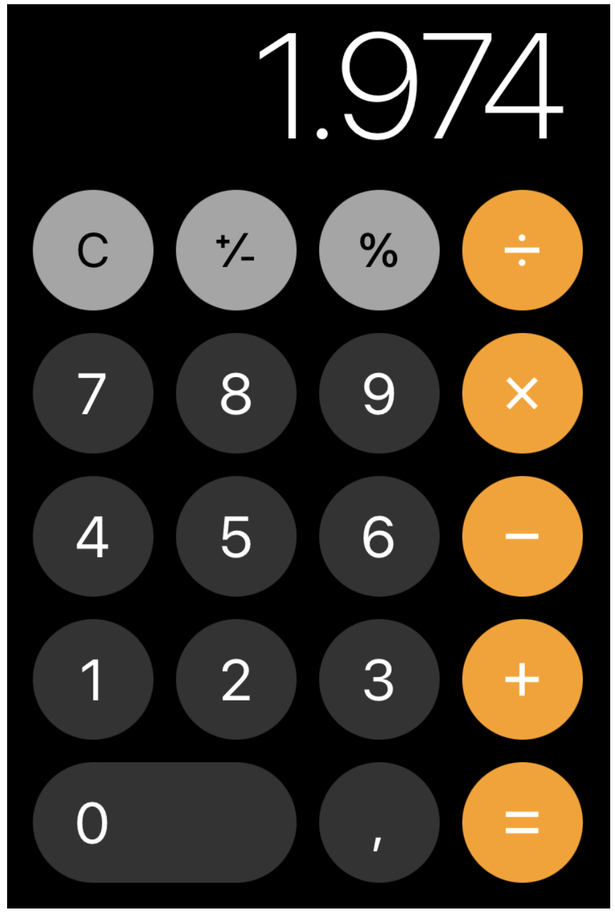
Java – Solution of assignments

Paolo Vercesi

Technical Program Manager

Assignment



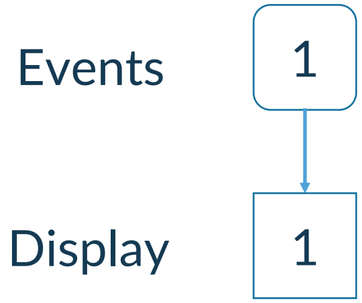
Define a calculator class that

1. receives “events” from a calculator keyboard
2. sends the output to a Display object

```
class Display {
    void display(String text) {
        System.out.println(text);
    }
}
```

```
class Calculator {

    final Display display;
    //...

    Calculator(Display display) {
        this.display = display;
    }

    void plusPressed() {
        //...
    }

    void zeroPressed() {
        //...
    }

    //...
}
```

Start simple

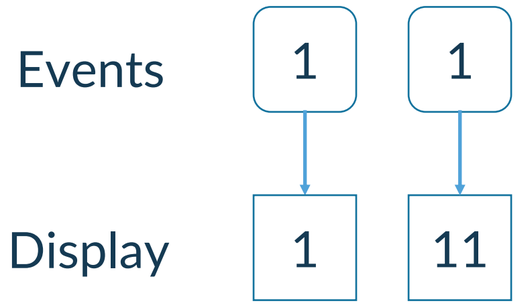



```
$ java Calculator.java
1
```

Calculator.java

```
class CalculatorMain {
    public static void main(String[] args) {
        var calculator = new Calculator(new Display());
        calculator.onePressed();
    }
}

public class Calculator {

    final Display display;

    public Calculator(Display display) {
        this.display = display;
    }

    void onePressed() {
        display.display("1");
    }

}
```

Add one more event

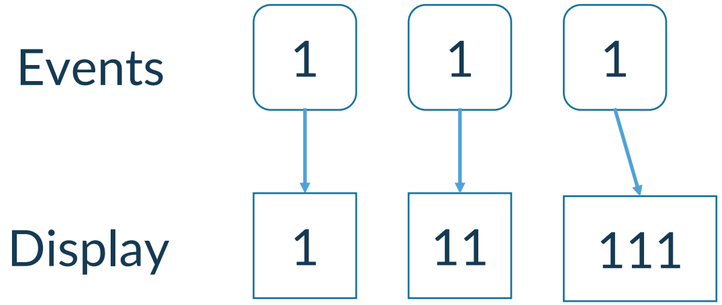



```
$ java Calculator.java
1
11
```

Calculator.java

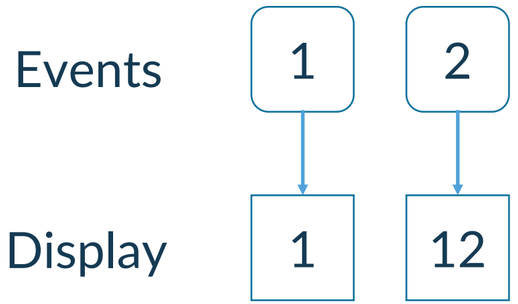
```
class CalculatorMain {
    public static void main(String[] args) {
        var calculator = new Calculator(new Display());
        calculator.onePressed();
        calculator.onePressed();
    }
}

public class Calculator {

    final Display display;
    String string;

    public Calculator(Display display) {
        this.display = display;
        string = "";
    }

    void onePressed() {
        string += "1";
        display.display(string);
    }
}
```

It seems to work

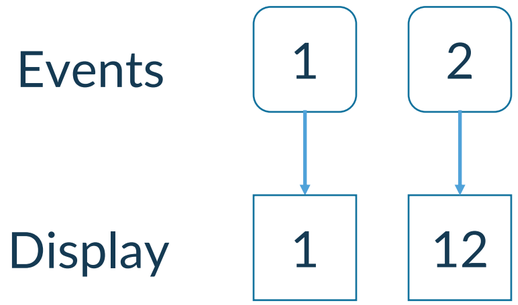



```
$ java Calculator.java
1
11
111
```

Let’s extends the capabilities of our Calculator with an event of different type



Add one different event

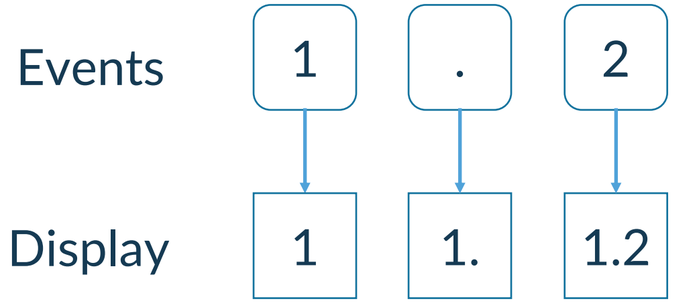



```
$ java Calculator.java
1
12
```

Calculator.java

```
class CalculatorMain {
    public static void main(String[] args) {
        var calculator = new Calculator(new Display());
        calculator.onePressed();
        calculator.twoPressed();
    }
}

public class Calculator {
    final Display display;
    int string;

    public Calculator(Display display) {
        this.display = display;
        string = "";
    }

    void onePressed() {
        string += "1";
        display.display(string);
    }

    void twoPressed() {
        string += "2";
        display.display(string);
    }
}
```

Dealing with dots

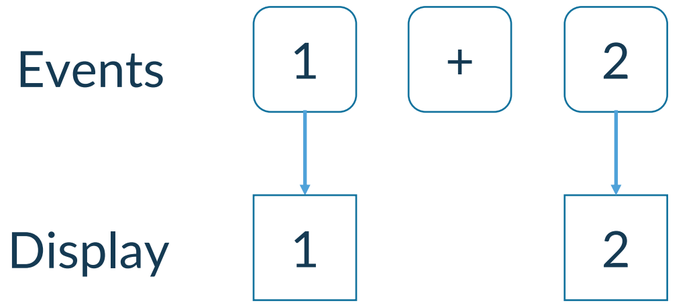



```
$ java Calculator.java
1
1.
1.2
```

Calculator.java

```java
class CalculatorMain {
    public static void main(String[] args) {
        var calculator = new Calculator(new Display());
        calculator.onePressed();
        calculator.dotPressed();
        calculator.twoPressed();
    }
}

public class Calculator {
    final Display display;
    int string;

    public Calculator(Display display) {
        this.display = display;
        string = "";
    }

    void onePressed() {
        string += "1";
        display.display(string);
    }

    void twoPressed() {
        string += "2";
        display.display(string);
    }

    void dotPressed() {
        string += ".";
        display.display(string);
    }
}
```

Introduce operators



Calculator.java

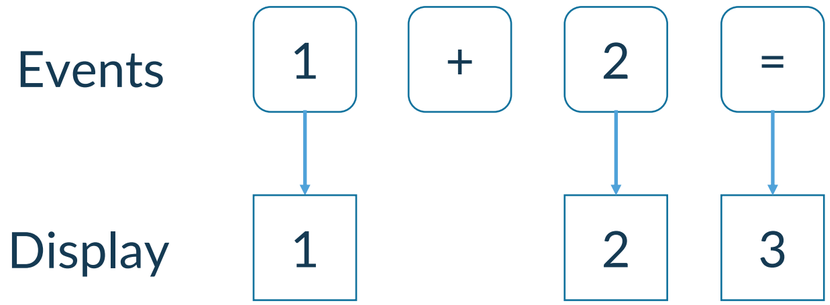
```
class CalculatorMain {
    public static void main(String[] args) {
        var calculator = new Calculator(new Display());
        calculator.onePressed();
        calculator.plusPressed();
        calculator.twoPressed();
    }
}
```

```
$ java Calculator.java
1
2
```

Calculator.java

```
public class Calculator {
    final Display display;
    int string;

    public Calculator(Display display) {
        this.display = display;
        string = "";
    }

    void onePressed() {
        string += "1";
        display.display(string);
    }

    void twoPressed() {
        string += "2";
        display.display(string);
    }

    void dotPressed() {
        string += ".";
        display.display(string);
    }

    void plusPressed() {
        string = "";
    }

}
```

Introduce equal



Calculator.java

```
class CalculatorMain {
    public static void main(String[] args) {
        var calculator = new Calculator(new Display());
        calculator.onePressed();
        calculator.plusPressed();
        calculator.twoPressed();
        calculator.equalPressed();
    }
}
```

```
$ java Calculator.java
1
2
3.0
```

Calculator.java

```
public class Calculator {
    final Display display;
    String string, operator;
    double op1;

    public Calculator(Display display) {
        this.display = display; string = "";
    }

    void onePressed() {
        string += "1"; display.display(string);
    }

    void twoPressed() {
        string += "2"; display.display(string);
    }

    void dotPressed() {
        string += "."; display.display(string);
    }

    void plusPressed() {
        op1 = Double.parseDouble(string);
        operator = "+"; string = "";
    }

    void equalPressed() {
        if ("+".equals(operator)) {
            double op2 = Double.parseDouble(string);
            double result = op1 + op2;
            display.display("" + result); operator = "";
        }
    }
}
```

What is missing?

A lot of scenarios

When we turn on the Calculator it must show “0”

We can add multiple dots

Operations are not concatenated

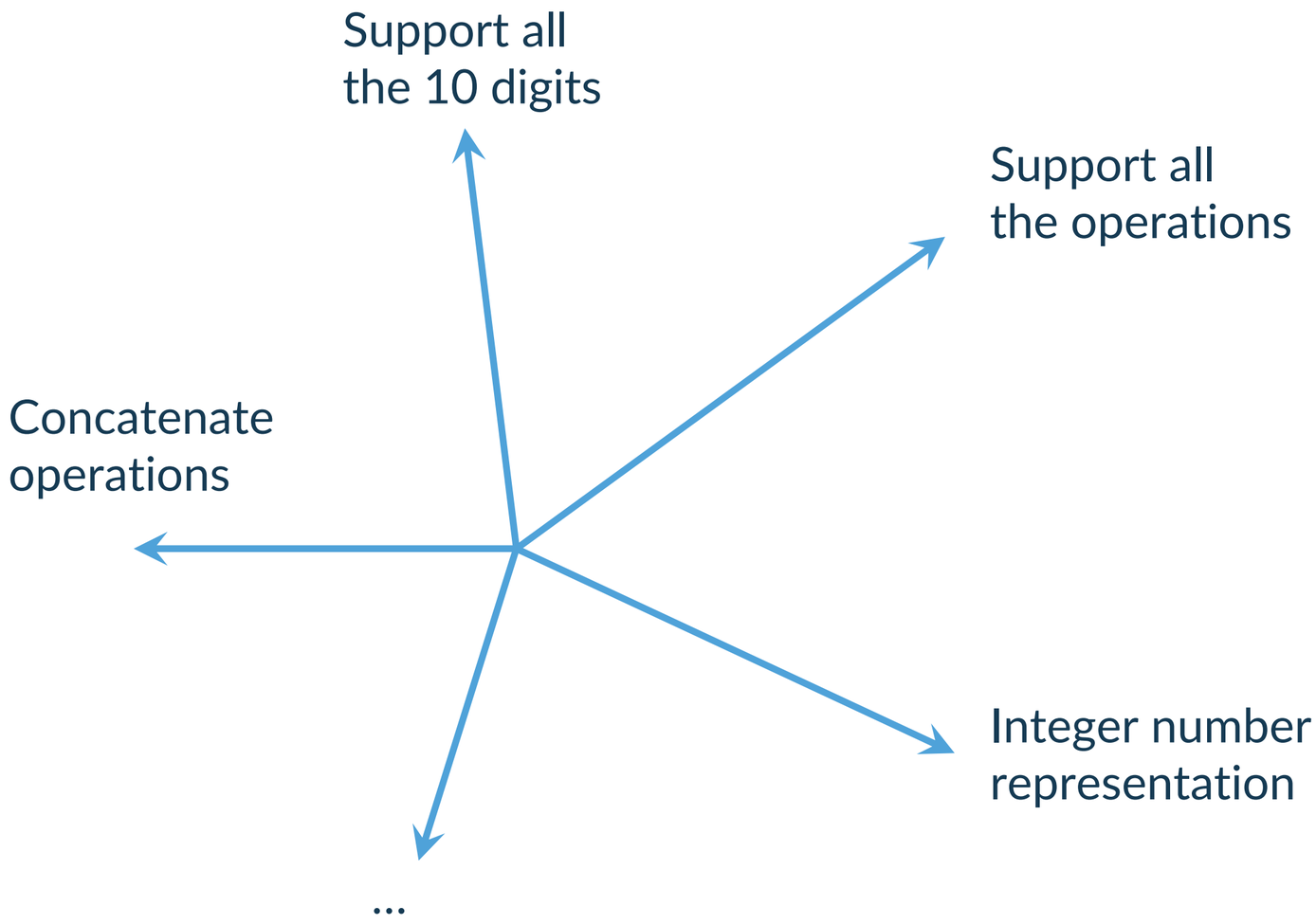
Typing numbers after result evaluation

and many more

What we have learnt

- ✓ Even an apparently trivial problem can become quickly very complex
- ✓ Different usage scenarios may trigger different behaviors
- ✓ The design of the solution is not unique and it not obvious from the beginning
- ✓ A problem should be decomposed over different directions
- ✓ An incremental approach can help developing the solution

Dealing with complexity



Incremental development

1. Select a usage scenario
2. Implement it
3. Don’t forget to test all the previous scenarios
4. Start again from point 1

More on this topic in the agile software development part of this course

Thank you!

esteco.com