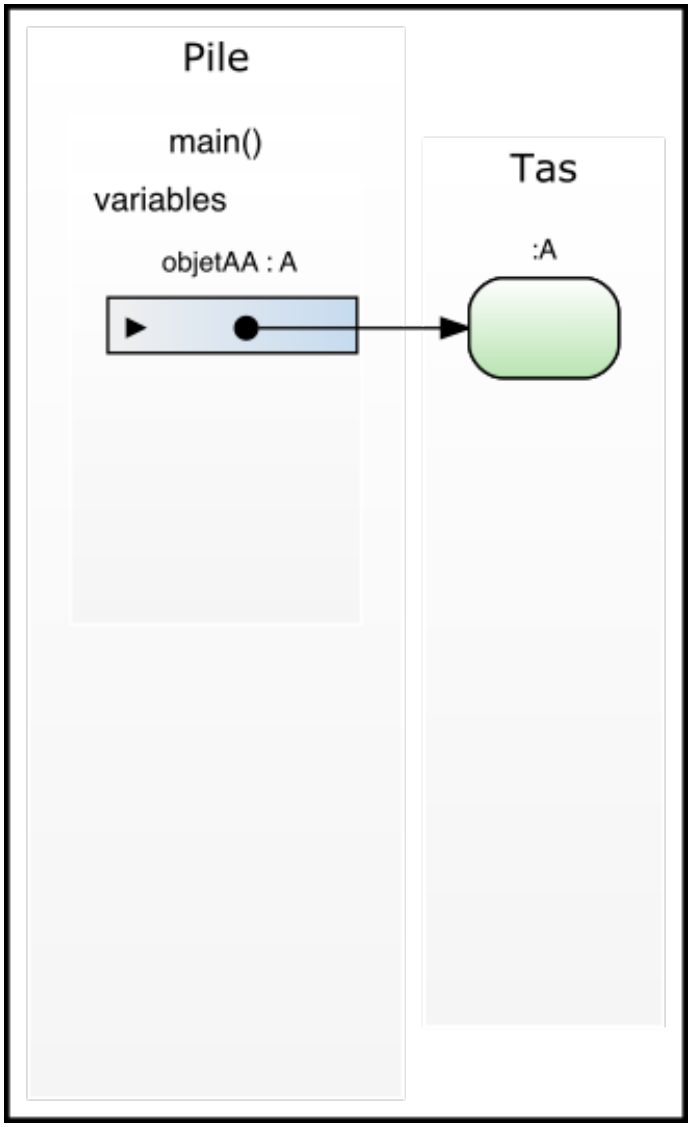
```
> A objet = new A();
> Constructeur de A()
> Affichage de A
```




```
1  public class Test {
2      public static void main(String[] args) {
3          System.out.println("A objetAA = new A();");
4           A objetAA = new A();
5          System.out.println("A objetAB = new B();");
6          A objetAB = new B();
7          System.out.println("B objetBA = new A();");
8          // B objetBA = new A();
9      }
10 }
```

– Etape 3 de l'instanciation –

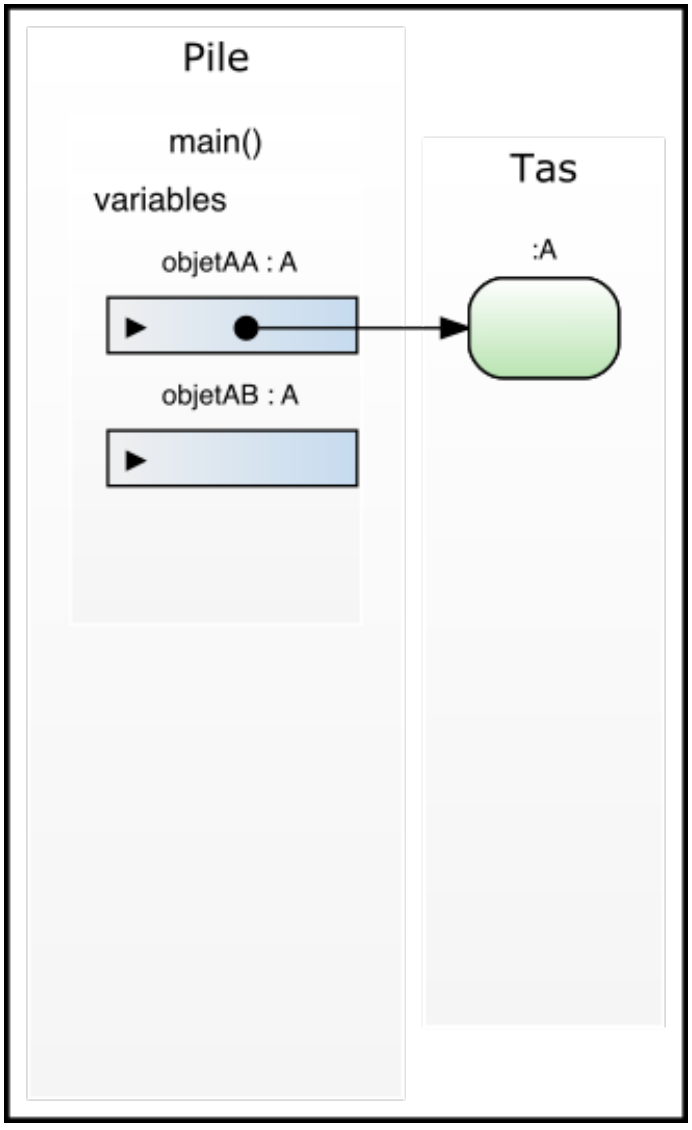
1. Appel à l'opérateur `new`
2. Appel au constructeur du type `A`
3. Affectation de la référence




```
1  public class Test {
2      public static void main(String[] args) {
3          System.out.println("A objetAA = new A();");
4          A objetAA = new A();
5           System.out.println("A objetAB = new B();");
6          A objetAB = new B();
7          System.out.println("B objetBA = new A();");
8          // B objetBA = new A();
9      }
10 }
```

– On affiche –

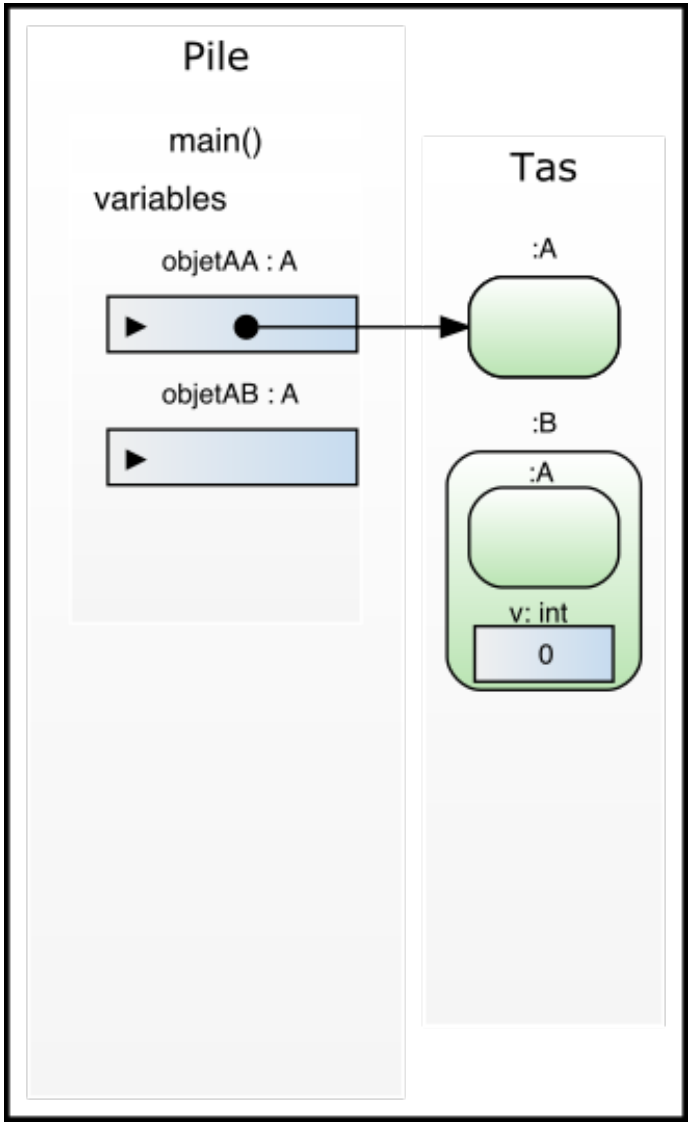
```
> A objetAA = new A();
> Constructeur de A()
> Affichage de A

> A objetAB = new B();
```




```
1  public class Test {
2      public static void main(String[] args) {
3          System.out.println("A objetAA = new A();");
4          A objetAA = new A();
5          System.out.println("A objetAB = new B();");
6           A objetAB = new B();
7          System.out.println("B objetBA = new A();");
8          // B objetBA = new A();
9      }
10 }
```

– Déclaration de la variable –

- nom: `objetAB`
- type: référence vers `A`

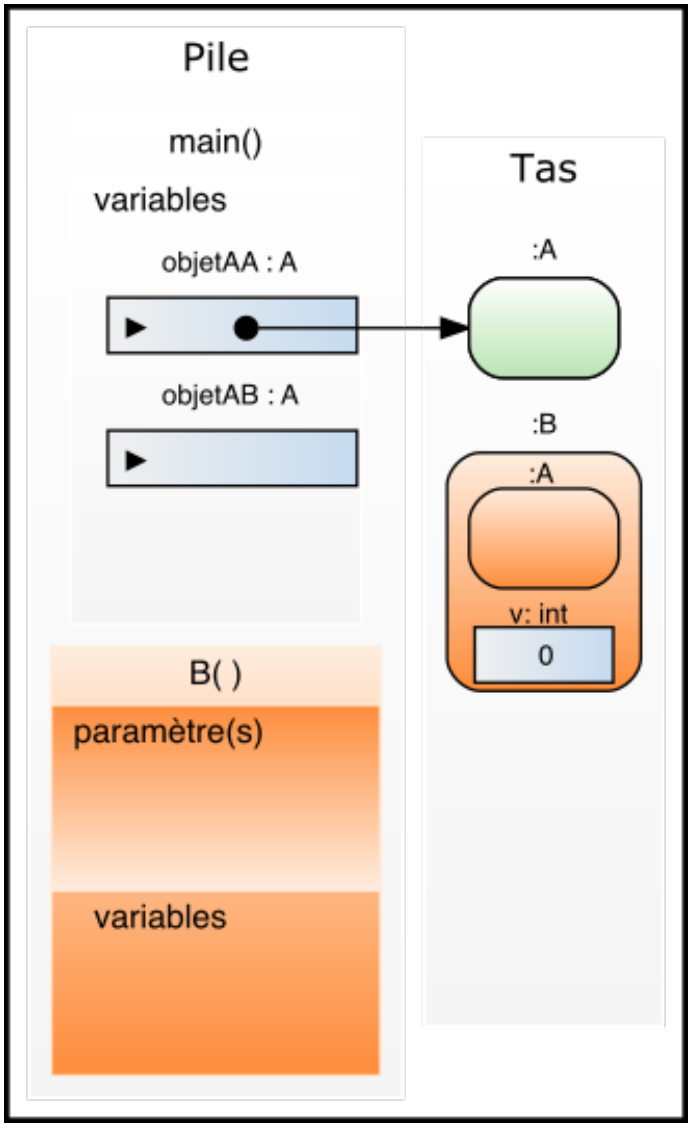



```
1  public class Test {
2      public static void main(String[] args) {
3          System.out.println("A objetAA = new A();");
4          A objetAA = new A();
5          System.out.println("A objetAB = new B();");
6           A objetAB = new B();
7          System.out.println("B objetBA = new A();");
8          // B objetBA = new A();
9      }
10 }
```

– Etape 1 de l'instanciation –

1. Appel à l'opérateur `new`

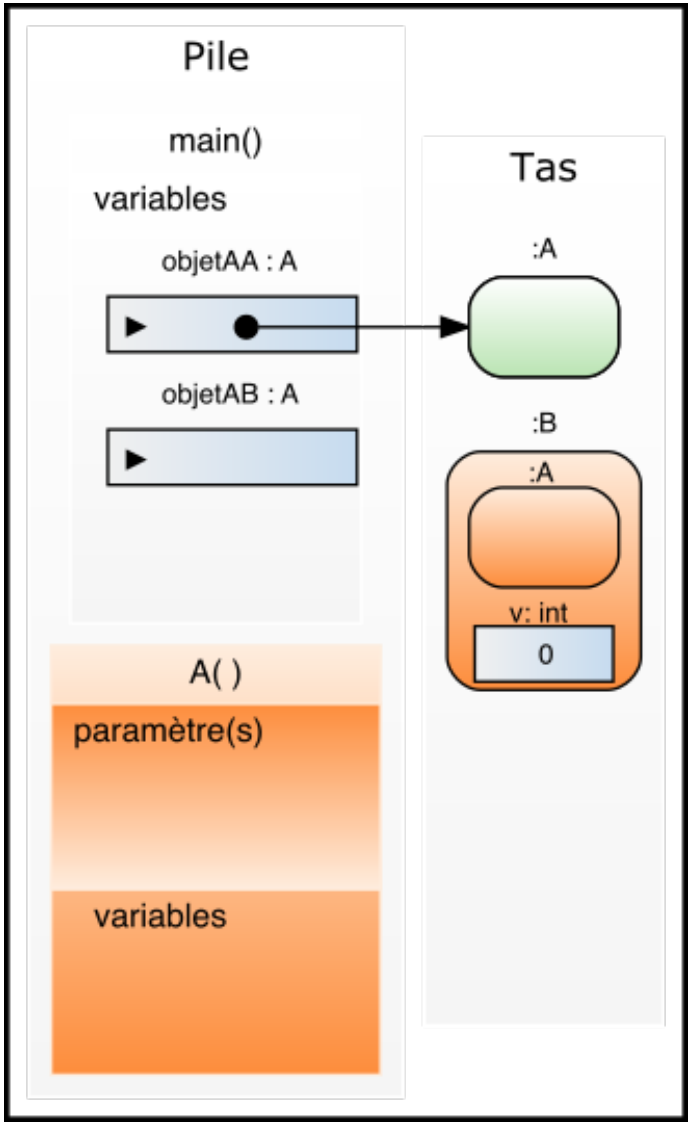
Ici, on remarque qu'une instance de `B` est aussi une instance de `A`, en ceci qu'elle contient les attributs et les méthodes de `A` en plus de ses propres attributs et propres méthodes.




```
1  public class B extends A {
2      private int v;
3       public B() {
4           System.out.println("> Constructeur de B()");
5           v = 5;
6           afficher();
7       }
8
9       public void afficher() {
10          System.out.println("> v = " + v);
11      }
12 }
```

– Etape 2 de l'instanciation –

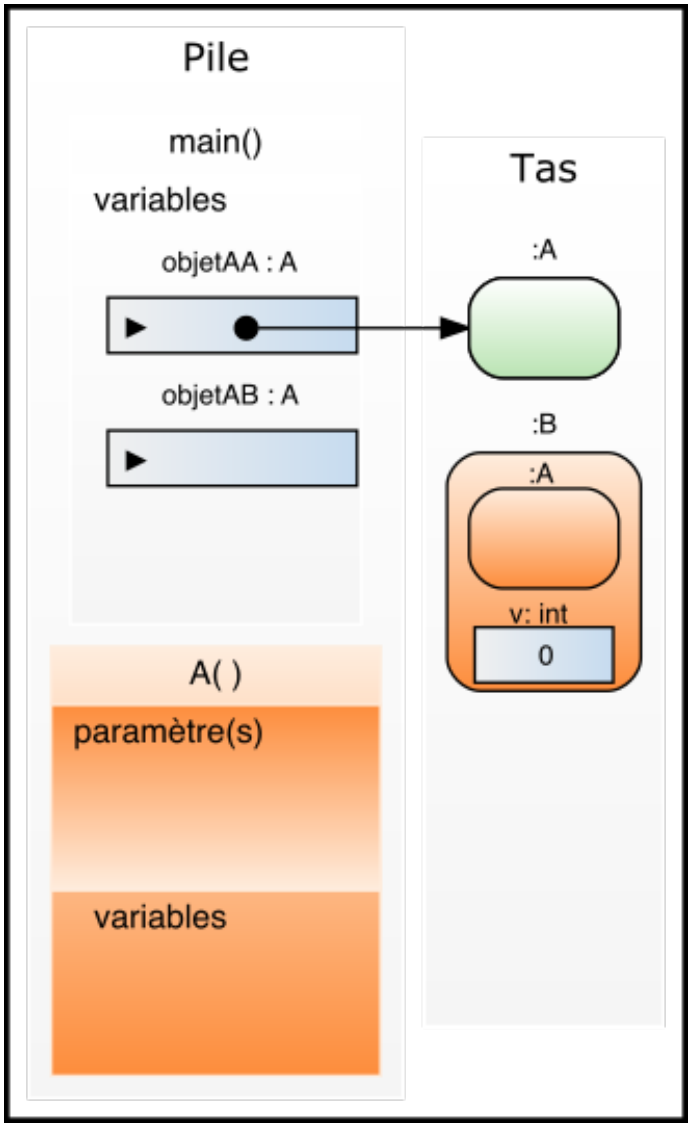
1. Appel à l'opérateur `new`
2. Appel au constructeur du type `B`




```
1  public class A {
2        public A() {
3            System.out.println("> Constructeur de A()");
4            afficher();
5        }
6
7        public void afficher() {
8            System.out.println("> Affichage de A");
9        }
10 }
```

– Constructeur de A –

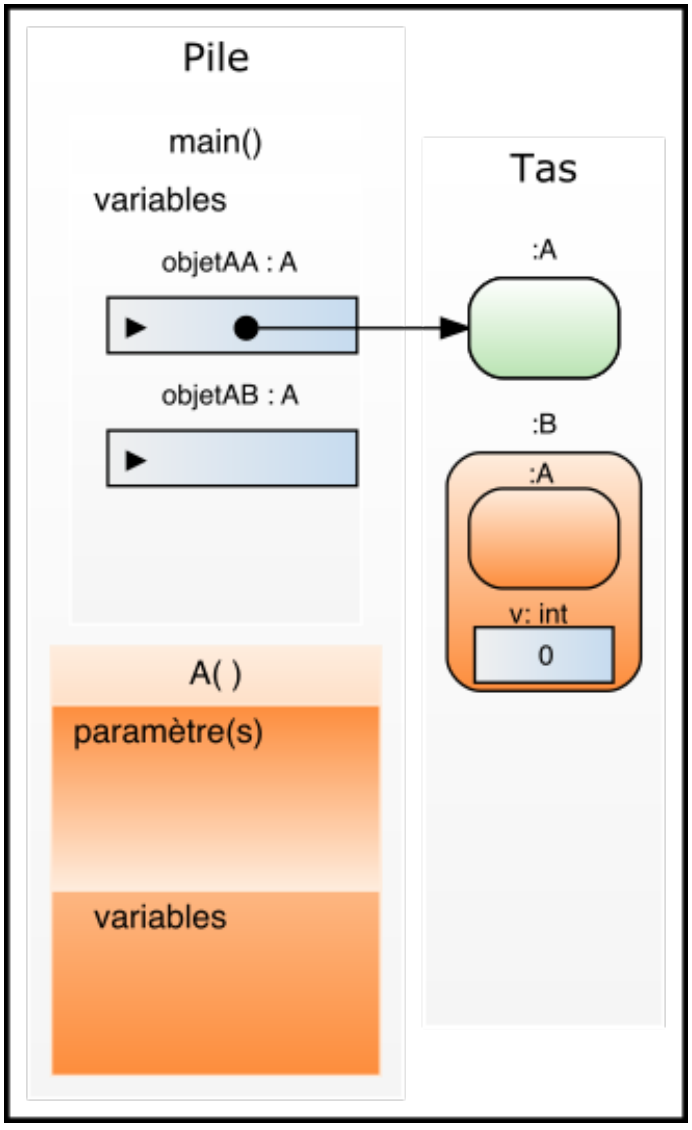
La première ligne du constructeur de `B` appelle silencieusement et implicitement le constructeur par défaut de `A`.




```
1   public class A {
2       public A() {
3           System.out.println("> Constructeur de A()");
4           afficher();
5       }
6
7       public void afficher() {
8           System.out.println("> Affichage de A");
9       }
10  }
```

– On affiche –

```
> A objetAA = new A();
> Constructeur de A()
> Affichage de A
> B objetBB = new B();
> Constructeur de A()
```

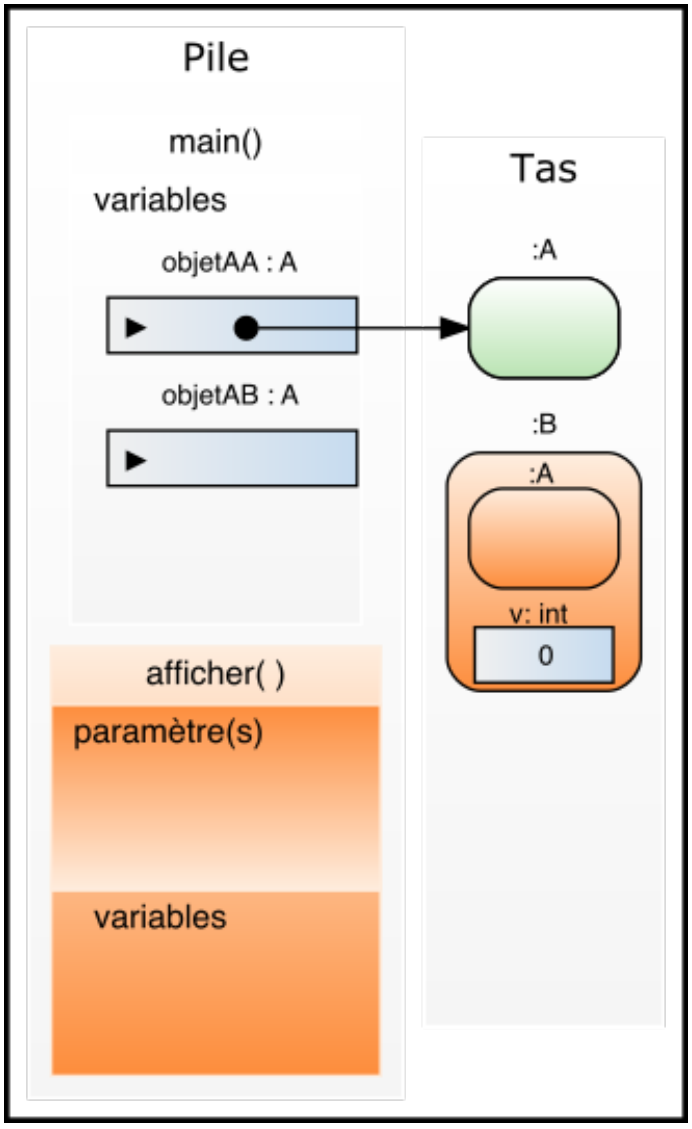



```
1  public class A {
2      public A() {
3          System.out.println("> Constructeur de A()");
4           afficher();
5      }
6
7      public void afficher() {
8          System.out.println("> Affichage de A");
9      }
10 }
```

– Appel à la méthode –

- `afficher()`
- sur l'instance courante de B...

En effet, l'instance de `B` a été créée. Il s'agit de l'instance courante, même dans le constructeur de `A`... Du coup, on appelle la méthode `afficher()` implémentée dans `B`.

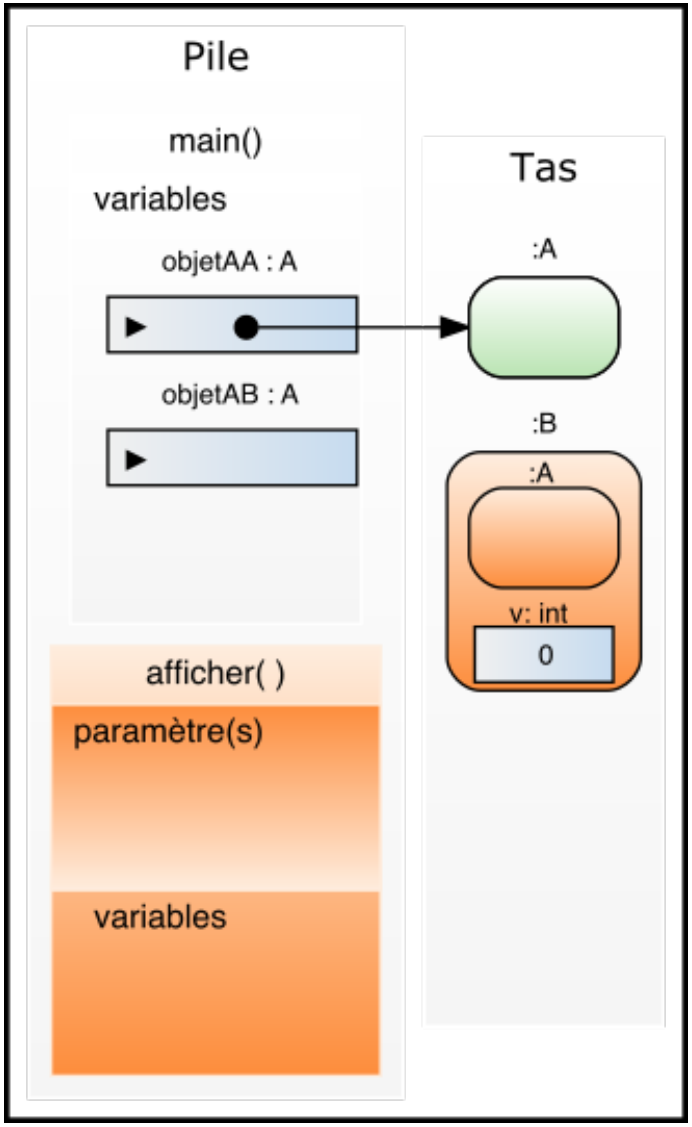



```
1  public class B extends A {
2      private int v;
3
4      public B() {
5          System.out.println("> Constructeur de B()");
6          v = 5;
7          afficher();
8      }
9
10      public void afficher()
11          System.out.println("> v = " + v);
12     }
13 }
```

– Appel à la méthode –

- `afficher()`
- sur l'instance courante de B...

En effet, l'instance de `B` a été créée. Il s'agit de l'instance courante, même dans le constructeur de `A`... Du coup, on appelle la méthode `afficher()` implémentée dans `B`.

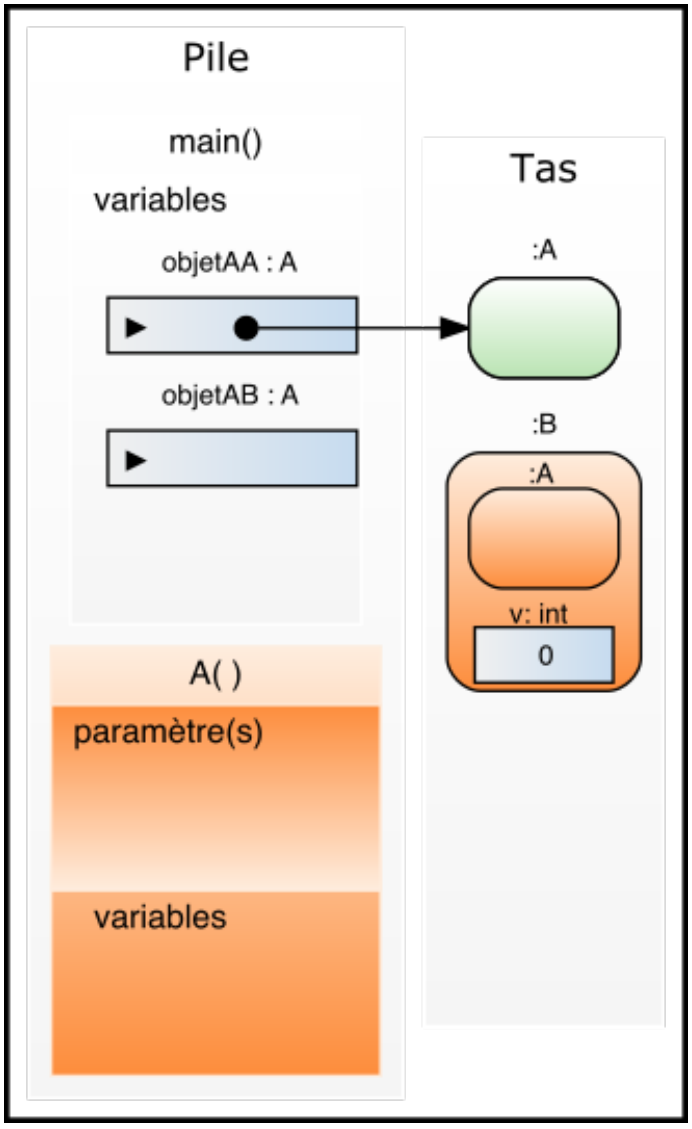



```
1  public class B extends A {
2    private int v;
3
4    public B() {
5          System.out.println("> Constructeur de B()");
6          v = 5;
7          afficher();
8    }
9
10   public void afficher() {
11         System.out.println("> v = " + v);
12   }
13 }
```

– On affiche –

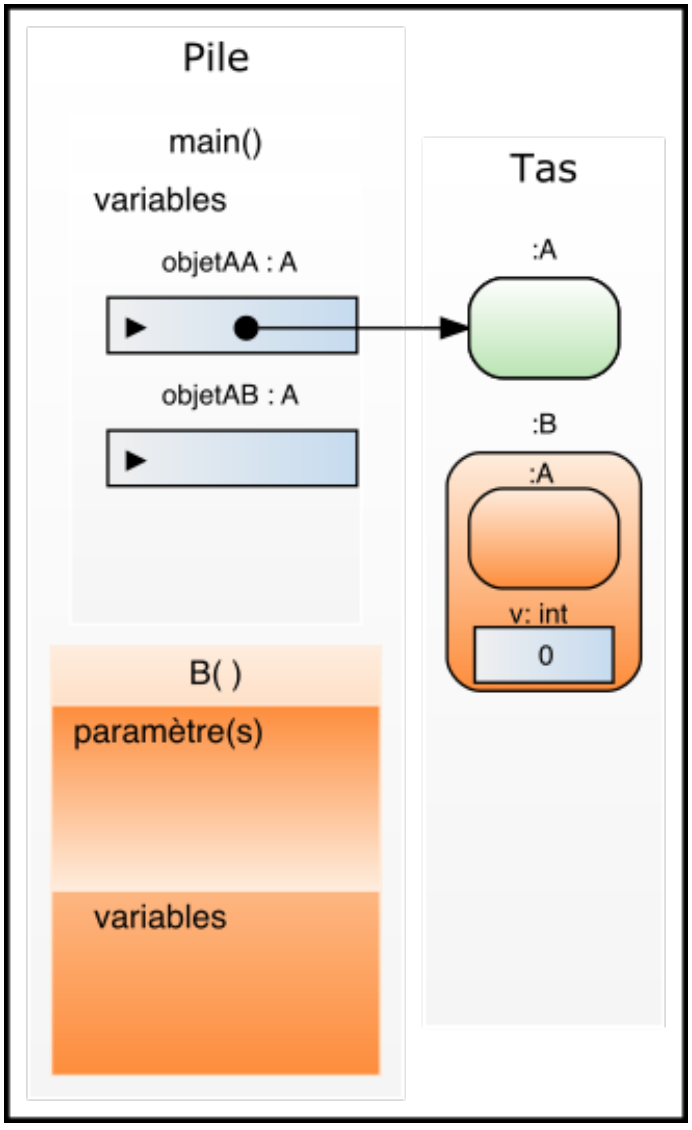
```
> A objetAA = new A();
> Constructeur de A()
> Affichage de A
> B objetBB = new B();
> Constructeur de A()
> v = 0
```

A ce stade, l'attribut `v` a été initialisé à 0 (valeur par défaut de int) lors de l'appel à l'opérateur `new`.




```
1  public class A {
2      public A() {
3          System.out.println("> Constructeur de A()");
4          afficher();
5        }
6
7      public void afficher() {
8          System.out.println("> Affichage de A");
9      }
10 }
```

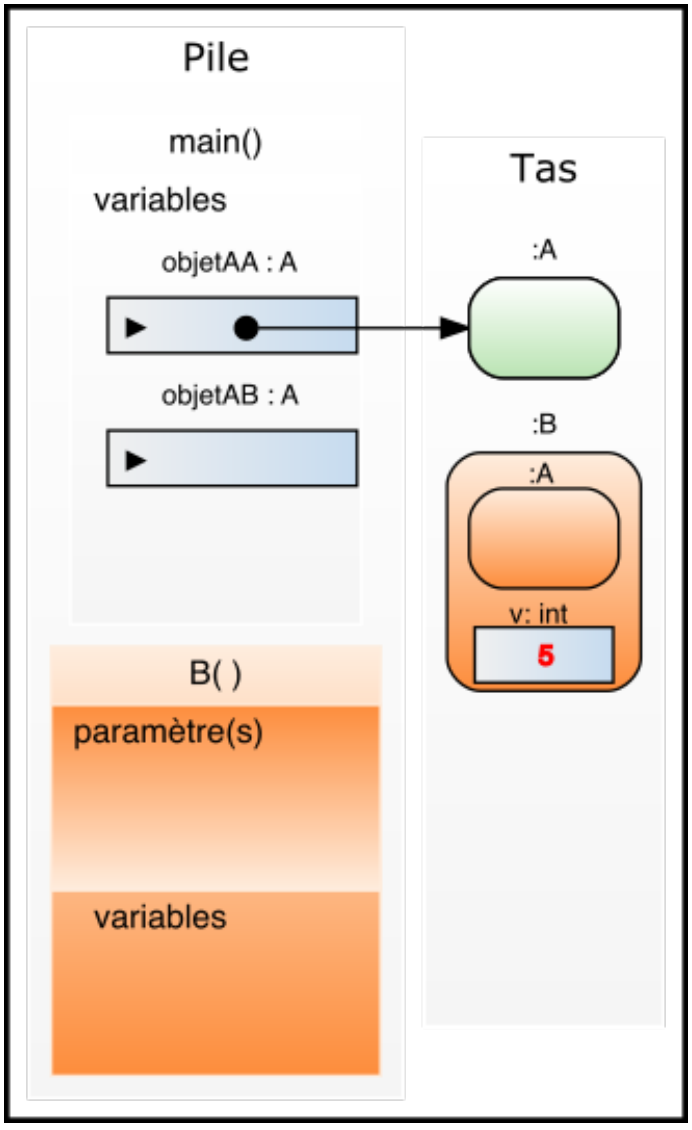
– Fin du constructeur de A –




```
1  public class B extends A {
2      private int v;
3      public B() {
4          System.out.println("> Constructeur de B()");
5          v = 5;
6          afficher();
7      }
8
9      public void afficher() {
10         System.out.println("> v = " + v);
11     }
12 }
```

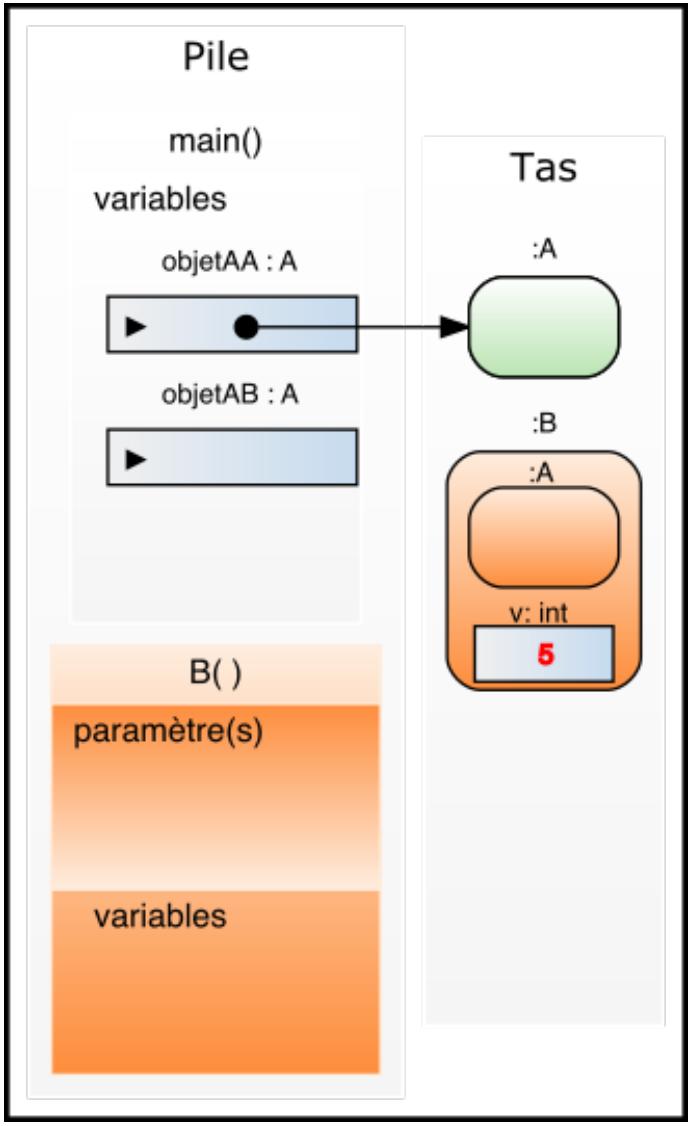
– On revient au constructeur de B –
– On affiche –

```
> A objetAA = new A();
> Constructeur de A()
> Affichage de A
> B objetBB = new B();
> Constructeur de A()
> v = 0
> Constructeur de B
```




```
1  public class B extends A {
2      private int v;
3      public B() {
4          System.out.println("> Constructeur de B()");
5           v = 5;
6          afficher();
7      }
8
9      public void afficher() {
10         System.out.println("> v = " + v);
11     }
12 }
```

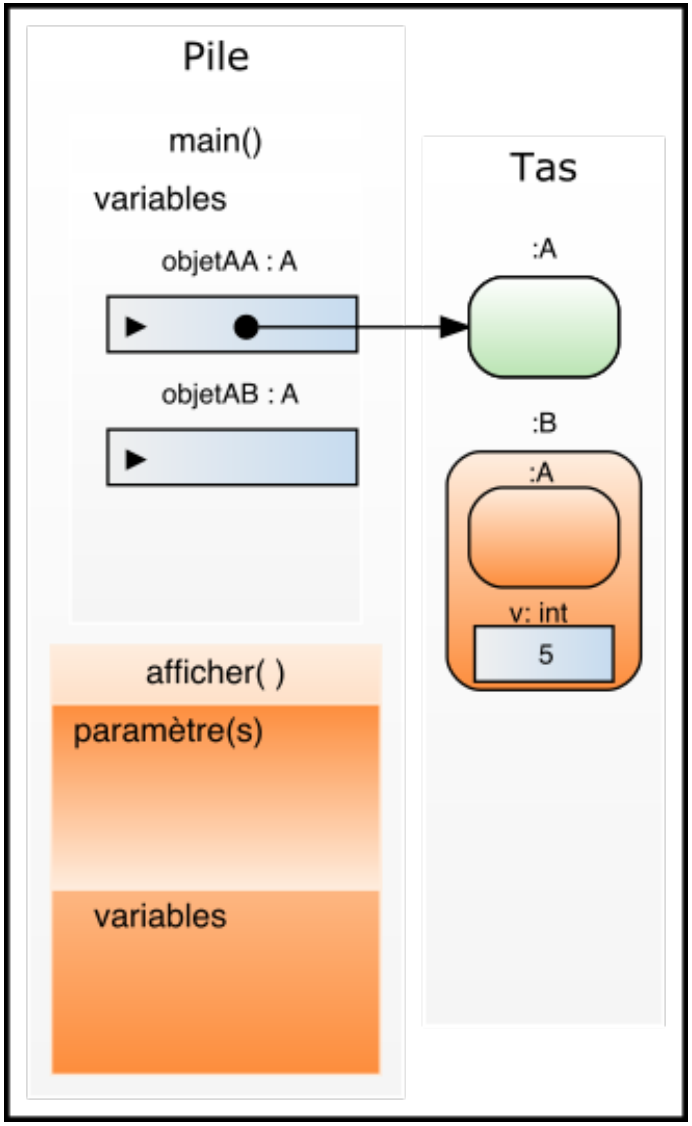
- l'attribut v
- prend la valeur 5.




```
1  public class B extends A {
2      private int v;
3      public B() {
4          System.out.println("> Constructeur de B()");
5          v = 5;
6           afficher();
7      }
8
9      public void afficher() {
10         System.out.println("> v = " + v);
11     }
12 }
```

– Appel à la méthode –

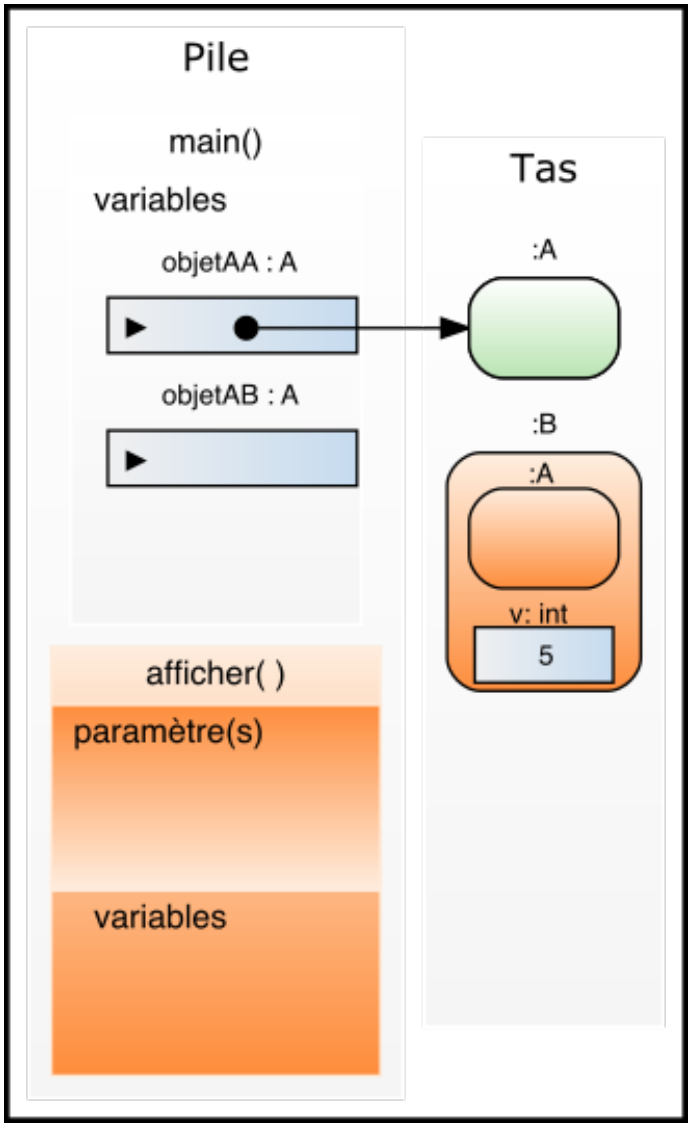
- `afficher()`
- sur l'instance courante de B...




```
1  public class B extends A {
2      private int v;
3
4      public B() {
5          System.out.println("> Constructeur de B()");
6          v = 5;
7          afficher();
8      }
9
10      public void afficher()
11          System.out.println("> v = " + v);
12     }
13 }
```

– Appel à la méthode –

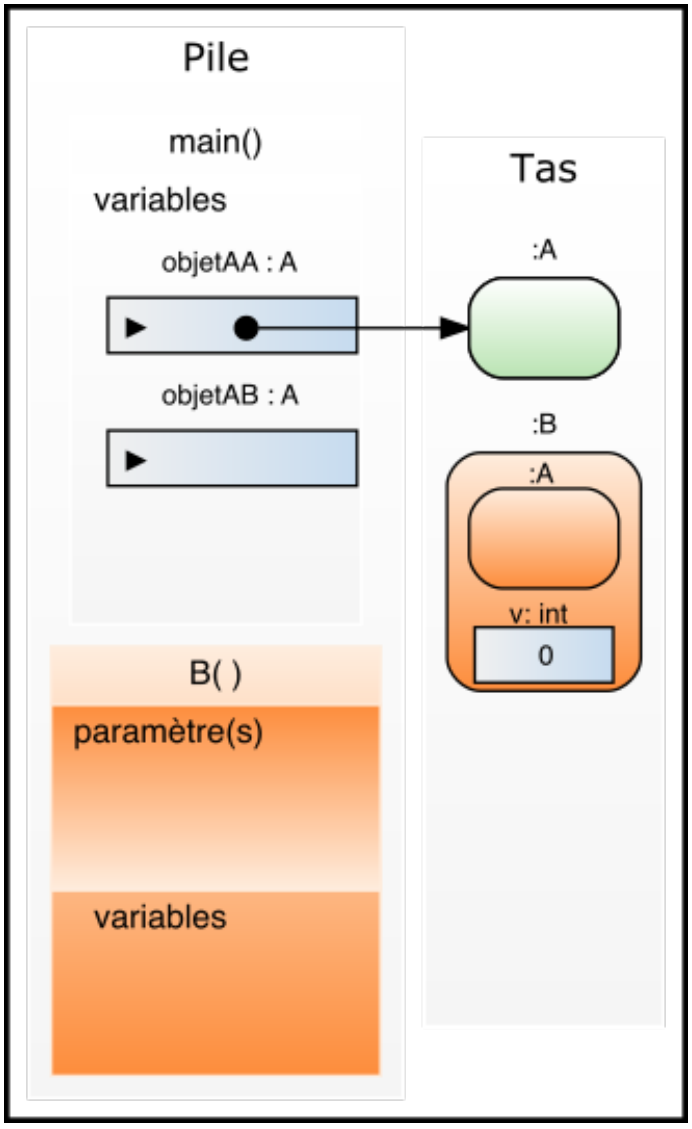
- `afficher()`
- sur l'instance courante de B...




```
1  public class B extends A {
2    private int v;
3
4    public B() {
5         System.out.println("> Constructeur de B()");
6         v = 5;
7         afficher();
8    }
9
10   public void afficher() {
11        System.out.println("> v = " + v);
12   }
13 }
```

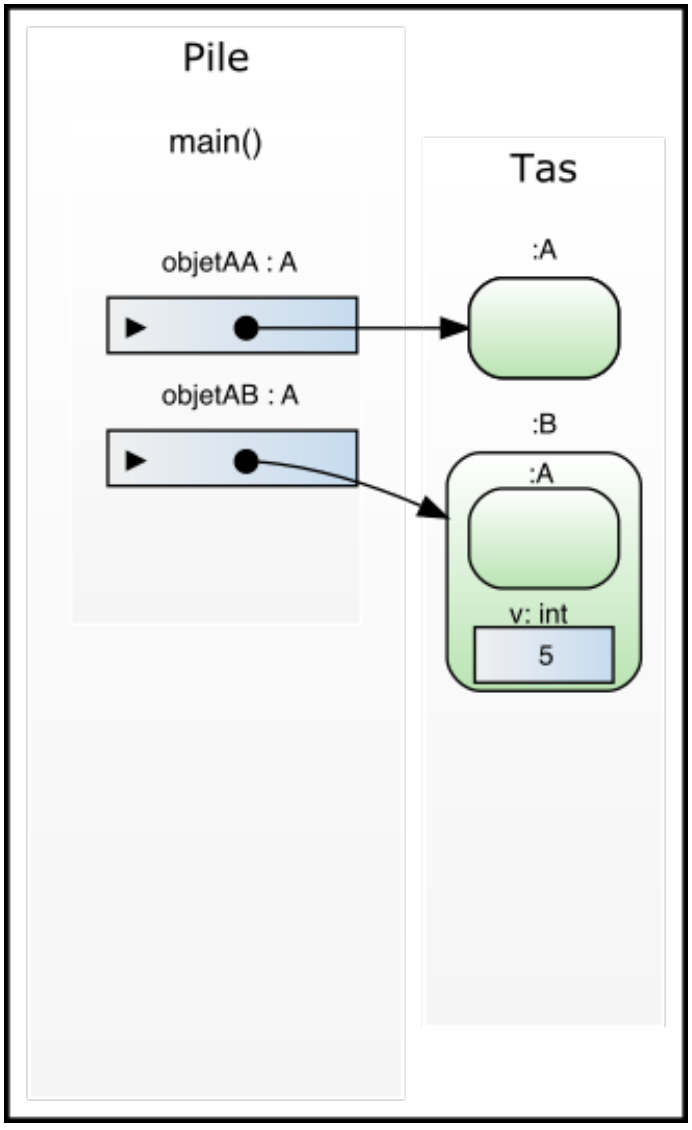
– On affiche –

```
> A objetAA = new A();
> Constructeur de A()
> Affichage de A
> B objetBB = new B();
> Constructeur de A()
> v = 0
> Constructeur de B
> v = 5
```




```
1  public class B extends A {
2      private int v;
3      public B() {
4          System.out.println("> Constructeur de B()");
5          v = 5;
6          afficher();
7    }
8
9      public void afficher() {
10         System.out.println("> v = " + v);
11     }
12 }
```

– Fin du constructeur de B –




```
1  public class Test {
2      public static void main(String[] args) {
3          System.out.println("A objetAA = new A();");
4          A objetAA = new A();
5          System.out.println("A objetAB = new B();");
6           A objetAB = new B();
7          System.out.println("B objetBA = new A();");
8          // B objetBA = new A();
9      }
10 }
```

– Etape 3 de l'instanciation –

1. Appel à l'opérateur `new`
2. Appel au constructeur du type `B`
3. Affectation de la référence

On peut noter qu'une référence vers `A` peut référencer une instance de `B` qui est aussi une instance de `A`.