# Feature-Incremental Program Development

Katrina Van Laan; kdvanlaan@gmail.com
Xinyuan Zhang; xinyuan.chn@gmail.com

*We used MDELite7 for this project because we could not find the link for MDELite8*

## Evidence that the program was built one feature at a time:

In our zip folder we include five projects, each project adds a new feature on top of the previous one.

## Base:

- Gates are created in this format:

  *Gate A = xx.Gate("A", 0, 1, "Input");*
- No print() method for the circuit
- Added constructors for Circuit, Gate, and Wire classes.

## Beautify:

- Making Gate abstract class
- Add concrete classes for each sub-type of Gates, and put these classes into source lib
- Gates are created in this format:

  *Gate A = new Input(xx,"A");*
- Make Wire class polymorphism

## Table:

- Add Lists to store gates and wires in each circuit object:

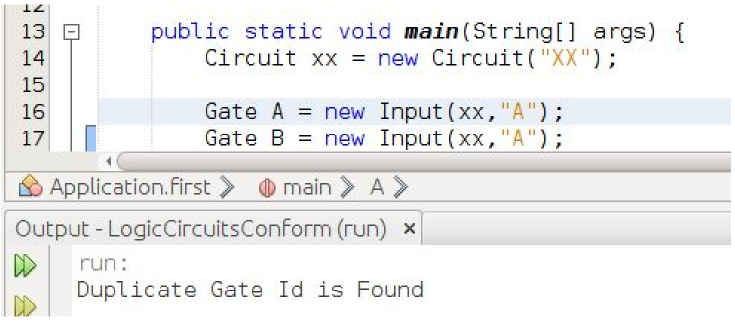
  *List<Gate> gates;*

  *List<Wire> wires;*
- Each time creating a new gate/wire to the circuit, add the Gate/Wire object to the corresponding circuit's lists.

## Constraints:

- Adds validate method to  Circuit class, checking for the following constraints:

  1: Each Gate has a unique id



```
    public static void main(String[] args) {
        Circuit xx = new Circuit("XX");

        Gate A = new Input(xx,"A");
        Gate B = new Input(xx,"A");
```

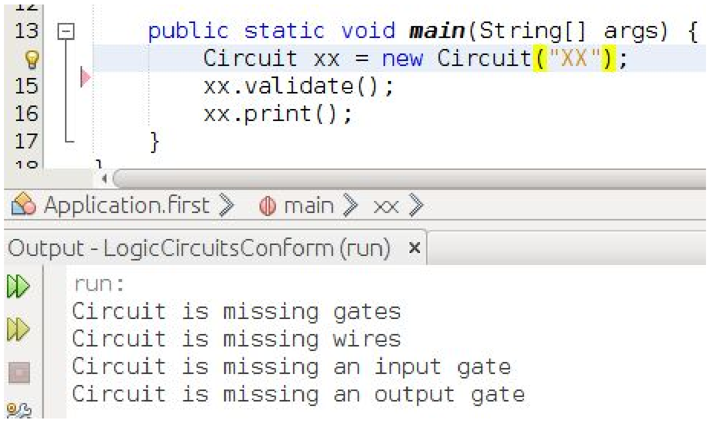
Application.first › main › A ›

Output - LogicCircuitsConform (run) ×

```
run:
Duplicate Gate Id is Found
```

  2. Circuit has >= 1 gates

  - This constraint triggers other constraints which we will show proof of independently



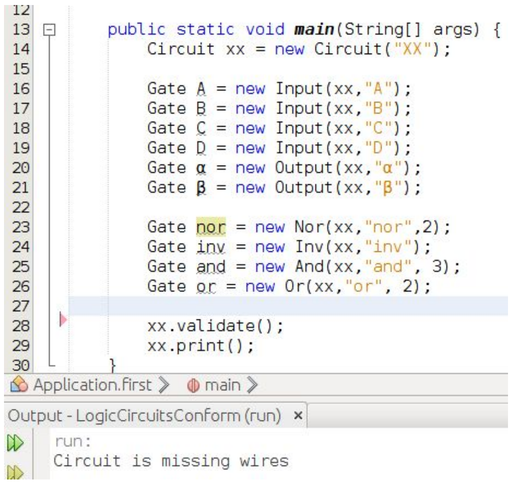
```
    public static void main(String[] args) {
        Circuit xx = new Circuit("XX");
        xx.validate();
        xx.print();
    }
```

Application.first › main › xx ›

Output - LogicCircuitsConform (run) ×

```
run:
Circuit is missing gates
Circuit is missing wires
Circuit is missing an input gate
Circuit is missing an output gate
```

  3. Circuit has >= 1 wires



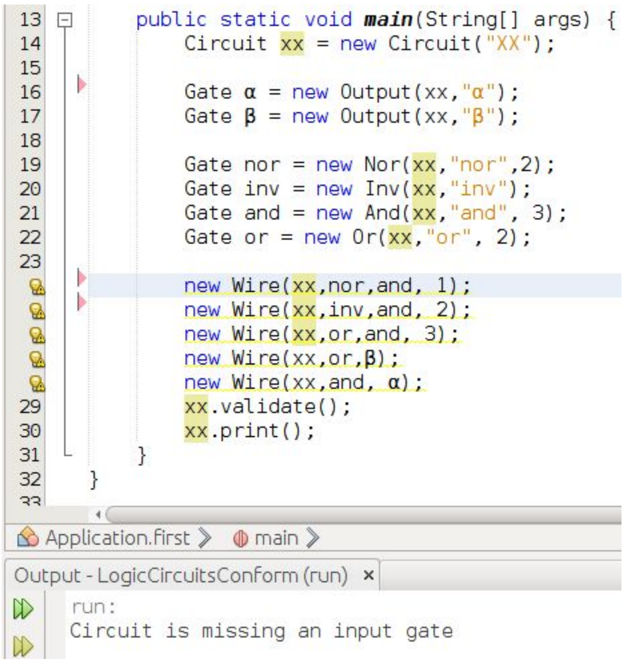
```
    public static void main(String[] args) {
        Circuit xx = new Circuit("XX");

        Gate A = new Input(xx,"A");
        Gate B = new Input(xx,"B");
        Gate C = new Input(xx,"C");
        Gate D = new Input(xx,"D");
        Gate α = new Output(xx,"α");
        Gate β = new Output(xx,"β");

        Gate nor = new Nor(xx,"nor",2);
        Gate inv = new Inv(xx,"inv");
        Gate and = new And(xx,"and", 3);
        Gate or = new Or(xx,"or", 2);

        xx.validate();
        xx.print();
    }
```

Application.first › main ›

Output - LogicCircuitsConform (run) ×

```
run:
Circuit is missing wires
```

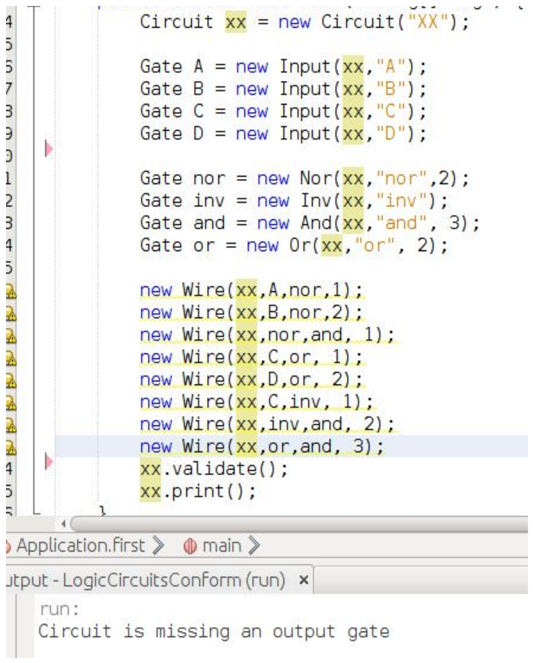
  4. Circuit has >= 1 input gates



```
    public static void main(String[] args) {
        Circuit xx = new Circuit("XX");

        Gate α = new Output(xx,"α");
        Gate β = new Output(xx,"β");

        Gate nor = new Nor(xx,"nor",2);
        Gate inv = new Inv(xx,"inv");
        Gate and = new And(xx,"and", 3);
        Gate or = new Or(xx,"or", 2);

        new Wire(xx,nor,and, 1);
        new Wire(xx,inv,and, 2);
        new Wire(xx,or,and, 3);
        new Wire(xx,or,β);
        new Wire(xx,and, α);
        xx.validate();
        xx.print();
    }
}
```

```
run:
Circuit is missing an input gate
```

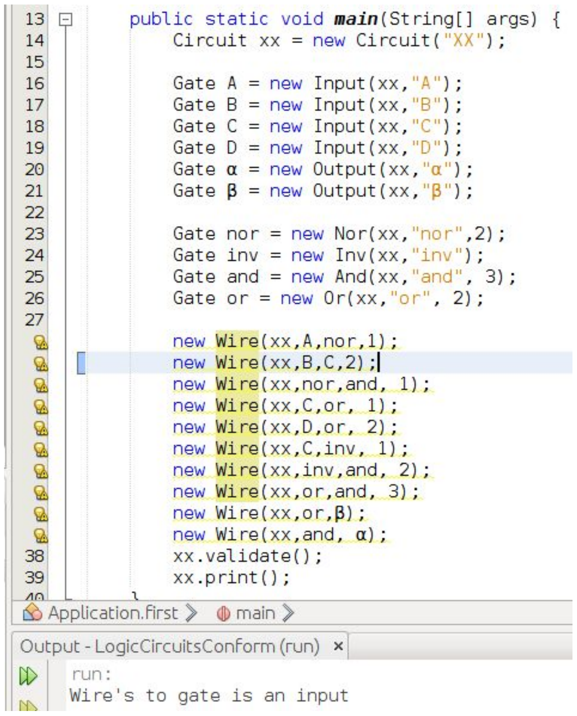
5. Circuit has >= 1 output gates



```
        Circuit xx = new Circuit("XX");

        Gate A = new Input(xx,"A");
        Gate B = new Input(xx,"B");
        Gate C = new Input(xx,"C");
        Gate D = new Input(xx,"D");

        Gate nor = new Nor(xx,"nor",2);
        Gate inv = new Inv(xx,"inv");
        Gate and = new And(xx,"and", 3);
        Gate or = new Or(xx,"or", 2);

        new Wire(xx,A,nor,1);
        new Wire(xx,B,nor,2);
        new Wire(xx,nor,and, 1);
        new Wire(xx,C,or, 1);
        new Wire(xx,D,or, 2);
        new Wire(xx,C,inv, 1);
        new Wire(xx,inv,and, 2);
        new Wire(xx,or,and, 3);
        xx.validate();
        xx.print();
```

```
run:
Circuit is missing an output gate
```

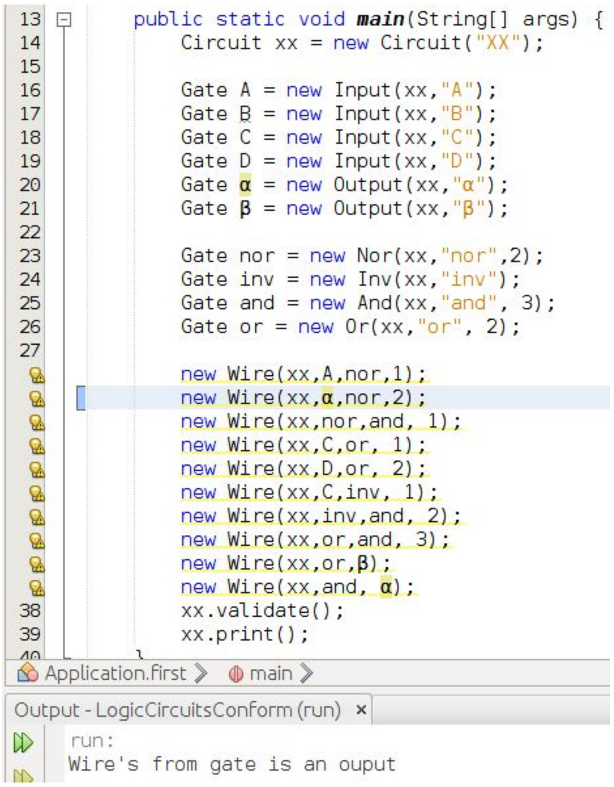
6. A wire must have a to and a from
- We have no proof for this constraint because the wire constructor requires a to and a from, thus the code will not compile without the constraint
7. A wire cannot have a to that is type input



```
    public static void main(String[] args) {
        Circuit xx = new Circuit("XX");

        Gate A = new Input(xx,"A");
        Gate B = new Input(xx,"B");
        Gate C = new Input(xx,"C");
        Gate D = new Input(xx,"D");
        Gate α = new Output(xx,"α");
        Gate β = new Output(xx,"β");

        Gate nor = new Nor(xx,"nor",2);
        Gate inv = new Inv(xx,"inv");
        Gate and = new And(xx,"and", 3);
        Gate or = new Or(xx,"or", 2);

        new Wire(xx,A,nor,1);
        new Wire(xx,B,C,2);
        new Wire(xx,nor,and, 1);
        new Wire(xx,C,or, 1);
        new Wire(xx,D,or, 2);
        new Wire(xx,C,inv, 1);
        new Wire(xx,inv,and, 2);
        new Wire(xx,or,and, 3);
        new Wire(xx,or,β);
        new Wire(xx,and, α);
        xx.validate();
        xx.print();
```

```
run:
Wire's to gate is an input
```

8. A wire cannot have a from that is type output



```
    public static void main(String[] args) {
        Circuit xx = new Circuit("XX");

        Gate A = new Input(xx,"A");
        Gate B = new Input(xx,"B");
        Gate C = new Input(xx,"C");
        Gate D = new Input(xx,"D");
        Gate α = new Output(xx,"α");
        Gate β = new Output(xx,"β");

        Gate nor = new Nor(xx,"nor",2);
        Gate inv = new Inv(xx,"inv");
        Gate and = new And(xx,"and", 3);
        Gate or = new Or(xx,"or", 2);

        new Wire(xx,A,nor,1);
        new Wire(xx,α,nor,2);
        new Wire(xx,nor,and, 1);
        new Wire(xx,C,or, 1);
        new Wire(xx,D,or, 2);
        new Wire(xx,C,inv, 1);
        new Wire(xx,inv,and, 2);
        new Wire(xx,or,and, 3);
        new Wire(xx,or,β);
        new Wire(xx,and, α);
        xx.validate();
        xx.print();
```

```
run:
Wire's from gate is an ouput
```

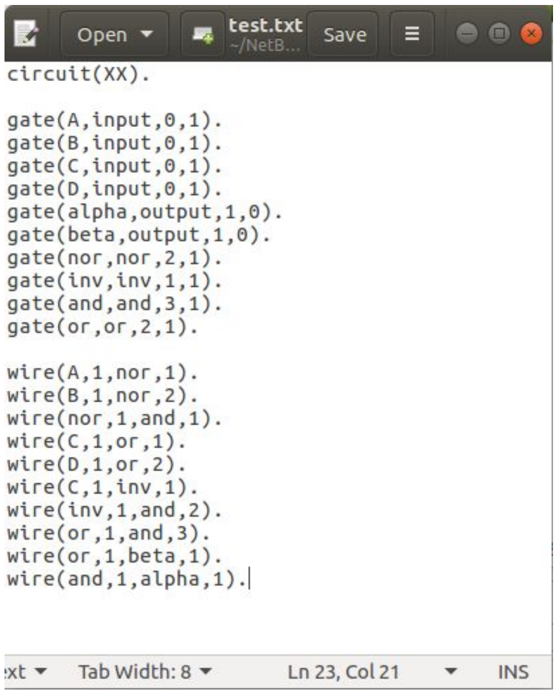
## Evaluation:

- Attach evaluation values to each gate
- Add initValues() method to circuit, so that each gate is set to be unknown initially.
- Add abstract method get() into Gate class, each concrete gate extends the get().
- Concrete gates looping through all its input pins, and track the corresponding input gate along the wire connecting to his input pins. This will recursively track all the gates to get the evaluation value until it reaches the input gates.
- The evaluation value of each concrete gate is based on the rule of gate operation.
- Output the output gate evaluation values.

## Regression Tests:

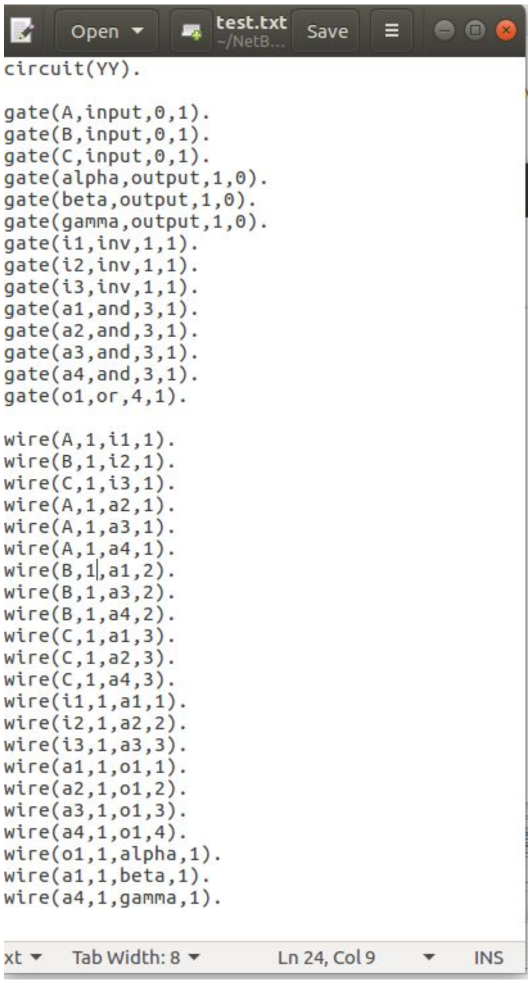
The program we are using to implement the regression tests is under the junit test package: test/logiccircuits/CircuitTest.java:

For circuit XX:



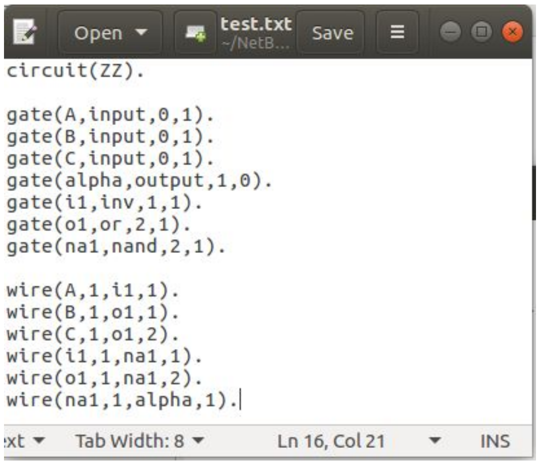
```
circuit(XX).

gate(A,input,0,1).
gate(B,input,0,1).
gate(C,input,0,1).
gate(D,input,0,1).
gate(alpha,output,1,0).
gate(beta,output,1,0).
gate(nor,nor,2,1).
gate(inv,inv,1,1).
gate(and,and,3,1).
gate(or,or,2,1).

wire(A,1,nor,1).
wire(B,1,nor,2).
wire(nor,1,and,1).
wire(C,1,or,1).
wire(D,1,or,2).
wire(C,1,inv,1).
wire(inv,1,and,2).
wire(or,1,and,3).
wire(or,1,beta,1).
wire(and,1,alpha,1).
```

For fircuit YY:



```
circuit(YY).

gate(A,input,0,1).
gate(B,input,0,1).
gate(C,input,0,1).
gate(alpha,output,1,0).
gate(beta,output,1,0).
gate(gamma,output,1,0).
gate(i1,inv,1,1).
gate(i2,inv,1,1).
gate(i3,inv,1,1).
gate(a1,and,3,1).
gate(a2,and,3,1).
gate(a3,and,3,1).
gate(a4,and,3,1).
gate(o1,or,4,1).

wire(A,1,i1,1).
wire(B,1,i2,1).
wire(C,1,i3,1).
wire(A,1,a2,1).
wire(A,1,a3,1).
wire(A,1,a4,1).
wire(B,1,a1,2).
wire(B,1,a3,2).
wire(B,1,a4,2).
wire(C,1,a1,3).
wire(C,1,a2,3).
wire(C,1,a4,3).
wire(i1,1,a1,1).
wire(i2,1,a2,2).
wire(i3,1,a3,3).
wire(a1,1,o1,1).
wire(a2,1,o1,2).
wire(a3,1,o1,3).
wire(a4,1,o1,4).
wire(o1,1,alpha,1).
wire(a1,1,beta,1).
wire(a4,1,gamma,1).
```

For circuit ZZ:



```
circuit(ZZ).

gate(A,input,0,1).
gate(B,input,0,1).
gate(C,input,0,1).
gate(alpha,output,1,0).
gate(i1,inv,1,1).
gate(o1,or,2,1).
gate(na1,nand,2,1).

wire(A,1,i1,1).
wire(B,1,o1,1).
wire(C,1,o1,2).
wire(i1,1,na1,1).
wire(o1,1,na1,2).
wire(na1,1,alpha,1).
```