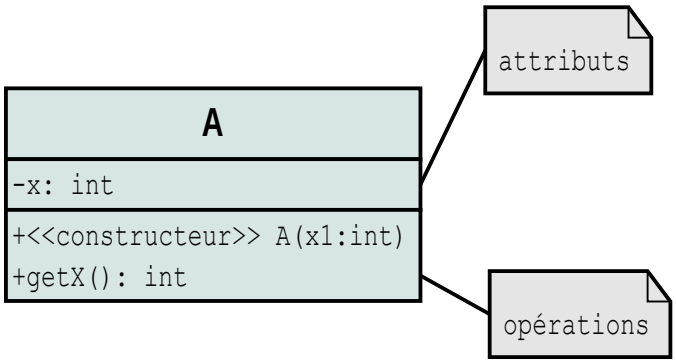
UML (rappels)

Classe

UML



java

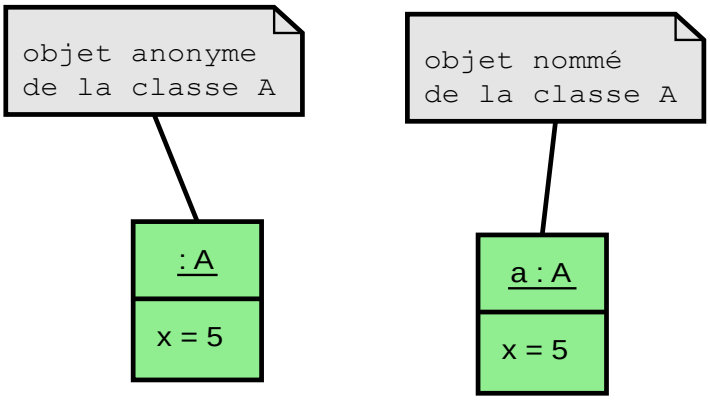
```
public class A
{
	private int x ;

	public A(int x1)
	{
		this.x = x1 ;
	}

	public int getX()
	{
		return this.x ;
	}
}
```

Objet

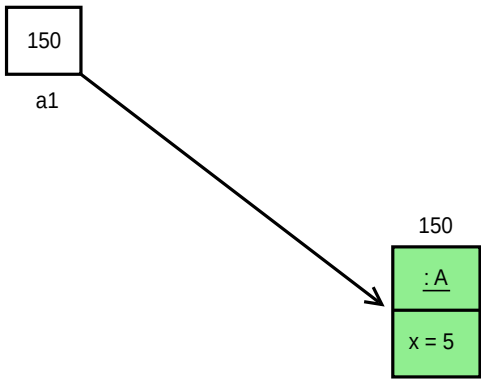
UML



java

```
{
        A a1 = new A(5)
}
```

Les objets créés par **new** sont anonymes, la variable *a1* contient la référence (l'adresse) de l'objet créé. Ici la machine virtuelle java a créé un objet anonyme de type *A* à la référence *150*, et affecté la variable *a1* avec cette référence.



Associations

Les associations écrites entre classes contraignent les objets de ces classes.

UML

Tout objet de la classe *A* est associé à 2 objets de la classe *B.*

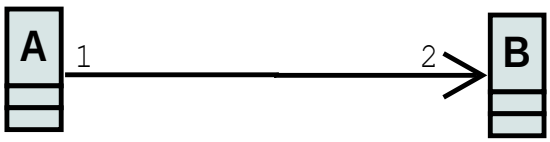
Tout objet de la classe *B* est associé à 1 objet de la classe *A*.

java

```
public class A
{
                        // association aux 2 objets B
        private B b1 ;
        private B  b2 ;
}
```

```
public class B
{
                        // associe à l'objet A
        private A a ;
}
```

Association unidirectionelle



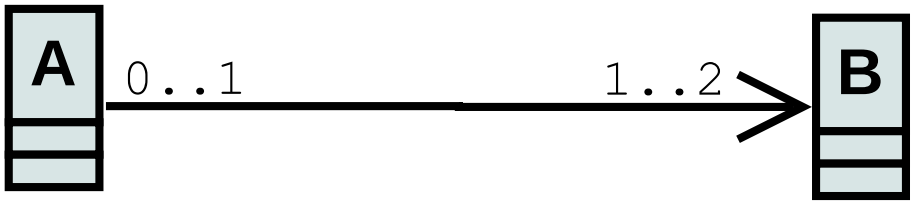
L'association est la même que précédemment mais d'un objet *A* on peut accéder aux 2 objets B associés, et d'un objet *B* on ne peut pas accéder à l'objet *A* associé

Seule la classe *B* change :

```
public class B
{
        // aucune variable vers un objet A
}
```

Multiplicité

La multiplicité précise le nombre d'objets associés



Tout objet *A* est associé a au moins 1 et au plus 2 objets *B*, tout objet *B* est associé à au plus 1 objet *A*.

java

```
public class A
{
                        // association aux 2 objets B
        private B b1 ;
        private B  b2 ;

                        // association avec 2 objets B
        public A(B x, B y)
        {
                this.b1 = x ;
                this.b2 = y ;
        }

                        // association avec 1 objet B
        public A(B x)
        {
                this.b1 = x ;
        }
}
```

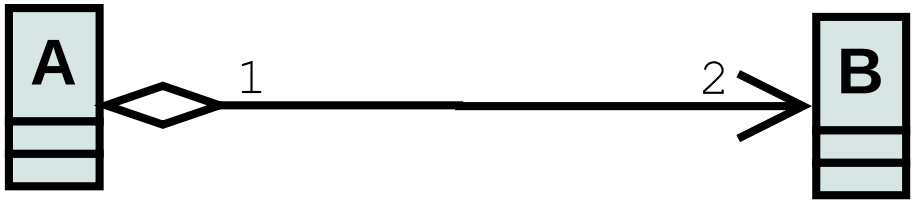
```
{               // exemple création d'association

                // association avec 2 objets B
        B x1 = new B() ;
        B y1 = new B() ;
        A a1 = new A(x1, y1) ;

                // association avec 1 objet B
        B x2 = new B() ;
        A a2 = new A(x2) ;
}
```

Agrégation

Indique simplement une notion de contenant/ensemble et contenu/élément.



tout objet A contient 2 objets B ou
tout objet A est formé de 2 objets B,
…

Aucune incidence sur le modèle, aucune traduction particulière en java.

Composition

Contrainte forte : un composite est fait de composants, et le tout est indissociable.

Tout objet *A* (composite) est composé de 2 objets *B* (composants),

1. si l'objet *A* (composite) est détruit les objets *B* (composants) le sont aussi,
2. un objet *B* ne peut être associé qu'a 1 composite *A*

java

Les deux contraintes doivent être assurées par le programmeur de la classes *A*.

1) si l'objet *A* est détruit les objets *B* le sont aussi => les objets *B* doivent être créés à l'intérieur de la classe *A* (voir cours java).

2) un objet *B* ne peut être associé qu'à 1 composite *A* => les objets *B* doivent être créés dans la classe *A*, le programmeur assurant la contrainte de non association d'un *B* avec un autre composite *A*.