Interpreters architecture

Common cases

- Complex configuration files (layout.xml in Android)
- Network communication (json, xml, custom strings...)
- User input
- Domain Specific Languages

Examples

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/androi
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:orientation="vertical"
        android:gravity="center_horizontal" >

        <Button
            android:id="@+id/registerBtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Register"
            android:textSize="24sp"
            android:layout_margin="16dp" />

        <Button
            android:id="@+id/unregisterBtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Unregister"
            android:textSize="24sp"
            android:layout_margin="16dp" />


    </LinearLayout>

</RelativeLayout>
```

```yaml
name: flutter_app_tests
description: A new Flutter application.

publish_to: 'none'

version: 1.0.0+1

environment:
  sdk: "⩾2.7.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.0
  provider: ^4.3.2+2

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_driver:
    sdk: flutter
  test: ^1.15.4

flutter:
  uses-material-design: true
```

```json
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    {
      "name": "Molecule Man",
      "age": 29,
      "secretIdentity": "Dan Jukes",
      "powers": [
        "Radiation resistance",
        "Turning tiny",
        "Radiation blast"
      ]
    },
    {
      "name": "Madame Uppercut",
      "age": 39,
      "secretIdentity": "Jane Wilson",
      "powers": [
        "Million tonne punch",
        "Damage resistance",
        "Superhuman reflexes"
      ]
    },
    {
      "name": "Eternal Flame",
      "age": 1000000,
      "secretIdentity": "Unknown",
      "powers": [
        "Immortality",
        "Heat Immunity",
        "Inferno",
        "Teleportation",
        "Interdimensional travel"
      ]
    }
  ]
}
```

Problem Analysis

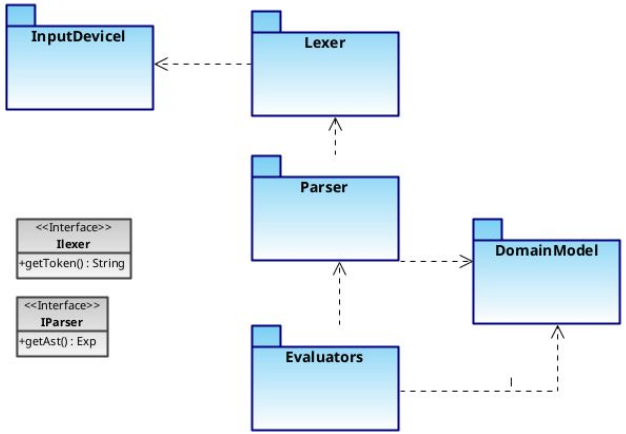
- In all this cases we have the set of admitted characters / words (tokens)
- Tokens forms the phrases/blocks of certain structure(syntax).
- The meaning of the phrases has to be valid (semantic).

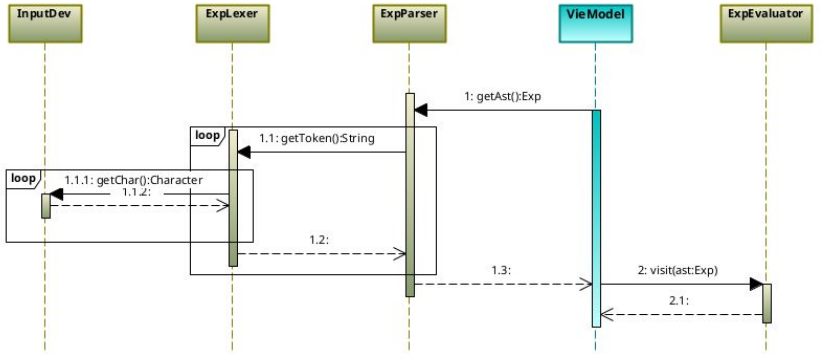
Case Study - Arithmetic expression

Design Interpreter for arithmetic expressions that should :

- Calculate the result
- Show APT in UI
- Write compiler for stack CPU
- Represent on xml yaml json
- .....

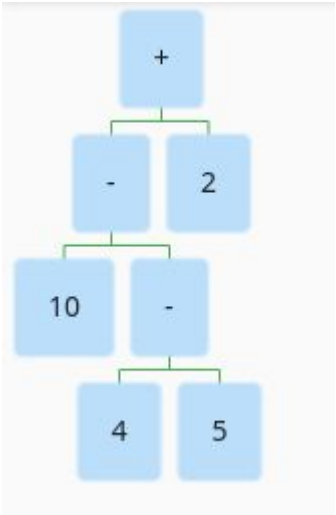
Logic Architecture





Gramar, Tree structure, Data Model

10-(4-5)+2



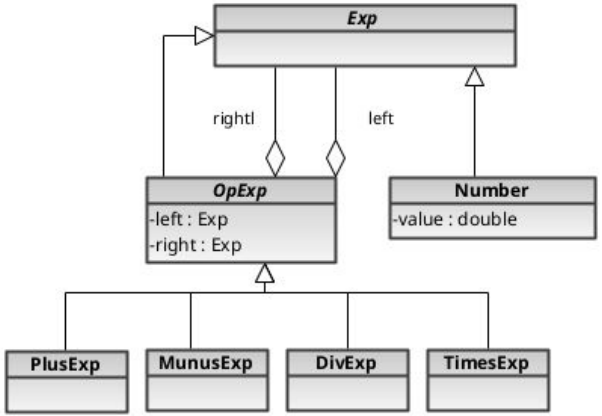
EXP ::= EXP + EXP // plusexp

EXP ::= EXP - EXP // minusexp

EXP ::= EXP * EXP // timesexp

EXP ::= EXP / EXP // divexp

EXP ::= num // numexp




```
PlusExp(MinusExp(NumExp(10),MinusExp(NumExp(4),NumExp(5))),NumExp(2));
```

Interpreter Functional vs OOP approach

Functional

```
public static int eval(Exp e) {
if (e instanceof NumExp) return ((NumExp)e).getValue();
else
if (e instanceof PlusExp)
return eval(((PlusExp)e).left) + eval(((PlusExp)e).right);
else
...
}
```

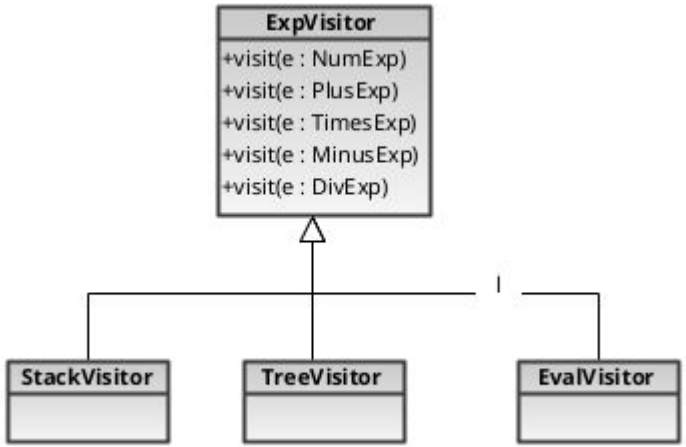
OOP

```
abstract class Exp {
  String toString();
  void accept(ExpVisitor visitor) => visitor.visit(this);
}

abstract class OpExp extends Exp {
  final Exp left, right;
  OpExp(this.left, this.right);
  String myOp();
  @override
  String toString() => '$left ${myOp()} $right';
}

class MinusExp extends OpExp {
  MinusExp(Exp left, Exp right) : super(left, right);
  @override
  String myOp() => "-";
}
```

Visitor pattern




```
class EvalExpVisitor extends ExpVisitor {
  double value = 0.0;

  double getEvaluation() => value;

  @override
  void visitDiv(DivExp e) => eval(e, (l, r) => value = l / r);

  @override
  void visitMinus(MinusExp e) => eval(e, (l, r) => value = l - r);

  @override
  void visitPlus(PlusExp e) => eval(e, (l, r) => value = l + r);

  @override
  void visitTimes(TimesExp e) => eval(e, (l, r) => value = l * r);

  @override
  void visitNumber(NumExp e) => value = e.value;

  void eval(OpExp e, Function(double, double) callback) {
    e.left.accept(this);
    final leftResult = getEvaluation();
    e.right.accept(this);
    final rightResult = getEvaluation();
    callback(leftResult, rightResult);
  }
}
```

Examples

Appinstaller uses this approach in order to manage Manifest.

Interpreter for arithmetic expressions in Flutter.

Code: https://github.com/euler2dot7/interpreter

Web App: https://euler2dot7.github.io/interpreter/#/