Feb 2 23:44 2020 Exp.ijava Page 1

```
Exp
%%%
    public abstract Val eval( Env env );
%%%
```

Feb 2 23:45 2020 LitExp.ijava Page 1

```
LitExp
%%%
    public String toString() {
        return lit.str;
    }
    public Val eval( Env env ) {
        return new Val( Integer.parseInt( lit.str ) );
    }
%%%
```

Feb 3 22:36 2019 VarExp.ijava Page 1

```
VarExp
%%%
    public String toString() {
        return var.str;
    }
    public Val eval( Env env ) {
        return env.applyEnv( var.str );
    }
%%%
```

Feb 2 23:34 2020 PrimAppExp.ijava Page 1

```
PrimAppExp:import
%%%
import java.util.Arrays;
%%%

PrimAppExp
%%%
    public Val eval( Env env ) {
        // evaluate the terms in the expression list
        // and apply the prim to the array of Vals
        List<Val> args = operands.evalOperands( env );
        Val [] va = args.toArray( new Val[ args.size() ] );
        return prim.apply( va );
    }

    public String toString() {
        return prim + "(" + operands + ")";
    }
%%%
```

Feb 2 23:33 2020 Operands.ijava Page 1

```
Operands:import
%%%
import java.util.stream.Collectors;
%%%

Operands
%%%
    /**
     * Fetch the values of each expression in the parameter (operands) list.
     */
    public List<Val> evalOperands( Env env ) {
        return expList.stream()
                            .map( exp -> exp.eval(env) )
                            .collect( Collectors.toList() );
    }

    public String toString() {
        return expList.stream()
                        .map( Exp::toString )
                        .collect( Collectors.joining( "," ) );
    }
%%%
```

Feb 3 23:22 2019 V1session1.txt Page 1

```
$ rep
--> 1
1
--> v
5
--> m
1000
--> +(2,3)
5
--> +(m,v)
1005
--> -(sub1(m),sub1(v))
995
--> add1(55)
56
-->
```

Feb 2 23:45 2020 Add1Prim.ijava Page 1

```
Add1Prim
%%%
    public Val apply(Val [] vals) {
        if ( vals.length != 1 )
            throw new RuntimeException( "One argument expected." );
        int i0 = vals[0].value;
        return new Val( i0 + 1 );
    }

    public String toString() {
        return "add1";
    }
%%%
```

Feb 2 23:45 2020 AddPrim.ijava Page 1

AddPrim
%%%
```
    public Val apply(Val [] vals) {
        if ( vals.length != 2 )
            throw new RuntimeException( "Two arguments expected." );
        int i0 = vals[0].value;
        int i1 = vals[1].value;
        return new Val( i0 + i1 );
    }

    public String toString() {
        return "+";
    }
```
%%%

Jan 30 23:15 2019 Prim.ijava Page 1

Prim
%%%
```
    /**
     * Apply this primitive (whatever it is) to the provided values.
     */
    public abstract Val apply( Val[] vals );
```
%%%

Feb 2 23:45 2020 Sub1Prim.ijava Page 1

Sub1Prim
%%%
```
    public Val apply(Val [] vals) {
        if ( vals.length != 1 )
            throw new RuntimeException( "One argument expected." );
        int i0 = vals[0].value;
        return new Val( i0 - 1 );
    }

    public String toString() {
        return "sub1";
    }
```
%%%

Feb 2 23:45 2020 SubPrim.ijava Page 1

SubPrim
%%%
```
    public Val apply(Val [] vals) {
        if ( vals.length != 2 )
            throw new RuntimeException( "Two arguments expected." );
        int i0 = vals[0].value;
        int i1 = vals[1].value;
        return new Val( i0 - i1 );
    }

    public String toString() {
        return "-";
    }
```
%%%