Evaluator implementation

```
abstract class Expr {
  abstract int value();
}
abstract class BinExpr extends Expr { }
class Add extends BinExpr {
  int value() {
    return getLeft().value() + getRight().value();
  }
}
class Sub extends BinExpr {
 ...
}
class IntExpr extends Expr {
  int value() {
    return Integer.parseInt(getIntExpr().getINT());
  }
}
```

The evaluator as a visitor

```
class Evaluator implements Visitor {
  Object visit(Add node, Object data) {
    int i1 = (Integer) node.expr1.accept(this, data);
    int i2 = (Integer) node.expr2.accept(this, data);
    return new Integer(i1 + i2);
  }

  static Object value(Expr expr) {
    Evaluator evaluator = new Evaluator();
    return expr.accept(evaluator);
  }
}
```

The unparser as a visitor

```
class Unparser implements Visitor {
  PrintStream stream;
  Object visit(Add node, Object data) {
    node.expr1.accept(this, data);
    stream.print('+');
    node.expr2.accept(this, data);
    return null;
  }
}

  static void unparse(Expr expr, PrintStream stream) {
    Unparser unparser = new Unparser(stream);
    expr.accept(unparser);
  }
```

CountIdentifiers as a visitor

```
class CountIdentifiers extends TraversingVisitor {
  int count = 0;
  Object visit(IdExpr node, Object data) {
    count++;
    return null;
  }
}
```