2. Opérations abstraites sur des intervalles

Une fois implémentées les classes `Bound` et `Interval`, vous êtes en mesure d'implémenter les opérations dont a besoin l'analyseur. **Le but des séances suivantes** de TP est d'implémenter ces opérations sur les intervalles. Le seul fichier à modifier est `IntervalLattice.java`.

Classes fournies

On fournit une classe `IntervalPair` qui représente un couple d'intervalles.

On fournit aussi la classe `IntervalVector` qui représente un vecteur d'intervalles. Cette classe est utilisée par l'analyseur : un vecteur d'intervalles est attaché à chaque point de contrôle. Un élément du vecteur représente l'intervalle abstrait d'une variable donnée.

Si n est la taille du vecteur, le vecteur peut prendre une valeur quelconque dans le produit cartésien du treillis des intervalles $\mathbb{I}^n$, privé de certaines valeurs. En effet, dès qu'une valeur porte l'intervalle `BOT` (l'état correspondant est inaccessible) toutes les autres valeurs du vecteur sont aussi à `BOT`.

Opérations sur les intervalles

La classe `IntervalLattice` est une classe outil, elle ne contient que des méthodes **static**. Ces méthodes représentent toutes les opérations abstraites dont peut avoir besoin l'analyseur.

▷ **Question 1 :** Compléter les méthodes de la classe `IntervalLattice`. Testez-les abondamment !

Pour chaque méthode, réfléchir sur des exemples, puis raisonner en écrivant les inéquations.

Exemple pour la méthode `plus` : on calcule le résultat d'une transition du graphe de contrôle correspondant à l'instruction `z := x + y;` Si $a \leq x \leq b$ et si $c \leq y \leq d$, alors $a + c \leq x + y \leq b + d$. En d'autres termes, $\text{Post}_{z:=x+y}(..., x \mapsto [a, b], y \mapsto [c, d], ...) = (..., z \mapsto [a + c, b + d], ...)$.
Ceci est vrai si a, b, c, d sont des entiers, qu'en est-il si certains d'entre eux valent $+\infty$ ou $-\infty$?

Focaliser vos efforts sur les méthodes qui permettront d'analyser le programme suivant :

```
program boucle
  x, y : integer;
begin
  x := 0; y := 4;
  while (x < y) loop
     x := x + 1;
  endloop;
end
```

▷ **Question 2 :** Pour vérifier vos calculs pas à pas, mettez en place des traces qui affichent, à chaque fois qu'une méthode d'`IntervalLattice` est appelée, son nom, ses paramètres et le résultat calculé.

▷ **Question 3 :** Compléter les autres méthodes, en testant à chaque fois sur un petit programme.