TD8 : Visitor

Exercice 1 - Visiteur

Voici un exemple d'un petit évaluateur et simplificateur d'expression arithmétique [visitor0.zip](http://igm.univ-mlv.fr/ens/IR/IR2/2011-2012/PrOO/visitor0.zip) (<http://igm.univ-mlv.fr/ens/IR/IR2/2011-2012/PrOO/visitor0.zip>). On cherche à ajouter un mécanisme de visiteur au code existant.

1/ Rappelez ce qu'est un visiteur, à quoi il sert et dans quel cas on l'utilise.

2/ On souhaite écrire un visiteur `SimplifyVisitor` effectuant la même opération que la méthode `simplify` de l'interface `Expr`.

```
public class SimplifyVisitor {
    public Expr visit(Variable variable) {
        ...
    }
    public Expr visit(Constant constant) {
        ...
    }
    public Expr visit(Plus plus) {
        ...
    }
    public Expr visit(Minus minus) {
        ...
    }
    public Expr visit(Star star) {
        ...
    }
    public Expr visit(Slash slash) {
        ...
    }
}
```

Le visiteur sera appelé sur chaque noeud de l'arbre afin de simplifier l'expression courante et de renvoyer l'expression simplifiée.

Dans le `main` on demandera alors la simplification de l'expression de cette façon :

```
Expr expr2=expr.accept(new SimplifyVisitor());
```

2.1/ Écrivez le code des méthodes `accept`.

2.2/ Écrivez le code de la classe `SimplifyVisitor`.

3/ On souhaite maintenant utiliser le même mécanisme de visiteur pour écrire un visiteur `ToStringVisitor` effectuant l'équivalent de la méthode `toString()` (que l'on n'utilisera donc pas).

Pour cela, nous allons créer une interface `Visitor` que les classes `SimplifyVisitor` et `ToStringVisitor` implanteront.

Pensez au generics !

3.1/ Écrivez le code de l'interface `Visitor`, changez le code des méthodes `accept` et de la classe `SimplifyVisitor` en conséquence.

3.2/ Écrivez le code de la classe `ToStringVisitor`.

3.3/ Ecrivez une variante `ToStringPrefixVisitor` qui donnera une version postfixe.

4/ On souhaite enfin écrire le visiteur correspondant à l'opération `eval`. Écrivez le code de la classe `EvalVisitor` et modifiez le code en conséquence.

Attention à la déclaration des exceptions !

5/ On doit ajouter un opérateur manquant : `Power`, pour calculer x^y

5.1/ Listez les modifications nécessaires

5.2/ Faites ces modifications.

5.3/ Que se serait-il passé si vous aviez déclaré une méthode "`... visit(BinaryOp)...`" dans vos interfaces `Visitor`.