Different Points of View

CS 1025 Computer Science Fundamentals I

Stephen M. Watt

University of Western Ontario

Many points of view

- Before we dive into object-oriented programming, let’s look at a couple of other points of view.
- We will take a quick look at a couple of other ways of programming and few different development environments.
- Don’t worry about getting every detail at this stage, this is more just to get an impression.

Imperative Programming

- This is a C program to compute factorials, n! = n × (n – 1) × ... × 3 × 2 × 1.

```
int factorial(int n) {
   int i, prod;
   prod = 1;
   for (i = 1; i <= n; i++)
        prod = prod * i;
   return prod;
}
```

- The main things to notice are the assignments and the loop.

Functional Programming

- This is a Scheme program to compute n!

```
(define factorial (lambda (n)
        (if (= n 1)
             1
             (* n (factorial (- n 1))) ) ))
```

- Scheme is a member of the Lisp family of languages, the first of which from about 1960.
- It *uniformly* uses the syntax **(operator arg1 arg2 ...).**

- This program uses *recursion* and the fact that, for n > 1,
  n! = n × (n – 1)!

Uniformity *vs* Convenience

- Which is better, to have convenient, but irregular syntax, like `1 + cos(theta)`,

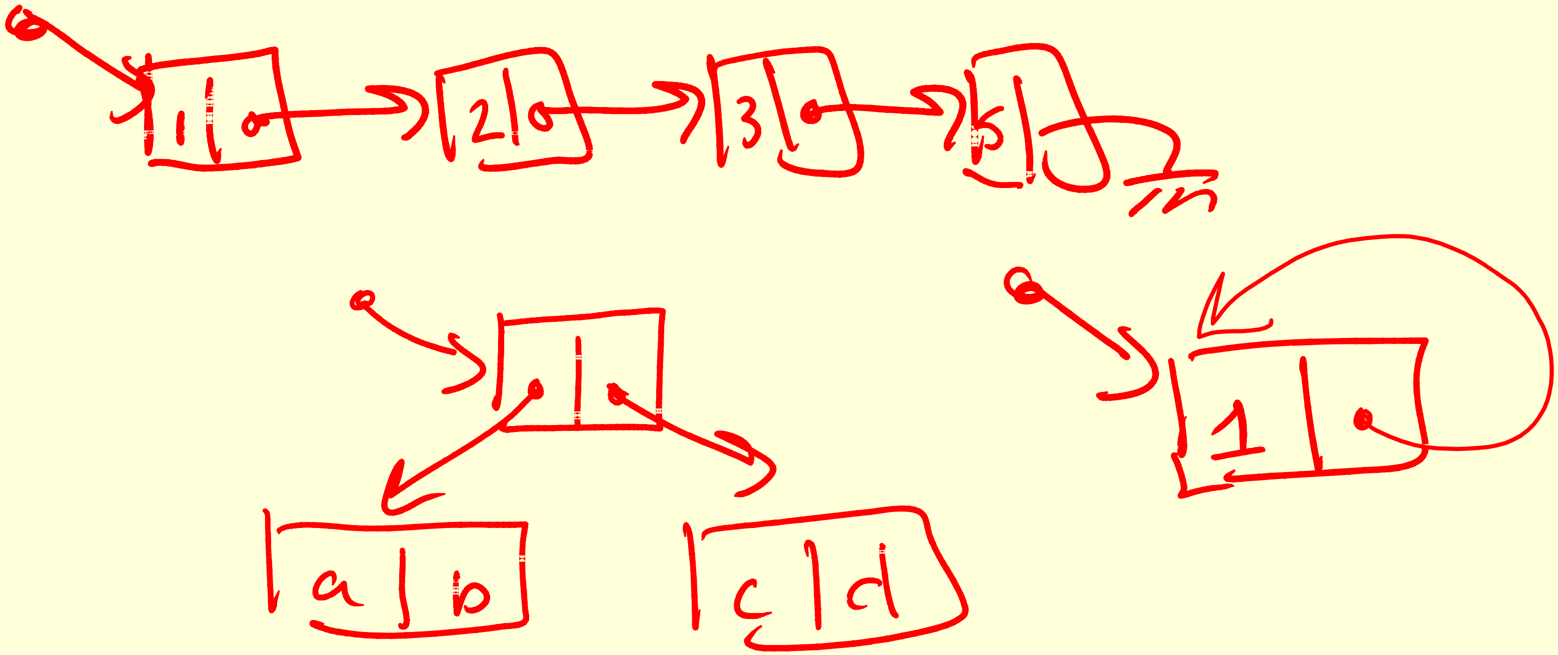
  or to have a completely uniform, but less convenient syntax, like `(+ 1 (cos theta))`.

- The irregular syntax is easier to use and the regular syntax is easier to compose.

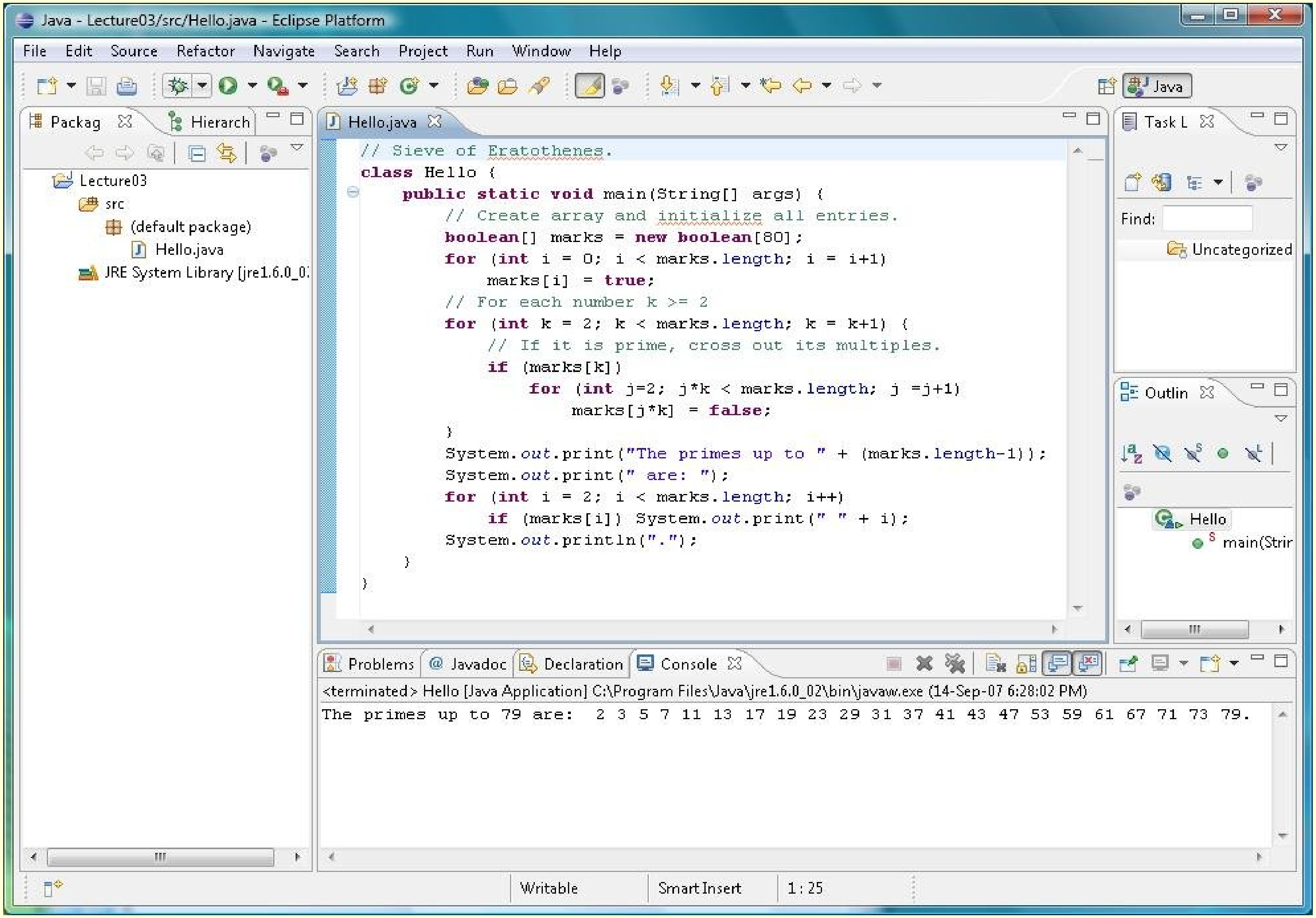
- For single jobs a Swiss Army knife is better than a Lego brick, but *how would you make a replica of the Empire State building out of Swiss Army knives.*

The Lisp family of languages

- Most things are made up of “cons” cells that contain two things.
- The things they contain can be values or pointers to other things.
- Can make complicated data structures from these.



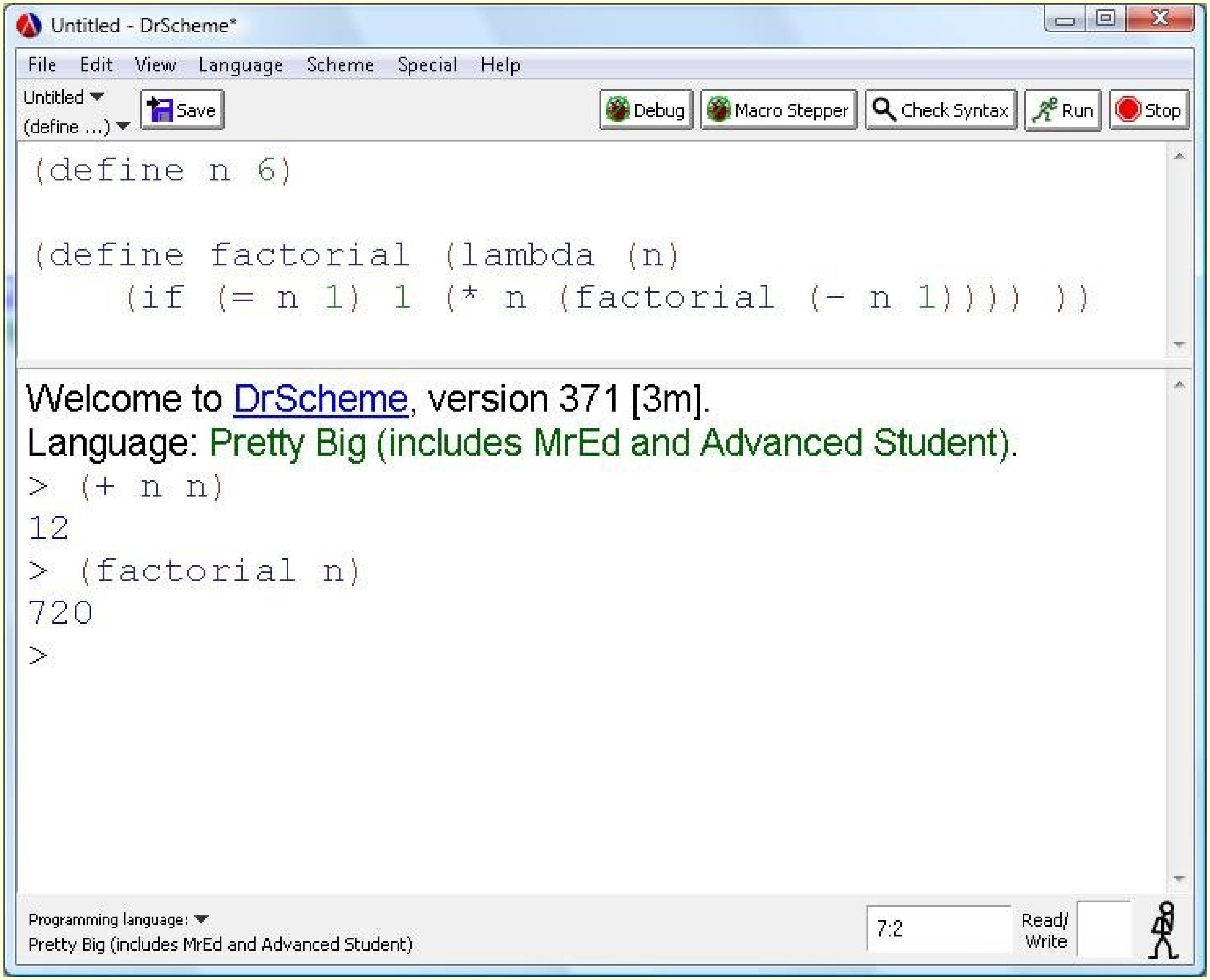
The *Eclipse* IDE for Java Development



```
// Sieve of Eratothenes.
class Hello {
    public static void main(String[] args) {
        // Create array and initialize all entries.
        boolean[] marks = new boolean[80];
        for (int i = 0; i < marks.length; i = i+1)
            marks[i] = true;
        // For each number k >= 2
        for (int k = 2; k < marks.length; k = k+1) {
            // If it is prime, cross out its multiples.
            if (marks[k])
                for (int j=2; j*k < marks.length; j =j+1)
                    marks[j*k] = false;
        }
        System.out.print("The primes up to " + (marks.length-1));
        System.out.print(" are: ");
        for (int i = 2; i < marks.length; i++)
            if (marks[i]) System.out.print(" " + i);
        System.out.println(".");
    }
}
```

```
<terminated> Hello [Java Application] C:\Program Files\Java\jre1.6.0_02\bin\javaw.exe (14-Sep-07 6:28:02 PM)
The primes up to 79 are:  2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79.
```

The *DrScheme* IDE for Scheme



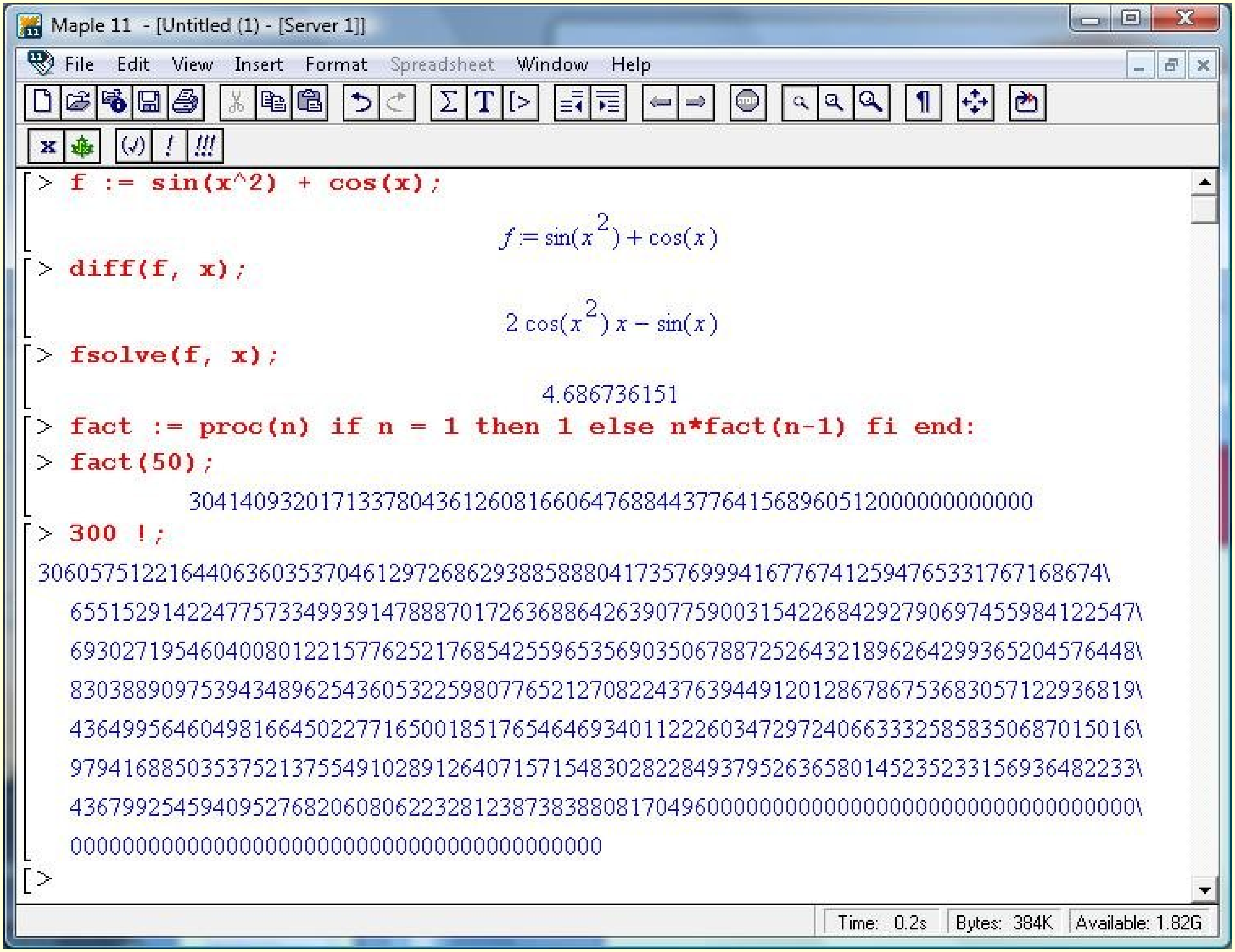
```
(define n 6)

(define factorial (lambda (n)
     (if (= n 1) 1 (* n (factorial (- n 1)))) ))
```

```
Welcome to DrScheme, version 371 [3m].
Language: Pretty Big (includes MrEd and Advanced Student).
> (+ n n)
12
> (factorial n)
720
>
```

Programming language:
Pretty Big (includes MrEd and Advanced Student)

The *Maple* Environment


Maple 11 - [Untitled (1) - [Server 1]]
File Edit View Insert Format Spreadsheet Window Help
Time: 0.2s
Bytes: 384K
Available: 1.82G

```
> f := sin(x^2) + cos(x);
```

$$f := \sin(x^2) + \cos(x)$$

```
> diff(f, x);
```

$$2\cos(x^2)\,x - \sin(x)$$

```
> fsolve(f, x);
```

$$4.686736151$$

```
> fact := proc(n) if n = 1 then 1 else n*fact(n-1) fi end:
> fact(50);
            30414093201713378043612608166064768844377641568960512000000000000
> 300 !;
3060575122164406360353704612972686293885888041735769994167767412594765331767168674\
   6551529142247757334993914788870172636886426390775900315422684292790697455984122547\
   6930271954604008012215776252176854255965356903506788725264321896264299365204576448\
   8303889097539434896254360532259807765212708224376394491201286786753683057122936819\
   4364995646049816645022771650018517654646934011222603472972406633325858350687015016\
   9794168850353752137554910289126407157154830282284937952636580145235233156936482233\
   4367992545940952768206080622328123873838808170496000000000000000000000000000000000\
   00000000000000000000000000000000000000000
>
```