

What is a Program

A program is a specification for the behaviour of a computer:

- What the computer should do when:
 - the program is started
 - the user types something
 - the user clicks the mouse
 - a message arrives over the network
 - some input from a camera/switch/sensor arrives.
 -
- Responses may be simple or very complex.
- A program consists of
 - descriptions of responses to events/requests
 - written as instructions
 - in a language the computer can understand:
 - Low level, High level, Specialised

Machine Language

- What the computer can understand
- Different for each computer
- Very detailed, low-level control of the computer
- Horrible to read

⋮
000XX00X 0X00XXXX
0XX0X00X 00XXX0X0
00X0X00X X0XX0X0X
⋮

copy the contents of memory location 143
into register 1.

add the contents of memory location 116
to the contents of register 1.

copy the contents of register 1
to memory location 181.

Pattern of bits controls
the switches that
operate the CPU

High Level Programming Languages

- Designed for people to use
- Designed to be translated into machine language
 - compiled (translated all at once), or
 - interpreted (translated one step at a time), or
 - compiled to an intermediate language, then interpreted

Must be

- **Precise:** no ambiguity about what to do
- **Expressive:** must be able to specify whatever you want done.
- **Readable:** People must be able to read the instructions.
- **Translatable:** able to be translated into machine language
- **Concise:** not “long-winded” or redundant

FORTRAN
 LISP
 Algol
 COBOL
 Basic
 C
 Pascal
 Simula
 Modula
 PHP
 Javascript

Smalltalk
 ML
 Ada
 C++
 Eiffel
 Prolog
 Haskell
 Miranda
 Java
 C#
 Python
 Scratch
 GameMaker
 Alice

Programming Languages

- Different languages support different “paradigms”:
(ways of designing programs)
 - imperative,
 - object-oriented,
 - functional,
 - logic programming, ...

Object Oriented programming languages:

- Organise program around Classes (types) of objects
- Each class of objects can perform a particular set of actions
- Most instructions consist of asking an object to perform one of its actions

NCEA vs University

- NCEA has lots of components with individual grades; not all needed.
 - being strategic on which components to do, and which to ignore
- Uni has lots of components that are combined into a single grade; all count.
 - being strategic on how much time to put into each component.
- NCEA (internal) may allow resubmission
- Uni generally does NOT allow resubmission
- NCEA focusses on getting Achieved; Excellence is very difficult.
 - if you have Achieved, may not be worth trying harder.
- Uni focusses on grades; A's are more achievable
 - Just passing is not enough
 - It's worth doing more because it will increase your grade.

Java

- A high-level Object-Oriented programming language
- Designed by Sun Microsystems, early-mid 1990's.
- Widely used in teaching and industry
- Related to C++, but simpler. Similar to C#.
- Good for interactive applications.
- Extensive libraries of predefined classes to support, UIs, graphics, databases, web applications, ...
- Portable between kinds of computers.

A Java Program

```

import ecs100.*;
/** Program to compute the average of a sequence of numbers */
public class MeanFinder {
    public MeanFinder () {
        Ul.addButton("Compute Mean" , this::doFindMean);
    }
    /** Ask for sequence of numbers and print the mean */
    public void doFindMean () {
        ArrayList<Double> numbers = Ul.askNumbers( "Enter numbers" );
        if (numbers.size() > 0) { Ul.println( "Mean = " + this.computeMean(numbers) ); }
        else { Ul.println( "You entered no numbers" ); }
    }
    /** Compute the mean (average) of a sequence of numbers */
    public double computeMean (ArrayList<Double> nums ) {
        double total= 0;
        for (int num : nums) {
            total = total + num;
        }
        return (total / nums.size() );
    }
}

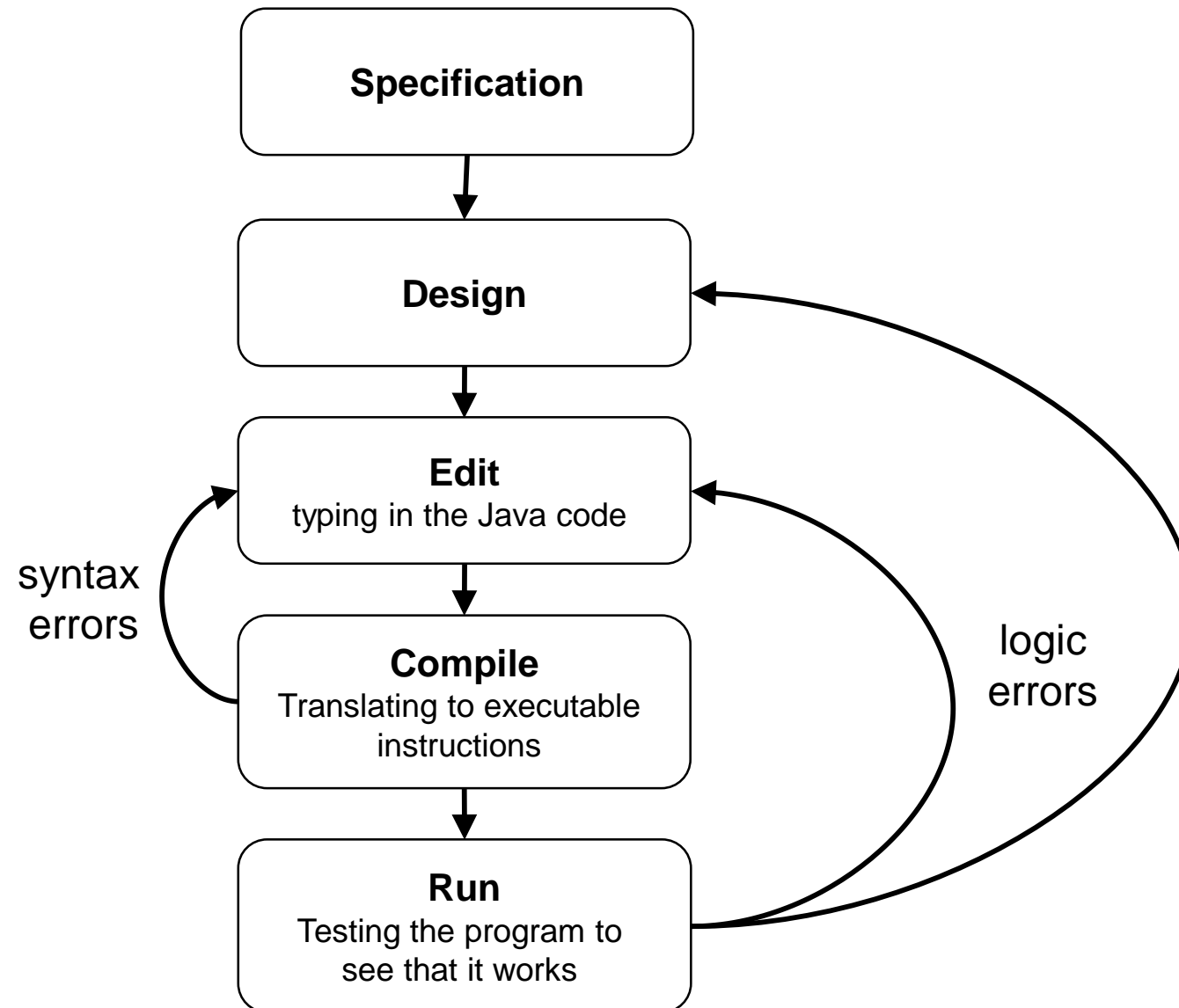
```

Learning to Program in Java

What's involved?

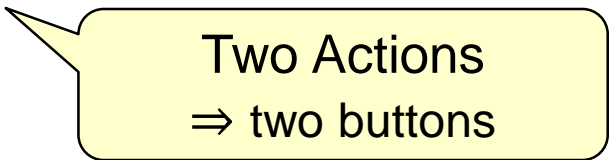
- Understand what the language can specify
- Problem solving:
 - program design,
 - data structuring,
- Programming language (Java):
 - syntax and semantics
 - style and common patterns
 - libraries of code written by other people
- Testing and Debugging (fixing).
- Common patterns in program design.
 - Important data structures and algorithms.

Constructing Programs in Java



A first Java Program

- Task: Write a temperature conversion program: $C \Leftrightarrow F$
- Step 1: Specification: what is it supposed to do?
 - Write a program that will let the user do two things:
 - print out the conversion formula
 - let user enter temperature in Fahrenheit, and print out in Celsius.
- Step 2: Design:
 - For calculate action:
 - Ask user for the Fahrenheit value to be converted
 - Print Celsius value:
 - Calculate Celsius value out of given value $(F-32.0)*5.0/9.0$
 - Print out the answer



Two Actions
 \Rightarrow two buttons

Designing the Java program

Step 3: Editing

- Need to write this design in the Java language.
 - ➔ Need an *object*: a "temperature calculator"
 - all actions must be performed on some object
 - ➔ Need a *class* to describe the object
 - ➔ The class needs a name
 - ➔ The class needs to specify a *constructor* to set up the interface
 - ➔ The class needs to specify the two actions its objects can do
 - ➔ Define *methods* to do things.
 - ➔ Give names to the methods
 - ➔ specify what the methods will do

Writing the Java code

```

import ecs100.*;
/** Program for converting between temperature scales */
public class TemperatureCalculator{
    /** Constructor: Set up interface */
    public TemperatureCalculator (){
        UI.addButton("Formula", this:: printFormula);
        UI.addButton("F->C", this:: doFahrenheitToCelsius);
    }
    /** Print conversion formula */
    public void printFormula ( ) {
        UI.println("Celsius = (Fahrenheit - 32) *5/9");
    }
    /** Ask for Fahrenheit and convert to Celsius */
    public void doFahrenheitToCelsius(){
        double fahrenheit = UI.askDouble("Fahrenheit:");
        this. convertToCelsius(fahrenheit);
    }
    /** Print Fahrenheit temperature as Celsius */
    public void convertToCelsius(double temp){
        double celsius = (temp - 32.0) * 5.0 / 9.0;
        UI.println(temp + " F -> " + celsius + " C");
    }
}

```

Comments

Keywords

Identifiers

Strings

Types

Numbers

Operators

Punctuation

I have chosen to split
into two separate
methods.

Could have just one
bigger method.

Elements of the program

Program Structure:

- Import
 - list the "libraries" you will use (We always use ecs100, and usually java.awt.Color and java.util.*)
- Class
 - Top level component of program
 - Describes a class of objects
 - Specifies the set of actions this kind of object can perform
 - (Can also specify information the objects can hold)
 - Note name, and conventions for naming.
- Constructor
 - Called when object is created
 - Typically sets up the user interface (in one-class programs)
- Methods
 - Main elements of a class
 - Each method describes an action that objects of this class can perform

Elements of the program

- Comments *vs* Code
- **Keywords** / Identifiers / **Strings** / **Types** / numbers / operators and punctuation
 - **Keywords** : words with special meaning in the Java Language
eg: **public**, **class**, **if**, **while**, ...
mostly to do with the structure of the program
 - **Identifiers** : other words, used to refer to things in the program.
mostly made up by the programmer,
some are predefined.
 - **Strings** : bits of text that the program will manipulate.
always surrounded by " and "
 - **Types** : names for kinds of values.
 - **numbers**
 - **operators and punctuation** : + - * / = % . ; , () { } [] ' "
all have precise meanings and rules for use

Actions in a program

- Method calls *object . method (arguments)*
 - telling an object to do one of its methods, passing the necessary information as arguments:


```
UI.println("Celsius = (Fahrenheit - 32) *5/9");
this.printCelsius(fahrenheit);
UI.drawRect(100, 200, 50, 75);
UI.addButton("Draw", this::doDraw);
```
 - What are the possible objects? what are the possible methods.
 - UI object has methods for
 - Printing, asking, drawing, buttons,
 - this object – the one we are defining – has the methods being defined in the class
- Assignment statements *place = value*
 - putting a value in a place


```
double celsius = (fahren - 32.0) * 5.0 / 9.0;
double fahren= UI.askDouble("Fahrenheit:");
```

BlueJ

- BlueJ is an IDE for Java
(Integrated Development Environment)
 - Class manager, for keeping track of the files in your program
 - Editor for entering and modifying the program
 - Built-in compiler interface to help compile and fix the syntax errors
 - Special interface to make it easy to construct objects and call methods on them.
- Let's do it... editing in BlueJ

Compiling and Running

Step 4: Compiling

- If there are syntax errors (invalid Java)
then the compiler will complain and list all the errors
 - ⇒ read the error message to work out what's wrong
 - ⇒ fixing syntax errors until it compiles without complaint
- BlueJ makes this process easier

Let's do it...

Compiling and Running

Step 4: Compiling

- If there are syntax errors (invalid Java)
then the compiler will complain and list all the errors
 - ⇒ read the error message to work out what's wrong
 - ⇒ fixing syntax errors until it compiles without complaint
- BlueJ makes this process easier

Step 5: Running and Testing

- Must run the program and test it on lots of different input.
 - BlueJ makes it easy to run individual methods.

Using BlueJ for Java Programs

Simple use of BlueJ for simple programs:

1. Edit the class file(s) to define the methods
2. Compile the class
3. Create an object of the class
 - right click on the rectangle representing the class
 - select “new.....”
⇒ a red square representing the object
4. Call methods on the object
 - right click on the square representing the object
 - select the method.

Writing your own programs

How?

- Use other programs as models, and then modify
 - Very useful strategy
 - Lectures have examples that you can use as models for your assignment programs

A new program

- Calculator to convert inches to centimeters

```
import ecs100.*;  
/** Program to convert inches to centimeters */
```

```
public class TemperatureCalculator{  
    public void doFahrenheitToCelsius(){  
        double fahrenheit = UI.askDouble("Fahrenheit:");  
        this.convertToCelsius(fahrenheit);  
    }  
    public void convertToCelsius(double temp){  
        double celsius = (temp - 32.0) * 5.0 / 9.0;  
        UI.println(temp + " F -> " + celsius + " C");  
    }  
}
```

Writing your own programs

How?

- Use other programs as models, and then modify
 - Very useful strategy

BUT

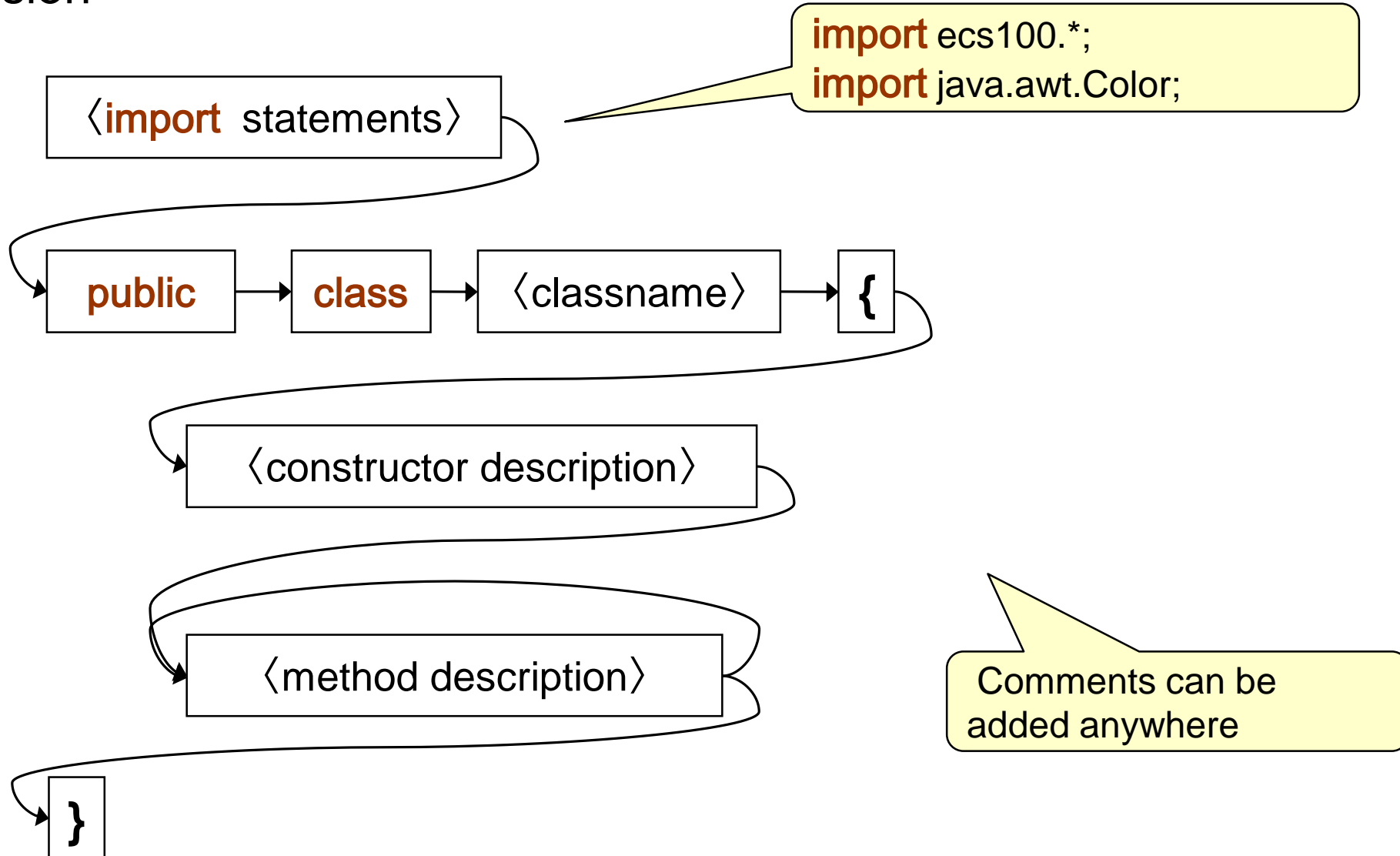
- It can be hard to work out how to modify
- It is very limiting

Need to understand the language

- ⇒ vocabulary
- ⇒ syntax rules
- ⇒ meaning (“semantics”)

Syntax rules: Program structure

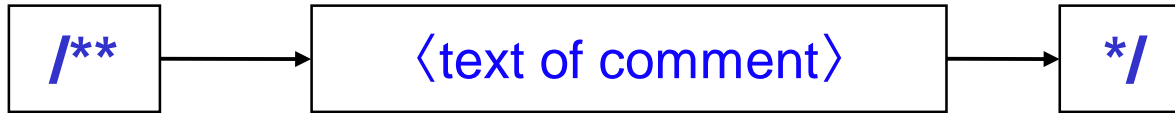
- First version



Comments

Three kinds of comments:

- Documentation comments



Top of class,
Before each method

eg `/** Program for converting between temperature scales */`

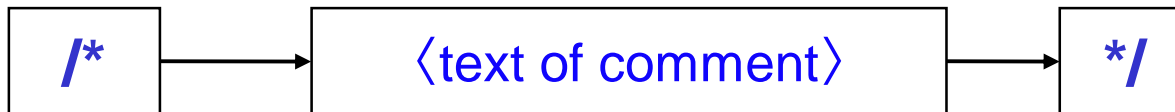
- end-of-line comments



at end of any line

eg `double celsius = (fahrenheit - 32.0) * 5.0 / 9.0; // compute answer`

- anywhere comments



multi-line, or
middle of line, or ...

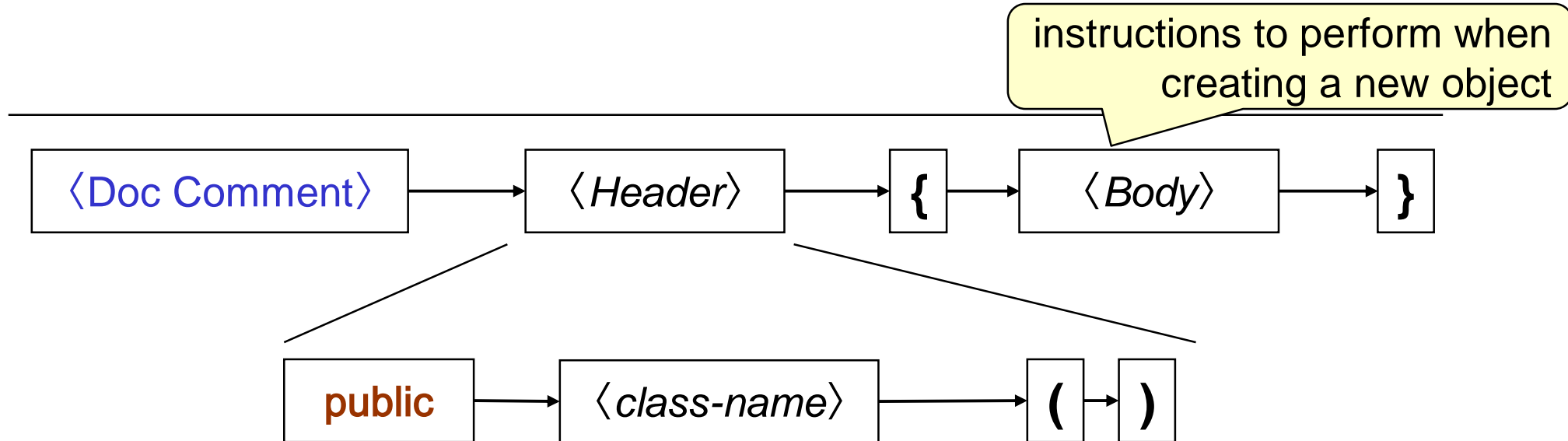
eg `/* double fahrenheit = celsius * 9 / 5 + 32;
 UI.println(celsius + "C is " + fahrenheit + " F"); */`

Constructor Definitions

```

/** Constructor: Set up interface */
public TemperatureCalculator (){
    UI.addButton("Formula", this :: printFormula);
    UI.addButton("F->C", this :: doFahrenheitToCelsius);
}

```



Method Definitions

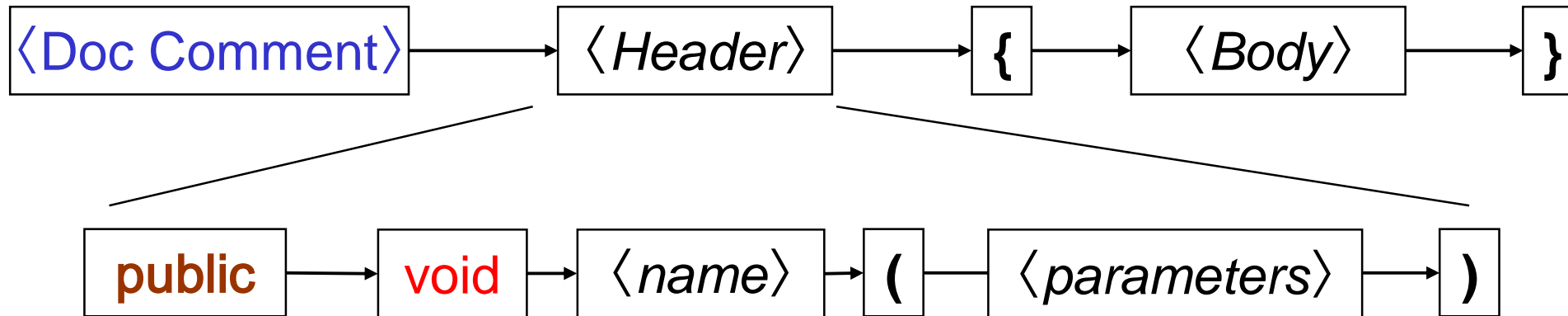
```
/** Print out the conversion formulas */
```

```
public void printFormula ( ) {
```

```
    UI.println("Celsius = (Fahrenheit - 32) *5/9");
```

```
}
```

instructions to perform
this action



Specifying the information
the action needs.
May be empty

“Statements” (instructions)

(Single instructions are called “statements” for silly historical reasons!)

Two important kinds of statements:

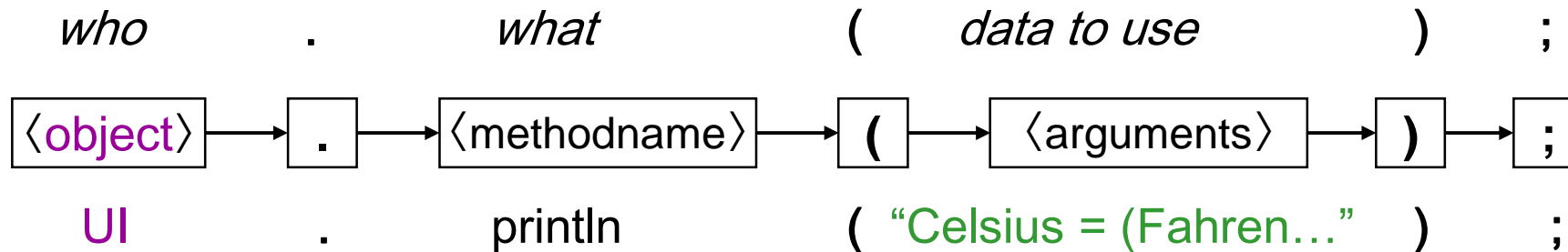
- method call statement:
 - tell some object to perform one of its methods.
 - eg:* tell the UI object to ask the user for a number
 - eg:* tell this object to print the celsius value of a temperature
 - eg:* tell the UI object to print out a string
 - eg:* tell the UI object to add a button
- assignment statement
 - compute some value and put it in a place in memory.

Method Calls

```
/** Print out the conversion formulas */
```

```
public void printFormula(){
    UI.println( "Celsius = (Fahrenheit - 32) *5/9" );
}
```

- Method call Statement:



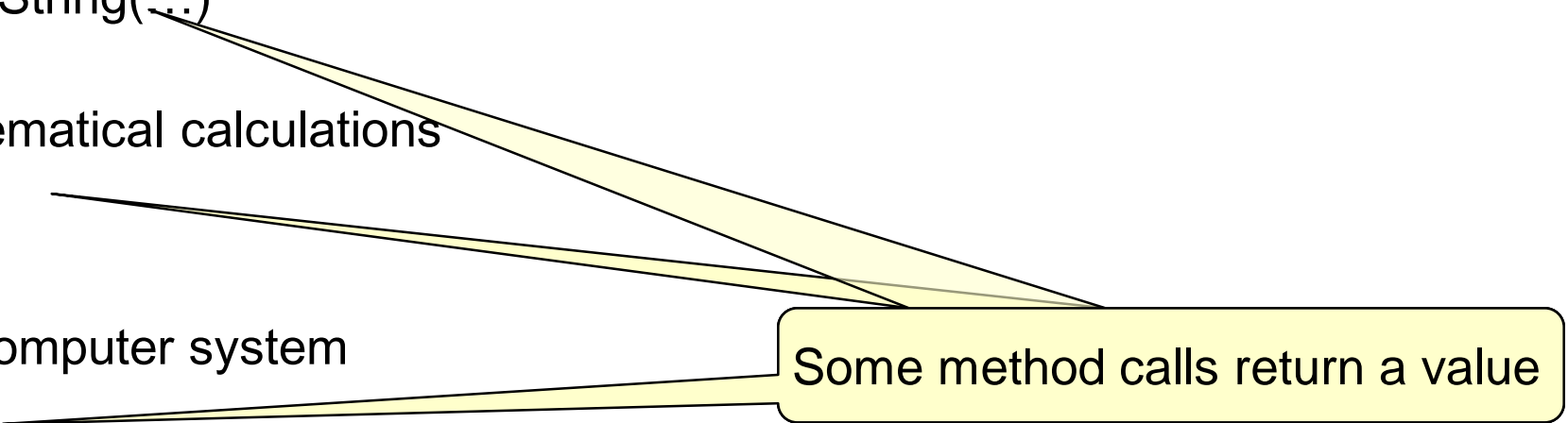
- Meaning of Statement:

- Tell the object
to perform the method
using the argument values provided

Objects and their methods in Java

- What objects are there?

Predefined eg:

- **UI** a "User Interface" window with several panes
→ initialize() quit() addButton(...) println(...) drawRect(...) clearGraphics(),
askDouble(...) askString(...)
 - **Math** methods for mathematical calculations
→ random(), sin(...)
 - **System** representing the computer system
→ currentTimeMillis()
- 
- Some method calls return a value

Others

- **this** The object(s) defined by this class in your program
- New objects that your program creates

Values / Data

There are lots of different kinds ("Types") of values:

- Numbers
 - Integers (**int** or **long**) 42 -194573203 Integer.MAX_VALUE
 - real numbers (**double** or **float**) 42.0 16.43 6.626e-34 Double.NAN, Double.POSITIVE_INFINITY, Double.MIN_VALUE Math.PI
 - ...
- Characters (**char**) 'X' '4'
- Text (**String**) " F -> "
- Colours (**Color**) Color.red Color.green
- Methods (strictly: Lambdas) this::doFahrenheitToCelsius
- Other Objects
- ...