ON EST EN OOP : ON A DU POLYMORPHISME PAR HÉRITAGE

LINK TO THE PATH

```
interface Direction{
    public int toInt();
    public String toString();
};
class North implements Direction{
     public int toInt(){ return 1;};
     public String toString(){ return "north";};
};
class East implements Direction{
     public int toInt(){ return 2;};
     public String toString(){ return "east";};
};
class South implements Direction{
     public int toInt(){ return 3;};
     public String toString(){ return "south";};
};
class West implements Direction{
     public int toInt(){ return 4;};
     public String toString(){ return "west";};
};

public class Main {
    public static final String label(Direction d) {
        return d.toString();
    }

    public static void main(String args[]) {
     System.out.println(label(new East()));
     System.out.println(label(new West()));
    }
}
```

MIAGE M2 - QUALITÉ DU SI - THOMAS HAESSLÉ

Le polymorphisme par héritage donne une garantie sur le fait de toujours avoir une implémentation de toString

Il ne fournit aucune garantie sur l'exhaustivité du domaine : on pourrait ajouter une classe NordEast par exemple ! Ou TripleBackflip !

Connaitre l'ensemble des implémentations peut être complexe (impossible avant Java 17)

LINK TO THE PATH

ON EST EN OOP : ON A DU POLYMORPHISME PAR HÉRITAGE

Le polymorphisme par héritage donne une garantie sur le fait de toujours avoir une implémentation de toString

Il ne fournit aucune garantie sur l'exhaustivité du domaine : on pourrait ajouter une classe NordEast par exemple ! Ou TripleBackflip !

Connaitre l'ensemble des implémentations peut être complexe (impossible avant Java 17)

```
interface Direction{
    public int toInt();
    public String toString();
};
class North implements Direction{
     public int toInt(){ return 1;};
     public String toString(){ return "north";};
};
class East implements Direction{
     public int toInt(){ return 2;};
     public String toString(){ return "east";};
};
class South implements Direction{
     public int toInt(){ return 3;};
     public String toString(){ return "south";};
};
class West implements Direction{
     public int toInt(){ return 4;};
     public String toString(){ return "west";};
};

public class Main {
    public static final String label(Direction d) {
        return d.toString();
    }

    public static void main(String args[]) {
     System.out.println(label(new East()));
     System.out.println(label(new West()));
    }
}
```

LINK TO THE PATH

QUE SE PASSE-T-IL SI ON RÉCUPÈRE LES DIRECTIONS D'UNE LIB

```
/**  lib tierce */
interface Direction{
    public int toInt();
};
class North implements Direction{
     public int toInt(){ return 1;};
};
class East implements Direction{
     public int toInt(){ return 2;};
};
class South implements Direction{
     public int toInt(){ return 3;};
};
class West implements Direction{
     public int toInt(){ return 4;};
};

/**  notre code */
interface DirectionImprove{
    public String toString();
};
class NorthImprove extends North implements DirectionImprove{
     public String toString(){ return "north";};
};
class EastImprove extends North implements DirectionImprove{
     public String toString(){ return "east";};
};
class SouthImprove extends North implements DirectionImprove{
     public String toString(){ return "south";};
};
class WestImprove extends North implements DirectionImprove{
     public String toString(){ return "west";};
};
```