



Travaux d'Etudes et de Recherches

**Développement d'un mini analyseur
statique de code intégré dans Eclipse**

Encadrants : MOY Matthieu
MONNIAUX David



Plan :

- Sujet
- Analyse Statique
- Interprétation abstraite
- Travail
- Structure outil
- Exemple outil



Sujet :

Développement d'un analyseur statique
pour un sous ensemble du langage Java
intégré dans Eclipse.

- Mini Java :
 - Une classe
 - Une méthode *main*.
 - Variables de type *int*.



Exemple de Programme Java :

```
public class Valeur_Absolue {  
  
    public static void main (String[] args){  
        int x , y = 0;  
  
        if (x <= 0){  
            x = -x;  
        }  
    }  
}
```

```
public class boucle {  
  
    public static void main (String[] args){  
        int x = 0;  
  
        while (x <= 100) {  
            x++;  
        }  
    }  
}
```

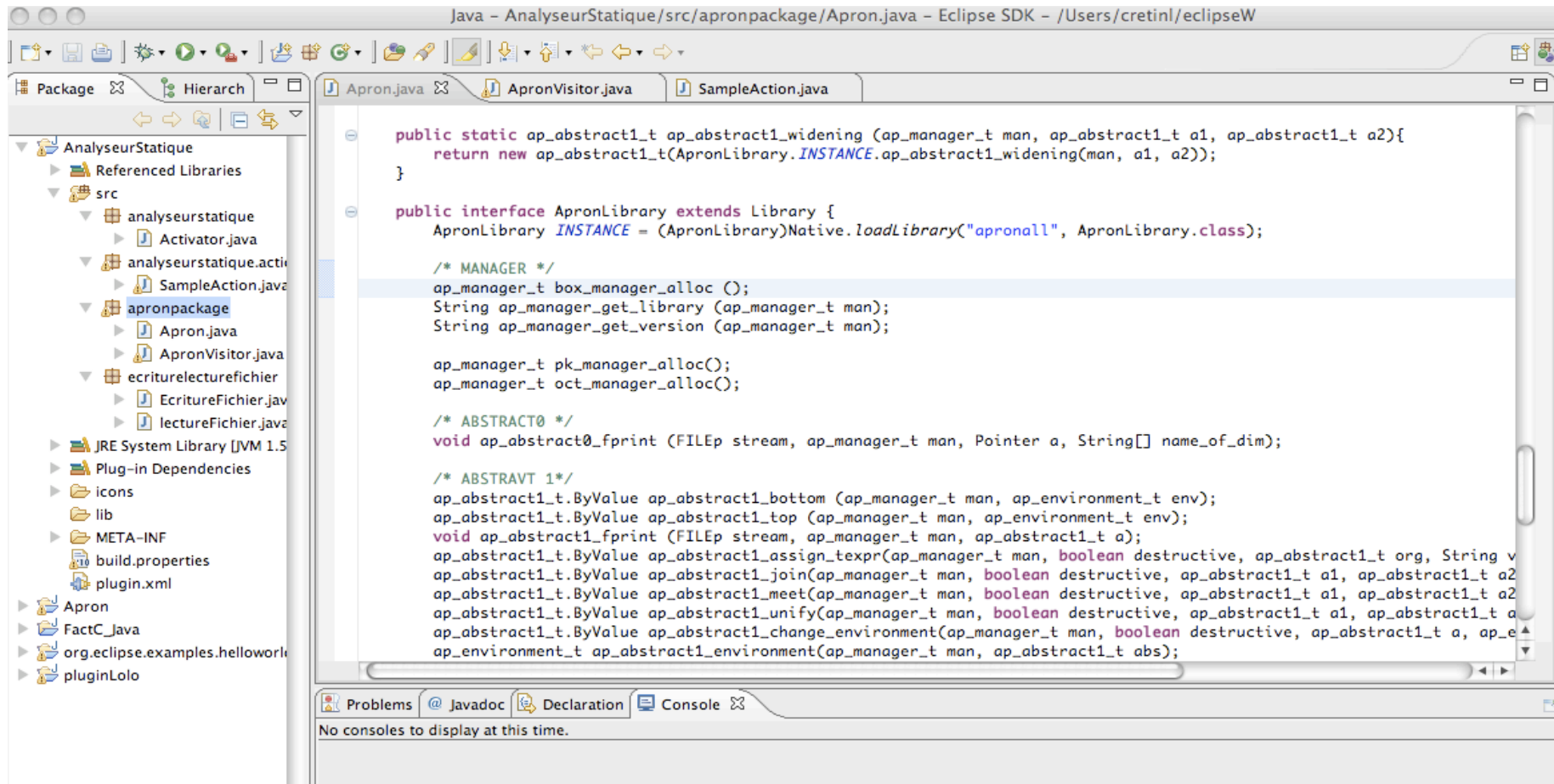


Sujet :

Développement d'un analyseur statique pour un sous ensemble du langage Java intégré dans Eclipse.

- Mini Java :
 - Une classe
 - Une méthode *main*.
 - Variables de type *int*.

- Eclipse :
 - Environnement de développement lancé par IBM.
 - Architecture développée autour de la notion de *plug-in*.
 - Obtention facile de l'arbre abstrait du programme java.



```
public static ap_abstract1_t ap_abstract1_widening (ap_manager_t man, ap_abstract1_t a1, ap_abstract1_t a2){
    return new ap_abstract1_t(ApronLibrary.INSTANCE.ap_abstract1_widening(man, a1, a2));
}

public interface ApronLibrary extends Library {
    ApronLibrary INSTANCE = (ApronLibrary)Native.loadLibrary("apronall", ApronLibrary.class);

    /* MANAGER */
    ap_manager_t box_manager_alloc ();
    String ap_manager_get_library (ap_manager_t man);
    String ap_manager_get_version (ap_manager_t man);

    ap_manager_t pk_manager_alloc();
    ap_manager_t oct_manager_alloc();

    /* ABSTRACT0 */
    void ap_abstract0_fprint (FILEp stream, ap_manager_t man, Pointer a, String[] name_of_dim);

    /* ABSTRACT1 */
    ap_abstract1_t.ByValue ap_abstract1_bottom (ap_manager_t man, ap_environment_t env);
    ap_abstract1_t.ByValue ap_abstract1_top (ap_manager_t man, ap_environment_t env);
    void ap_abstract1_fprint (FILEp stream, ap_manager_t man, ap_abstract1_t a);
    ap_abstract1_t.ByValue ap_abstract1_assign_texpr(ap_manager_t man, boolean destructive, ap_abstract1_t org, String v);
    ap_abstract1_t.ByValue ap_abstract1_join(ap_manager_t man, boolean destructive, ap_abstract1_t a1, ap_abstract1_t a2);
    ap_abstract1_t.ByValue ap_abstract1_meet(ap_manager_t man, boolean destructive, ap_abstract1_t a1, ap_abstract1_t a2);
    ap_abstract1_t.ByValue ap_abstract1_unify(ap_manager_t man, boolean destructive, ap_abstract1_t a1, ap_abstract1_t a2);
    ap_abstract1_t.ByValue ap_abstract1_change_environment(ap_manager_t man, boolean destructive, ap_abstract1_t a, ap_environment_t ap_environment_t ap_abstract1_environment(ap_manager_t man, ap_abstract1_t abs);
```



Analyse Statique

- Utilisé dans le monde industriel pour le développement de logiciel critique.
- Obtenir des résultats sur l'exécution du programme sans l'exécuter.
- Permet de :
 - Repérer des erreurs de programmation
 - Prouver formellement des propriétés

- Analyse Statique => Interprétation abstraite.



Interprétation abstraite

L'interprétation Abstraite est une théorie de la résolution approchée d'équations de point fixe appliquée à l'analyse de programmes.

- Permet de prouver des propriétés de manières automatiques.
- Découverte d'invariants.
- Principe : Définir un domaine abstrait où l'on exécutera le programme. Par exemple chaque ensemble de valeurs en tout point du programme.



Travail

- Interprétation abstraite de domaines numériques :
 - Calculer, en chaque point du programme, les intervalles dans lesquels se trouvent les variables.

- Un “vrai” langage mais peu de temps.
- Choix techniques :
 - Analyse Arrière / Analyse avant
 - Graphe de flot de contrôle / Arbre Abstrait
 - Elargissement



Travail

- Interprétation abstraite de domaines numériques :
Calculer, en chaque point du programme, les intervalles dans lesquels se trouvent les variables.
- Librairie APRON pour les domaines abstraits (v0.9.8).
 - Expressions
 - Séparation domaine abstrait et analyseur
 - Générique
 - Intervalles : BOX
 - Octogones : OCT
 - Polyhèdres convexes : NEWPOLKA



Travail

- Interprétation abstraite de domaines numériques :
Calculer, en chaque point du programme, les intervalles dans lesquels se trouvent les variables.
- Java Native Access pour l'appel de fonctions C en java.
 - Accès facile à des bibliothèques partagées.
 - Accès dynamique au moment de l'exécution.
 - Interface Java pour décrire les fonctions et les structures C.



Travail

➤ postCondition **Analyser_IF**(predCondition, Arbre) {

 resIF = **Analyser_Arbre** (Condition *Inter* predCondition, Branche_THEN);

 resELSE = **Analyser_Arbre**(!Condition *Inter* predCondition, Branche_ELSE);

 return resIF Union resELSE;

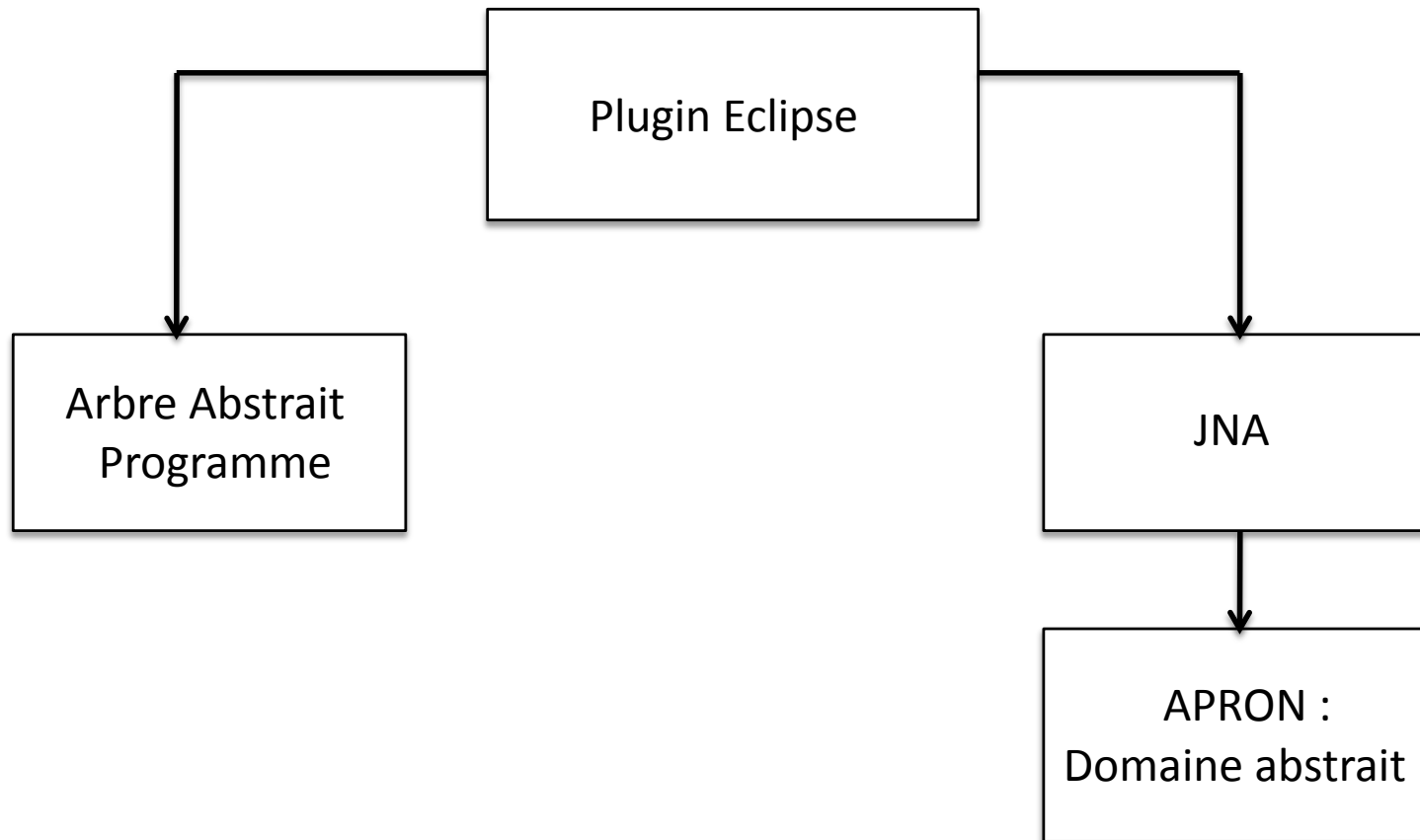
}



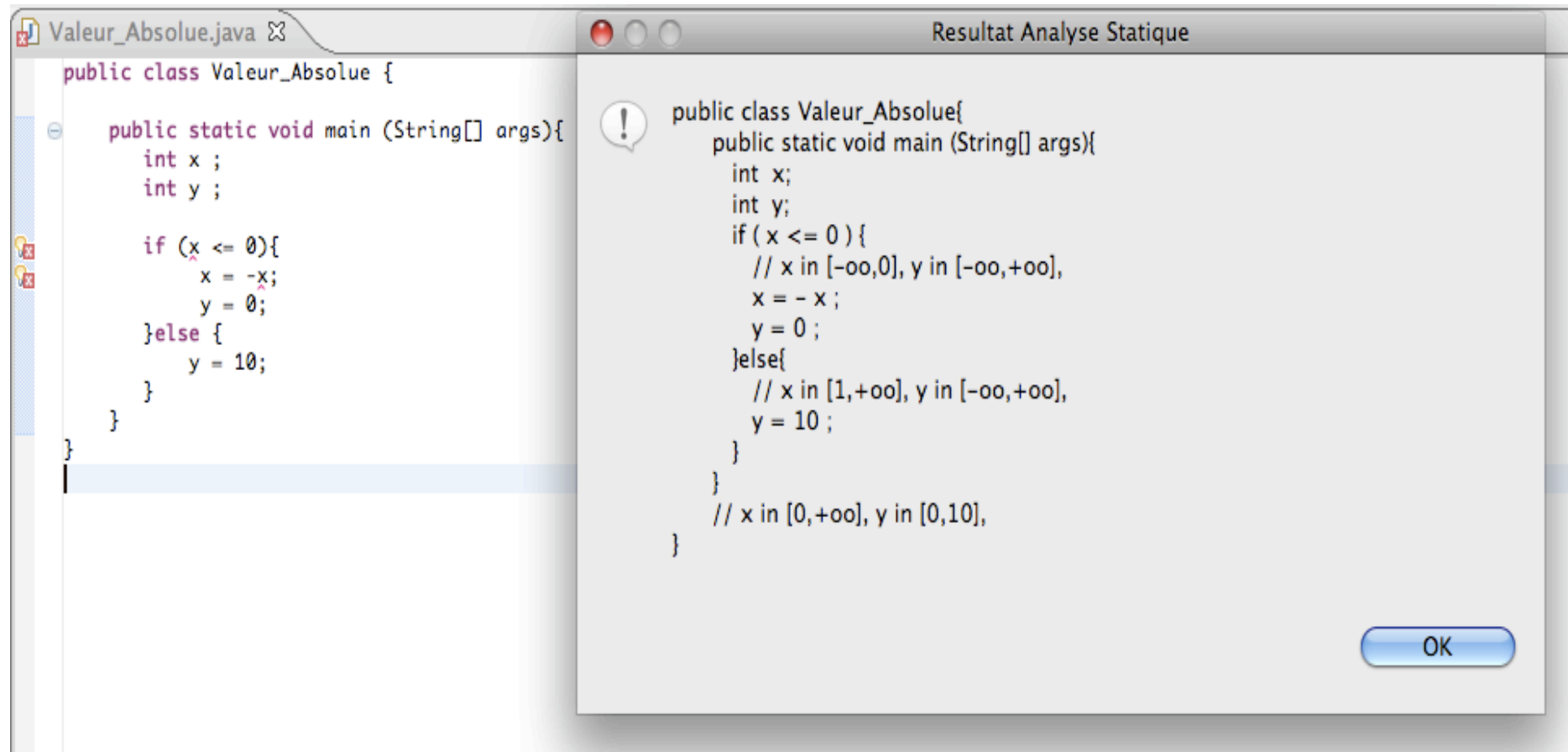
Travail

```
➤ postCondition Analyser_WHILE(predCondition, Arbre) {  
  x = predCondition;  
  do {  
    tmp = Analyser_Arbre (Condition Inter x, Branche_BODY);  
    y = predCondition Union tmp;  
    x = Widening(predCondition, y);  
  } while (!Inclu(y,x));  
  // rétrécissement  
  x = Analyser_Arbre (Condition Inter x, Branche_BODY);  
  y = predCondition Union tmp;  
  // sortie de la boucle : filtrage  
  return y Inter !Condition;  
}
```

Structure outil



Exemple : If (c) then ... else ...



The image shows a screenshot of an IDE with two windows. The left window, titled 'Valeur_Absolue.java', contains the following Java code:

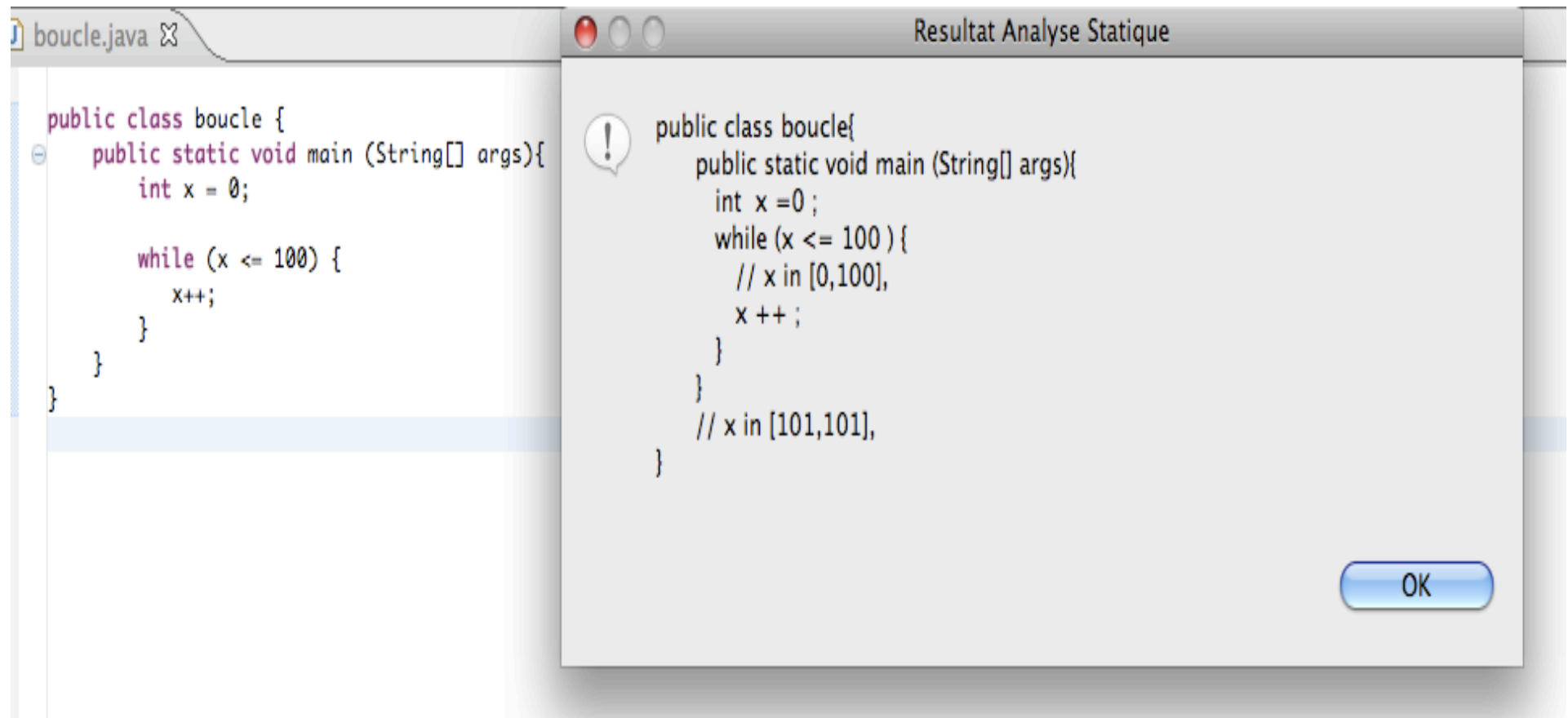
```
public class Valeur_Absolue {  
    public static void main (String[] args){  
        int x ;  
        int y ;  
  
        if (x <= 0){  
            x = -x;  
            y = 0;  
        }else {  
            y = 10;  
        }  
    }  
}
```

The right window, titled 'Resultat Analyse Statique', displays the static analysis result for the same code, including a warning icon:

```
public class Valeur_Absolue{  
    public static void main (String[] args){  
        int x;  
        int y;  
        if ( x <= 0 ) {  
            // x in [-oo,0], y in [-oo,+oo],  
            x = - x ;  
            y = 0 ;  
        }else{  
            // x in [1,+oo], y in [-oo,+oo],  
            y = 10 ;  
        }  
    }  
    // x in [0,+oo], y in [0,10],  
}
```

An 'OK' button is visible at the bottom right of the static analysis window.

Exemple : while (c) ...



The image shows a code editor window on the left and a static analysis tool window on the right. The code editor window, titled 'boucle.java', contains the following Java code:

```
public class boucle {  
    public static void main (String[] args){  
        int x = 0;  
  
        while (x <= 100) {  
            x++;  
        }  
    }  
}
```

The static analysis tool window, titled 'Resultat Analyse Statique', displays the same code with annotations. A warning icon is shown next to the first line of the code. The annotations are as follows:

```
public class boucle{  
    public static void main (String[] args){  
        int x =0 ;  
        while (x <= 100 ) {  
            // x in [0,100],  
            x ++ ;  
        }  
    }  
    // x in [101,101],  
}
```

An 'OK' button is visible in the bottom right corner of the static analysis tool window.



Merci de votre attention