Methods that return values

COMP 102

Victoria University of Wellington

ReactionTimeMeasurer Problem

- A good experiment would measure the average time over a series of trials
  - Our program measures and reports for each trial.

- Need to add up all the times, and compute average:
  - problem:
    - MeasureReactionTime needs to add up the times
    - MeasureQuestion actually measures the time, but prints it out.
    - How do we get the time back from MeasureQuestion to MeasureTime?

  - We need to make MeasureQuestion return the time value to MeasureTime.

Methods that return values

- Some methods just have "effects":

```
UI.println("Hello there!");
UI.printf("%4.2f miles is the same as %4.2f km\n", mile, km);
UI.fillRect(100, 100, wd, ht);
UI.sleep(1000);
```

- Some methods just return a value:

```
long  now = System.currentTimeMillis();
double distance = 20 * Math.random();
double ans = Math.pow(3.5, 17.3);
```

- Some methods do both:

```
double height = UI.askDouble("How tall are you");
```

Defining methods to return values

Improving ReactionTimeMeasurer:

make measureQuestion **return** a value instead of just printing it out.

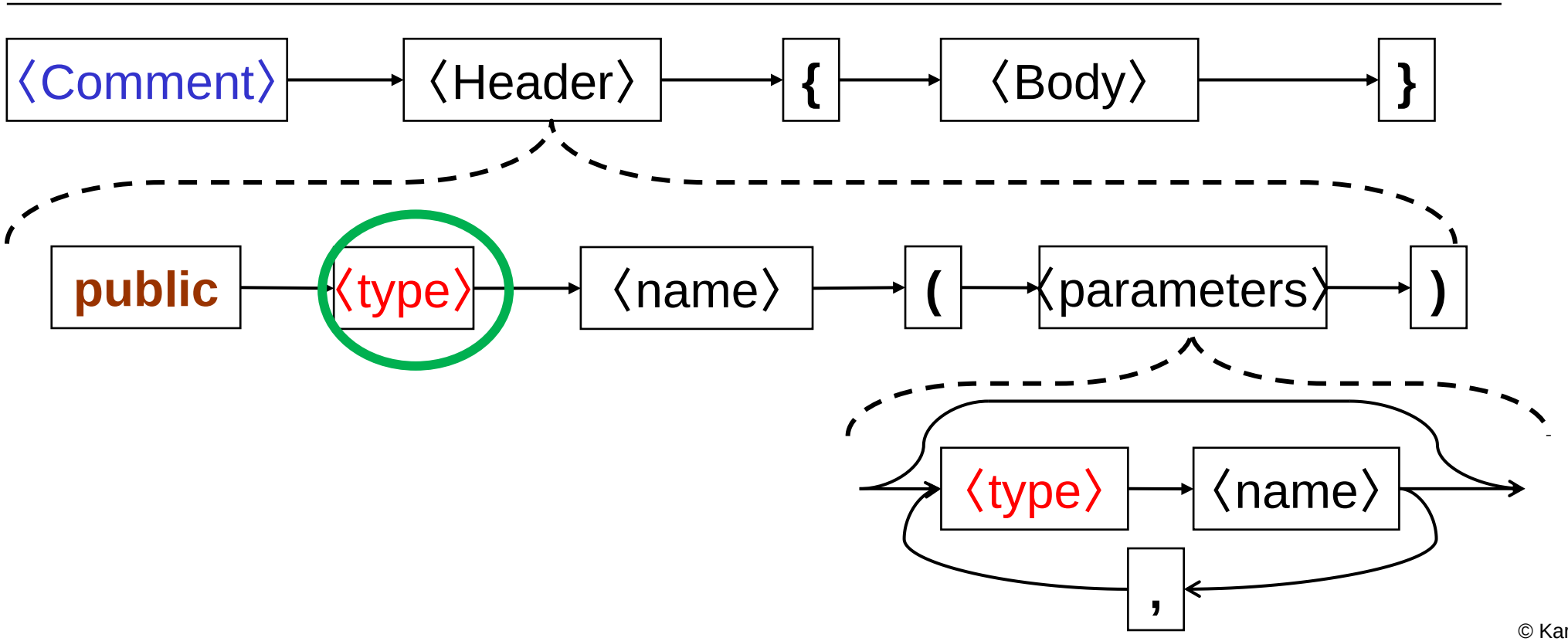
```
public void measureReactionTime() {
    long time = 0;
    time = time + this.measureQuestion("John Quay was the Prime Minister");
    time = time + this.measureQuestion("11 x 13 = 143");
    time = time + this.measureQuestion("Summer is warmer than Winter");
    time = time + this.measureQuestion(" Wellington's pop > 1,000,000 ");
    UI.printf("Average reaction time = %d milliseconds\n", (time / 4));
}
```

Specifies the type of value returned.
void means "no value returned"

```
public long measureQuestion(String fact) {
    long startTime = System.currentTimeMillis();
    ……
}
```

Syntax: Method Definitions (v3: return type)

```
/** Measure time taken to answer a question*/
        public long measureQuestion ( String fact ){
                long startTime = System.currentTimeMillis();
                    :
```



Defining methods to return values

If you declare that a method returns a value,
then the method body must return one!

```
public  long  measureQuestion(String fact) {
    long startTime = System.currentTimeMillis();
    String  ans = UI.askString("Is it true that " + fact);
    long endTime = System.currentTimeMillis();
   return  (endTime - startTime) ;
}
```

New kind of statement
Means: exit the method and return the value
The value must be of the right type

- Maybe show in debugger

More about Return

- If a method has a return type, it must have a **return** statement that returns a value

- It must return a value for every possible path

  ⇒ may need several **return** statements:

```
public String fullDayName(String str){
    str = str.toLowerCase();
    if (str.startsWith("m")){
        return "Monday";
    }
    else if (str.startsWith("tu")){
        return "Tuesday";
    }
    else if (str.startsWith("w")){
        return "Wednesday";
    }....
```

More about Return

- **return** does two things:
  - specifies the value that will be returned to the calling method
  - exits the current method, skipping over all remaining statements.

- Methods with a void return type:
  - Can't return a value
  - Can still have a **return** statement (**return**;) with no value.

    ⇒ exit method at this point.

```
public void drawLollipop(double x, double y, double size, double length,){
    if (size < 2 || length < size/2){     // invalid parameters
            return;
    }
    // draw the lollipop
    UI.setColor(Color.red);
    UI.fillRect(x-size/2; y-size/2, size, size);
      :
```