# TP6 : spécification et validation de l'algorithme de recherche dichotomique

## Installation

**1.** Ouvrez une "fenêtre de commandes" Windows (onglet Démarrer, puis Exécuter "cmd"). Dans la fenêtre de commandes, tapez la commande suivante:

```
set Path=%Path%;\\info\Bibliotheque\MF-dev\TP-EscJava\windows\escjava\bin
```

**2.** Placez-vous dans le répertoire `<REPERTOIRE_TP>\Exercices\4.tableauxTries`

```
> CD <REPERTOIRE_TP>\Exercices\4.tableauxTries
```

Pour utiliser ESC/Java, tapez la commande (Windows) suivante:

```
> escjava -loopSafe RechercheDichotomique.java
```

### Méthode `estTrie`

Copiez la spécification et le code de la méthode `estTrie` de `Exercices\4.TableauxTries\TableauTrie`. Le code doit compiler sans erreur avec `javac` et ESC/Java doit le vérifier.

### Méthode `rechercheLineaire`

Copiez la spécification et le code de la méthode `rechercheLineaire` de `Exercices\4.TableauxTries\TableauTrie`. Le code doit compiler sans erreur avec `javac` et ESC/Java doit le vérifier.

**Rappel sur la spécification informelle d'algorithme de recherche** dans un tableau d'entiers.

Si l'indice retourné par l'algorithme de recherche, nommé `i`, est strictement inférieur à `zone.length` alors le contenu de `zone[i]` est égale à `val`. Si l'indice retourné est `zone.length` alors la valeur de `val` n'est pas contenu dans le tableau.

### Méthode `rechercheDichotomique`

1. Reprenez la spécification de `rechercheLineaire` (directives `ensures`).
2. Ajoutez des pré-conditions (directive `requires`).
3. Validez votre code : aucune mise en garde ne doit être générée par `escjava`.

### Méthode `recherche`

4. Reprenez la spécification de `rechercheLineaire` (directives `ensures`).
5. Validez votre code : aucune mise en garde ne doit être générée par `escjava`.

```
public class RechercheDichotomique {
   private /*@ spec_public */ /*@ non_null */ int[] zone;

   private boolean estTrie () {return true;}

   public int rechercheLineaire(int val){ return 0;}

     /*@
           requires  indDebut >=0;
           requires indFin <= zone.length-1;
     */
    private int rechercheDichotomique(int val, int indDebut, int indFin){
     if (indDebut > indFin) {
           return zone.length;
     }
     else {
           int pivot = indDebut+((indFin - indDebut) / 2);
           if (val == zone[pivot]) return pivot;
           else{
                 if (val < zone[pivot]) {
                       return rechercheDichotomique(val, indDebut, pivot-1);
                 }
                 else {
                       return rechercheDichotomique(val,  pivot+1, indFin);
                       }
           }
     }
   }

   public int recherche(int val){
     if (! estTrie()) return rechercheLineaire(val);
            return rechercheDichotomique(val,  0, zone.length-1);
    }

  public  RechercheDichotomique() {zone = new int [0];}

}
```