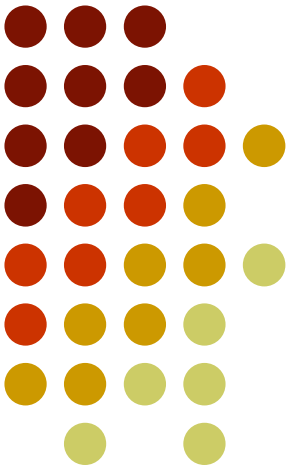
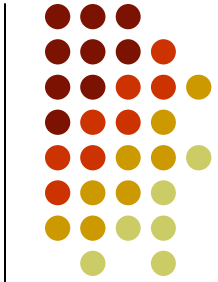
History of Programming Languages

CS181: Programming Languages

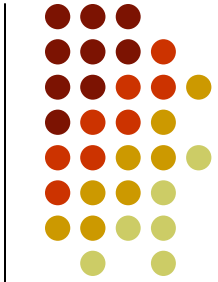


Topics:



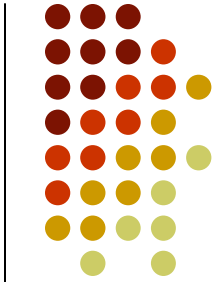
- Historical overview of features introduced by various programming languages
- Code examples
- Family tree of programming languages
- Programming languages naming patterns

Plankalkül



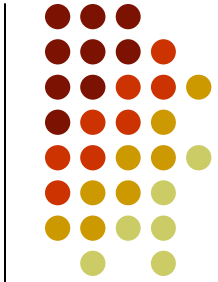
- 1942-45, Konrad Zuse
- Used to program his Z4 computer
- Introduced:
  - the assignment operation
  - if's (but no else's)
  - loops

Fortran



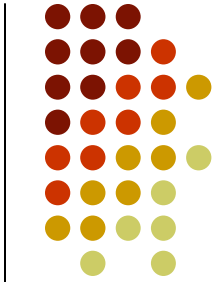
- 1954-57, J. Backus
- Numeric computing
- Introduced:
  - Parameter pass by value
  - Static allocation
  - Separate compilation (because hardware failures were very frequent, length of a program could not exceed 300/400 lines) (FORTRAN II)
  - Modularity (separately developed subprograms)
  - Sharing of data among modules via a global environment
- Still in existence today, mostly in science/academia

Fortran



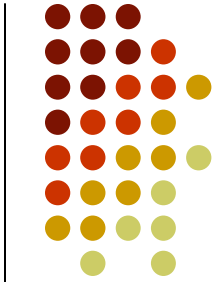
```
PROGRAM Rad
! Simple FORTRAN program
    REAL P,R,C
    IF (.NOT. (R = 0.0)) THEN
          P = 3.1415926
          R = 2.5
          C = P * R
          PRINT *, "C = ", C
    END IF
END
```

Algol60



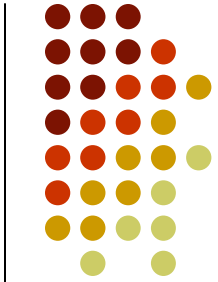
- 1958-60
- Numeric computing
- Descended from Fortran
- Introduced:
  - Stack allocation (Algol58)
  - Stack dynamic variables
  - Compound statements (group statements into one) (Algol58)
  - Explicit typing (Algol58)
  - BNF (Backus-Naur Form) was used to describe Algol60's syntax
  - Block structure
  - Block nesting with scope
  - Recursive procedures
- Spawned numerous other languages

Algol60



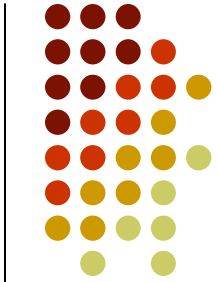
```
begin
    integer N;
    Read Int(N);

    begin
        real array Data[1:N];
        real sum, avg;
        integer i;

        sum:=0;
        for i:=1 step 1 until N do
            begin real val;
                Read Real(val);
            end;
        …
    end
end
```

Cobol



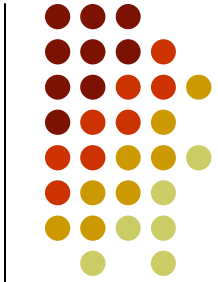
- COBOL (1960)
- Business data processing
- Introduced:
  - Structures (records)
  - Files
  - Macros
  - Comments
  - Programming in a quasi-natural language (see sample code)
- Focuses on moving and formatting data, rather than on heavy computation
- Still in existence today, mostly in the business environment

Cobol



```
IDENTIFICATION DIVISION
PROGRAM-ID.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
        SELECT INP-DATA ASSIGN TO INPUT.
        SELECT RESULT-FILE ASSIGN TO OUTPUT.
DATA DIVISION.
FILE SECTION.
    FD INP-DATA LABEL RECORD IS OMITTED.
    01 ITEM-PRICE
    02 ITEM PICTURE X(30).
    02 PRICE PICTURE 9999V99.
    02 FILLER PICTURE X(44).
FD RESULT-FILE LABEL RECORD IS OMITTED.
```

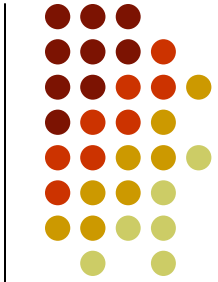
Fortran and Cobol



- “A computer without FORTRAN and COBOL is like a chocolate cake without mustard and ketchup.”

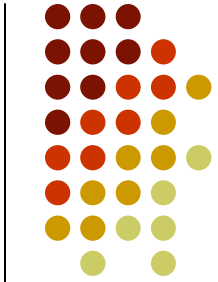
  – Internet wisdom

Lisp



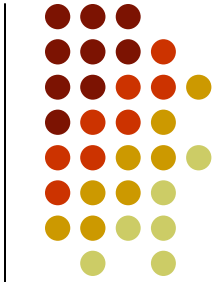
- 1962, J. McCarthy
- Symbolic computing
- AI (mostly in the US)
- Introduced:
  - Garbage collection
  - Heap allocation
- Father and mother of all functional languages
- Free from Von Neumanean notions
  of variables, assignments, goto's etc.
- One data structure available (and needed) – a list
- LISP interpreters are simple to write and difficult to execute

Lisp



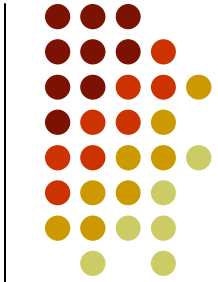
```
;; Simple factorial routine
(defun fact1 (num)
   (cond ((not (integerp num)) nil)
   ((<= num 1) 1)
   (t (* num (fact1 (- num 1)))))
)
```

PL/I



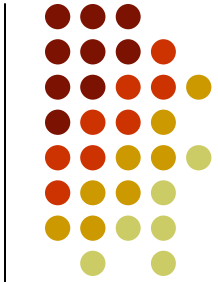
- 1964, IBM
- Name is an abbreviation for “Programming Language 1”
- General purpose programming language, all unifying
- Descended from Fortran, Algol60 and Cobol
- Introduced:
  - Exception handling
  - Pointer datatype
  - Multitasking facilities
- Large and complex

PL/I



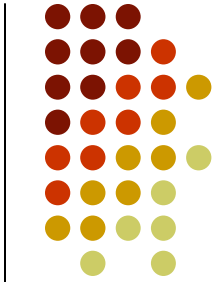
```
FINDSTRINGS: PROCEDURE OPTIONS(MAIN)
    DECLARE PAT VARYING CHARACTER(100),
    LINEBUF VARYING CHARACTER(100),
    (LINENO, NDFILE, IX) FIXED BINARY;

    NDFILE = 0; ON ENDFILE(SYSIN) NDFILE=1;
    GET EDIT(PAT) (A);
    LINENO = 1;
    DO WHILE (NDFILE=0);
        GET EDIT(LINEBUF) (A);
        IF LENGTH(LINEBUF) > 0 THEN DO;
            IX = INDEX(LINEBUF, PAT);
        END;
    END;
END FINDSTRINGS;
```

BASIC



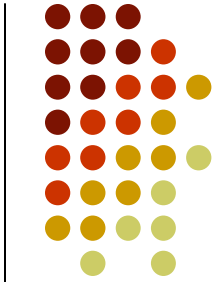
- BASIC (1964), J. Kemeny and T. Kurz
- Educational
- Interactive
- Descended from FORTRAN
- Simple syntax
- Limited data structures
- Arrays start with index 1
- Ancestor of Visual BASIC

Basic



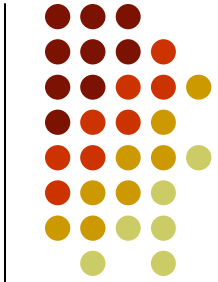
```
   REM Very very simple QBasic program
   PRINT "Press 1 to clear the screen, or 2 to say
   'Hello'!“
   INPUT "What do you want to do"; choice
   IF choice = 1 THEN GOTO clrscr
   IF choice = 2 THEN GOTO hello
clrscr: CLS PRINT "Done." END
hello: PRINT "Hello, hello, hello!“
   END
```

Simula67



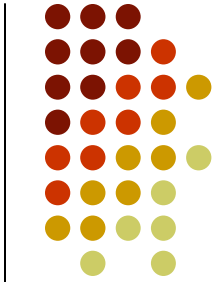
- 1967, O.-J. Dahl
- Simulation problems
- Descended from ALGOL 60
- Introduced:
  - Object-oriented Programming (OO)
  - Abstract data types
  - Classes
  - Inheritance

Simula67



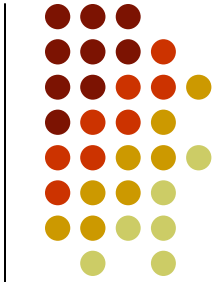
```
Class Line(a,b,c);
real a,b,c;
begin boolean procedure parallelto(l); ref(Line) l;
    if l =/= none then
            parallelto := abs(a*l.b - b* l.a) < 0.00001;

    ref(Point) procedure meets(l); ref(Line) l;
            begin real t;
                    if l =/= none and ~parallelto(l) then
                            begin
                                    t := 1/(l.a * b - l.b * a);
                                    meets :- new Point(..., ...);
                            end;
                    end;
…
```

Algol68



- 1968
- General purpose programming language
- An improvement of ALGOL 60
- Formal language specification
- Pure, used in academia, but not really user friendly
- Introduced:
  - Operator overloading
  - Orthogonality
  - User-defined data types
  - References
  - Variable declaration anywhere in a block

Pascal

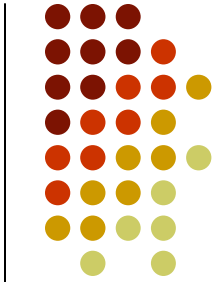


- 1971, Niklaus Wirth
- Named after Blaise Pascal
- general purpose
- Educational
- Simple
- Enforces programming discipline

  “GOTO statements make spaghetti out of a program”
  – Edsger Dijkstra

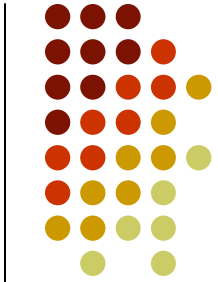
- Lived through many reincarnations (modlarity, OO, etc.)

Pascal



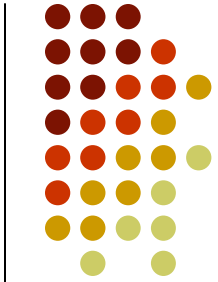
```
program sort_array(input,output);
    const max = 9;
    type integer_array = array[1..max] of integer;
    var I, J, K, swaps, temp : integer;
    test_array : integer_array;

begin
    test_array[1] := 4; test_array[2] := 7; test_array[3] := 9;
    test_array[4] := 3; test_array[5] := 15; test_array[6] := 2;

    writeln(' *** Initial Array ***');
    for I := 1 to (max-1) do
    write('|',test_array[I]:2,' ');
    …
end.
```

Prolog



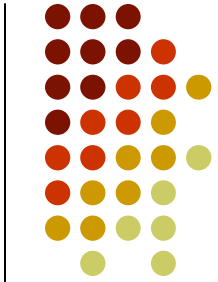
- 1972, A. Colmerauer
- AI (mostly in Europe and Japan)
- more about Prolog in the following lectures

Prolog



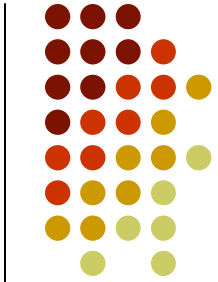
```
split( X,[Y | Tail], [Y | Small], Big) :- gtq( X, Y), !,
        split( X, Tail, Small, Big).
split( X, [Y | Tail], Small, [Y | Big] ) :-
        split( X, Tail, Small, Big).
conc([],L,L).
conc( [X | L1], L2, [X | L3]) :- conc( L1, L2, L3).
```

C



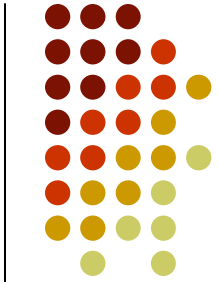
- C (1972), Dennis Ritchie
- Systems programming
- Descended from ALGOL 68
- Tied to the development of UNIX
- Most widely used programming language
- Very, very, very fast

C



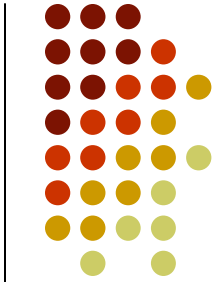
```
#include <stdio.h>

/* count lines of standard input */

main(int argc, char *argv[]) {
    char lbuf[256];
    int lcnt;
    for(lcnt = 0; fgets(lbuf,sizeof(lbuf) - 1, stdin); cnt );
    printf("%d lines\n", lcnt); exit(0);
}
```

Ada



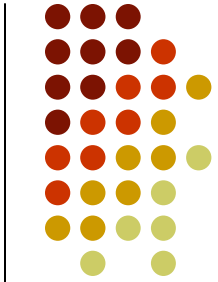
- 1979, J. Ichibiah
- named after Ada Lovelace
- Descended from Pascal
- General purpose programming language
- Embeded systems
- Sponsored by the D.O.D.
- Robust

Ada



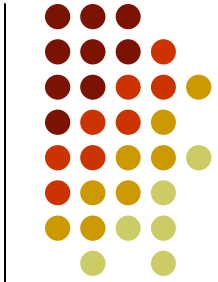
```
-- simple programming with floating-point #s
with Ada.Float_Text_IO;
use Ada.Float_Text_IO;
procedure Think is
    A, B : Float := 0.0;
    I, J : Integer := 1;
    begin
          A := B * 7.0;
          I := J * 3;
          B := Float(I) / A;
    Put(B);
end Think;
```

Smalltalk



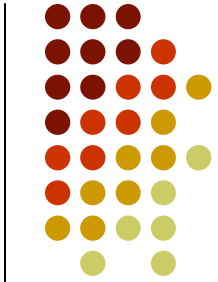
- Smalltalk (1980), A. Kay
- Personal computing
- Fully OO
- Descended from SIMULA 67 and LISP

Smalltalk



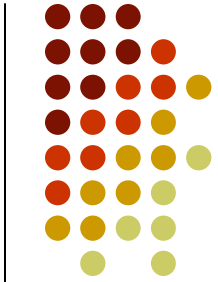
```
Point subclass: #GriddedPoint
        instanceVariableNames: ''
        classVariableNames: ''
        poolDictionaries: ''
        category: 'Exercise11.5'!
!GriddedPoint methodsFor: 'accessing'!
x: xInteger
        "Set the x coordinate gridded to 10 (using
        rounding, alternatively I could use truncating)."
super x: (xInteger roundTo: 10)!
```

C++



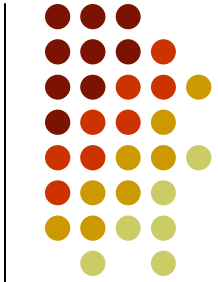
- 1984, Bjarne Stroustrup
- General purpose programming language
- Descended from C, SIMULA 67
- Introduced:
  - Assignment operator overloading

C++



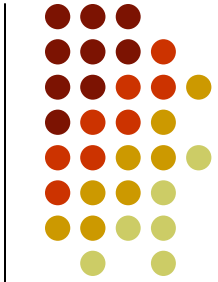
```
#include <iostream.h>
#include <String.h>

main(int argc, char *argv[]) {
    String *s1;
    s1 = new String("Hello World!");
    cout << *s1 << endl << "Length is:" << s1->length()
    << endl;
}
```

ML



- ML (1984), R. Milner
- Symbolic computing
- Descended from LISP
- Inrtoduced:
  - Parametric Polymorphism

ML

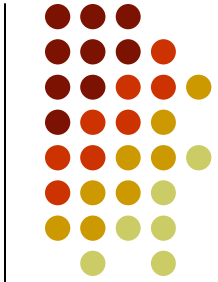


- fun sort nil = nil : int list
  | sort(h::t) = let
  fun insert(i,nil) = [i]
  | insert(i,h::t) = if i>h then i::h::t
  else h::insert(i,t)
  in insert(h, sort t) end;

  fun mean l = let
  fun sl(nil ,sum,len) = sum div len
  | sl(h::t,sum,len) = sl(t,sum+h,len+h)
  in sl(l,0,0) end;

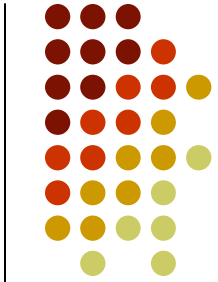
  mean(sort [2,3,5,7,11,13] @ [6,14,28])

Perl



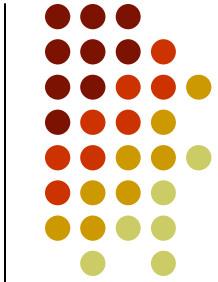
- 1990, Larry Wall
- Scripting language
- Descended from OS shell languages

Perl



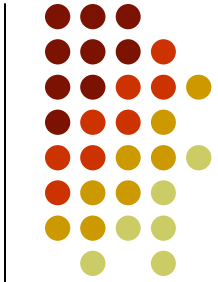
```
#!/usr/bin/perl
$total = 0;
sub sumcolumn {
    my $col = shift;
    my $lin = shift;
    my @fields;
    if ($lin) {
        @fields = split(/:/,$lin);
        $total = $fields[2];
    }
 }

while (<>)
    sumcolumn(3,$_);
print "Total of column 3 is $total\n";
```

Python



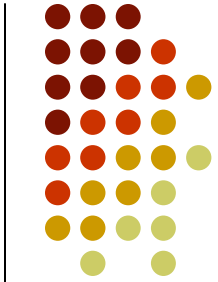
- 1991, G. van Rossum
- named after *Monty Python's Flying Circus*, a legendary BBC show (the show was in turn named after the WWII Ally pilot who shot down most Ally fighter planes)
- descended from OS shell languages
- scripting language with a fanatic following (which certainly holds for UCR)

Python



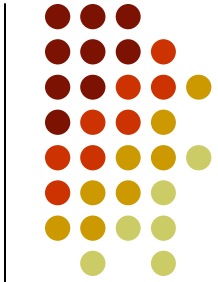
```
class binary_tree:
    def init (self):
            self.tree = None

    def    insert (self, key):
            if self.tree:
                    self._insert (self.tree, key)
            else:
                    self.tree = node(key)

    def _insert (self, tree, key):
            if key < tree.key:
                    if tree.left:
                            self._insert (tree.left, key)
                    else:
                            tree.left = node(key)
...
```

Java

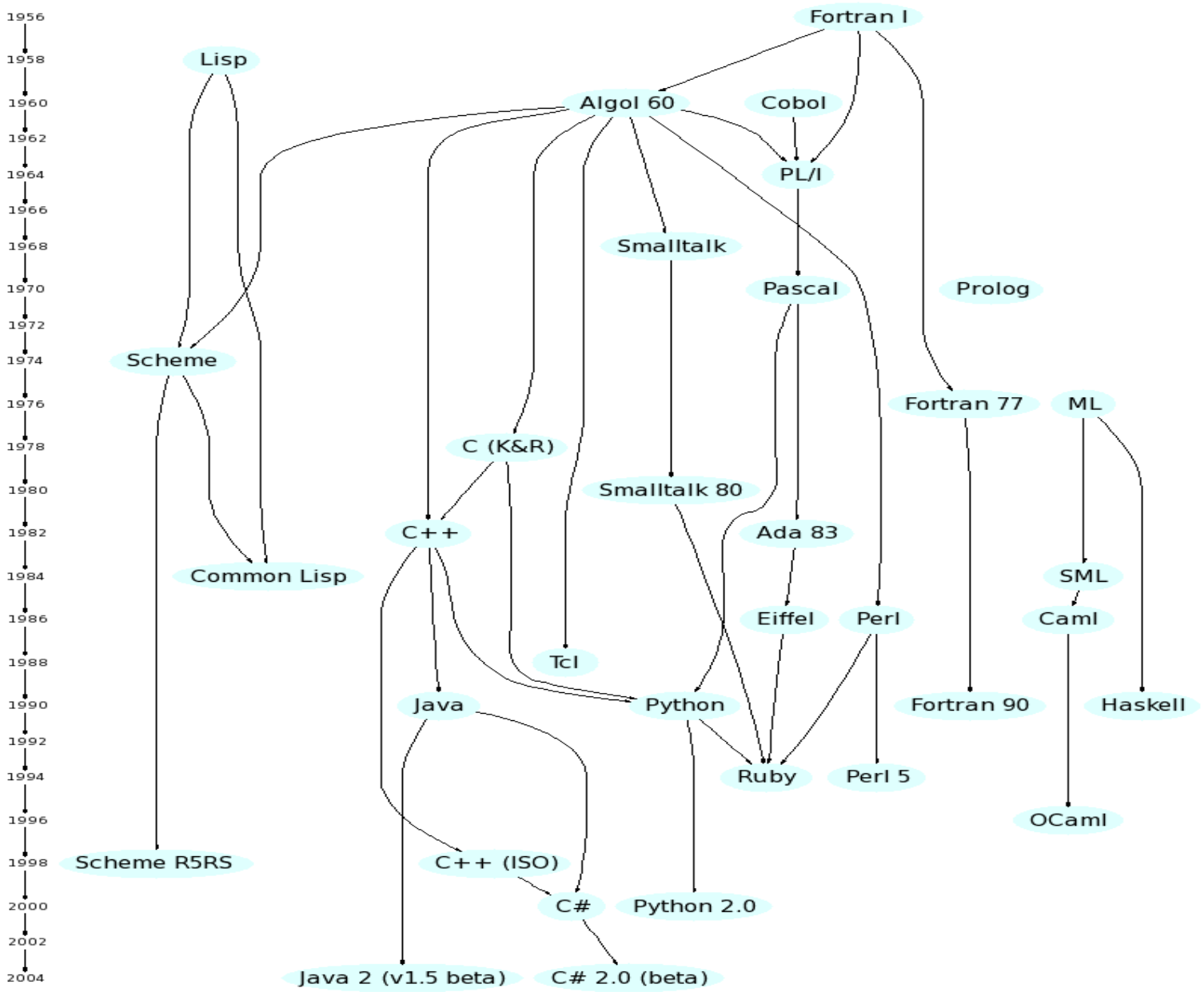


- 1995, Sun Microsystems (led by James Gosling)
- Descended from C++
- Built-in support for network computing

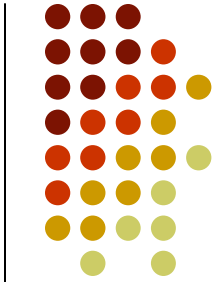
Java



```
import java.awt.*;
import java.util.*;

public class Showtime extends Frame implements Runnable {
    Button quitBtn;
    Thread tthread;

    public Showtime() {
        quitBtn = new Button("Quit");
        add(quitBtn);
        show();
        tthread = new Thread(this);
        tthread.run();
    }
    …
}
```

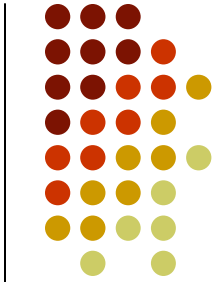

1956
1958
1960
1962
1964
1966
1968
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
Fortran I
Lisp
Algol 60
Cobol
PL/I
Smalltalk
Pascal
Prolog
Scheme
Fortran 77
ML
C (K&R)
Smalltalk 80
C++
Ada 83
Common Lisp
SML
Eiffel
Perl
Caml
Tcl
Java
Python
Fortran 90
Haskell
Ruby
Perl 5
OCaml
Scheme R5RS
C++ (ISO)
C#
Python 2.0
Java 2 (v1.5 beta)
C# 2.0 (beta)

Naming patterns



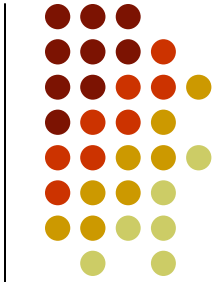
- Acronyms, abbreviations:
  - ALGOL (**ALGO**rithmic **L**anguage)
  - APL (**A** **P**rogramming **L**anguage)
  - BASIC (**B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode)
  - BCPL (**B**asic **C**ombined **P**rogramming **L**anguage)
  - BLISS (**B**asic/**B**ill's **L**anguage for **I**mplementation of **S**ystem **S**oftware)
  - COBOL (**CO**mmon **B**usiness **O**riented **L**anguage)
  - DYLAN (**DY**namic **LAN**guage)
  - FORTRAN (**FOR**mula **TRAN**slator)
  - IAL (**I**nternational **A**lgorithmic **L**anguage, the original 1958 name of ALGOL)

Naming patterns



- More acronyms, abbreviations:
  - LISP (**LIS**t **P**rocessor)
  - ML (**M**eta **L**anguage)
  - PL/I (**P**rogramming **L**anguage One [**I**])
  - Prolog (**PRO**grammation en **LOG**ique)
  - SEQUEL (**S**tructured **E**nglish **QUE**ry **L**anguage, the original name of SQL before another SEQUEL language was noticed)
  - Simula (**SIMU**lation **LA**nguage)
  - SQL (Structured Query Language)
  - TCL (Tool Command Language)
  - WSFN (**W**hich **S**tands **F**or **N**othing - ever heard of this one?)

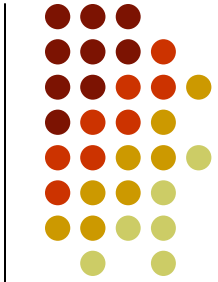
Naming patterns



- Food and beverage:
  - Java (name thought up during a brainstorming session. Original name was Oak.)
  - Pizza (needs no explanation)

- Pop culture:
  - Brenda (A subset of Dylan, written in Dylan. Used as a programming assignment in the Cornell University data structures course around the same time Beverly Hills 90210 was popular)

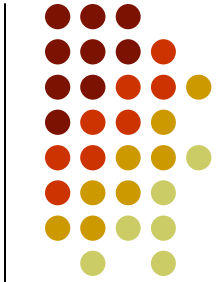
Naming patterns



- Self-descriptive names:
  - LISP (Lots of Irritating Silly Parentheses or Lost In a Sea of Parentheses)
  - Modula ("modular" language)

- Precious objects:
  - Perl (Its name actually was "Pearl" for a short time, until Larry saw a reference to a graphics language called "pearl", plus 4 letters are better than 5.)
  - Ruby

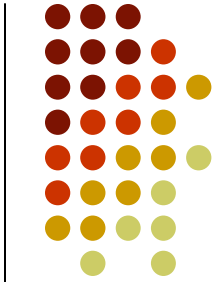
Naming patterns



- Mathematicians:
  - Pascal (after Blaise Pascal)
  - Ada (after Countess Ada Lovelace)
  - Goedel (logic programming language, named for Kurt Goedel)

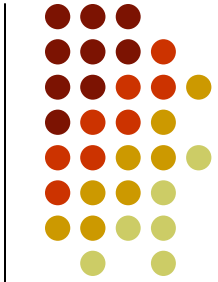
- Literature:
  - Oberon (from the king of the fairies in Shakespeare's *Midsummer Night's Dream*)
  - Miranda (from a character in Shakespeare's *Tempest*; it's also the Latin for "admirable")
  - Oz Language (from *The Wizard of Oz*)
  - Alice Language (from *Through The Looking Glass*)

Naming patterns



- Names derived from earlier languages:
  - BCPL (derived from CPL)
  - B (derived from BCPL, first letter in acronym)
  - C (derived from BCPL, second letter in acronym, or more obviously, next letter in alphabet)
  - C++ (derived from C; there was debate as to whether this should be called **D** or **P**)
  - C# (derived from C++)
  - D (next letter after C)
  - JavaScript (from Java Language. Not that JavaScript actually has anything to do with Java, but the Netscape marketing department wanted to attach some of the glamour that Java then had to their new language, Mocha)

References:



- *Dictionary of Programming Languages*. On line. http://cgibin.erols.com/ziring/cgi-bin/cep/cep.pl
- Rigaux, Pascal. *Tree of programming languages history*. On line. http://merd.sourceforge.net/pixel/language-study/diagram.html
- Programming Language Naming Patterns. http://c2.com/cgi/wiki?ProgrammingLanguageNamingPatterns