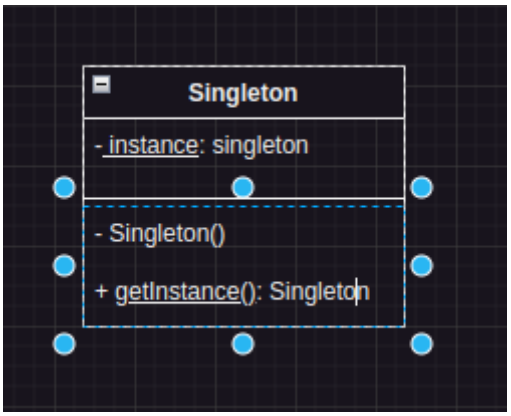
Patron de Construction : Singleton

modèle générique :



code :

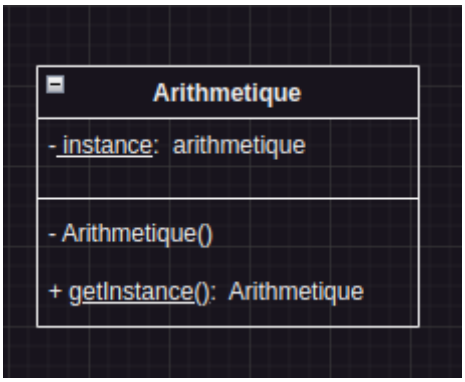
```java
public final class Singleton
{
private static Singleton instance = null;
// D'autres attributs, classiques et non "static".
private int x;
private int y;
// Constructeur de l'objet.
private Singleton()
{
// La présence d'un constructeur privé supprime le constructeur public par défaut.
  super();
}
private Singleton(int x, int y){
    this.x = x;
    this.y = y;
}
// Méthode renvoyant une instance de la classe Singleton
public static Singleton getInstance()
{
if (instance == null)
{
instance = new Singleton();
}
return instance;
}
public static Singleton getInstance(int x,int y)
{
if (instance == null)
{
instance = new Singleton(x,y);
}
return instance;
```

```
}
return instance;
}
// D'autres méthodes classiques et non "static".

public int somme(int x, int y)
{

  return x+y;
}
public float moyenne(int x, int y)
{
    return  somme(x ,y)/2;
}
public void affiche()
{
    System.out.println("\nJe suis une instance mes valeurs sont : x = " + this.x + " et y = " +this.y);
}
@Override
public Object clone() throws
CloneNotSupportedException {
throw new CloneNotSupportedException();
}
}
```

modèle générique :



code :

```
public final class Arithmetique {

    private static Arithmetique instance = null;

    private int x;
    private int y;
    private String z;

    private Arithmetique(){
        super();
    }

    private Arithmetique(int x, int y, String z){
        this.x = x;
        this.y = y;
        this.z = z;
    }

    public static Arithmetique getInstance() {
        if (instance == null) {
            instance = new Arithmetique();
        }
        return instance;
    }

   public static Arithmetique getInstance(int x, int y) {
      if (instance == null) {
         instance = new Arithmetique(x, y, z:"");
    }
    return instance;
}
```

```
    public int somme(int x, int y) {
        this.z = "somme";
        this.x = x;
        this.y = y;
        return x + y;
    }

    public int soustraction(int x, int y) {
        this.z = "soustraction";
        this.x = x;
        this.y = y;
        return x - y;
    }

 public int multiplication(int x, int y) {
    this.z = "multiplication";
    this.x = x;
    this.y = y;
    return x * y;
}

    public float division(int x, int y) {
        if(y == 0) {
            System.out.println("Division par zéro !");
            return Float.NaN;  // Not a Number
        }
        this.z = "division";
        this.x = x;
        this.y = y;
        return (float) x / y;
    }
```

```
    public void affiche() {
        System.out.println("Je suis une instance. Mes valeurs sont : x = " + this.x + ", y = " + this.y
    }
}
```

Code TestArithmetique :

```
public class TestArithmetique {
 public static void main(String[] args)
 {
 int som = Arithmetique.getInstance().somme(2, 5);
 System.out.printf("\nla somme est %d",som);
 float div = Arithmetique.getInstance().division(2, 5);
 System.out.printf("\nla division est %f",div);
 int soustraction = Arithmetique.getInstance().soustraction(8, 5);
 System.out.printf("\nla soustraction est %d",soustraction);
int multiplication = Arithmetique.getInstance().multiplication(8, 5);
 System.out.printf("\nla multiplication est %d\n",multiplication);
 Arithmetique s1 = Arithmetique.getInstance(8, 3);
 s1.affiche();
 Arithmetique s2 = Arithmetique.getInstance(5, 9);
 s2.affiche();
 }
}
```