Chapter 1: History & Background

CMSC 124, 1st Semester, AY 2009-10

Something to Ponder

What comes to your mind whenever you hear “PROGRAMMING LANGUAGE”?

What is your favourite PL?

Something to Ponder

Chapter 1: History & Background

Definition

Programming Language

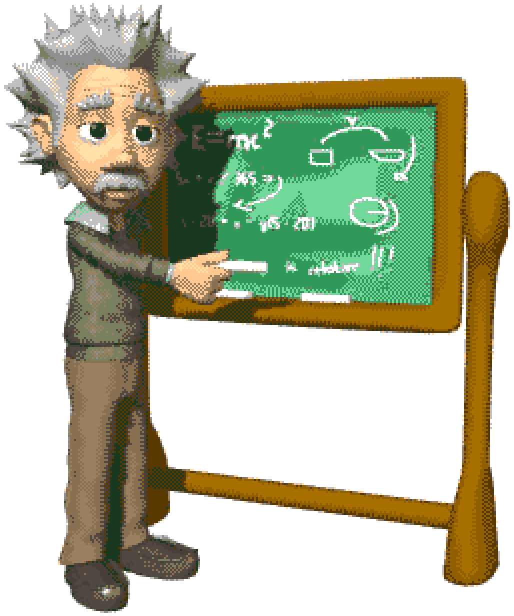
- System for describing computation
- System of signs to communicate a task/algorithm to a computer, causing the task to be performed.

— You're fluent in twenty-four programming languages, but you can't even talk about the weather with me!"
Igor Aleshin
DATAART
Enjoy IT

Chapter 1: History & Background

Why Study Programming Languages? Why Do We Need To?

1. **TO IMPROVE ability to develop effective algorithms.**



2. TO IMPROVE use of existing PL's.

Why Study Programming Languages? Why Do We Need To?

3. TO INCREASE your vocabulary of useful programming constructs.

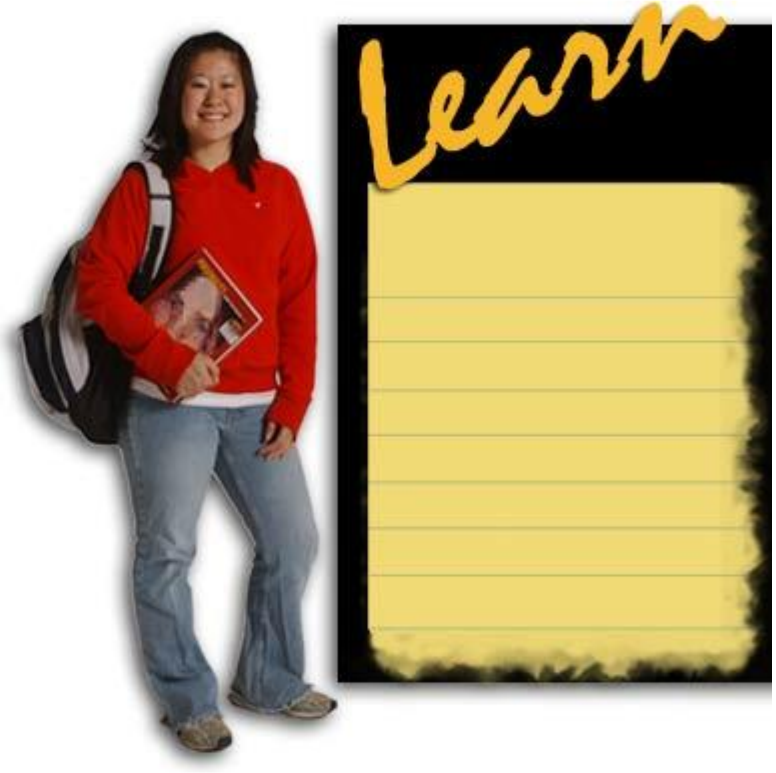
Question: Why do we study a natural language such as English?

4. TO ALLOW a better choice of PL.

Example (Analogy):

- Choosing the right language in a certain place.
- Choosing the right jutsus/techniques in Naruto.

5. TO MAKE it easier to learn a new PL.



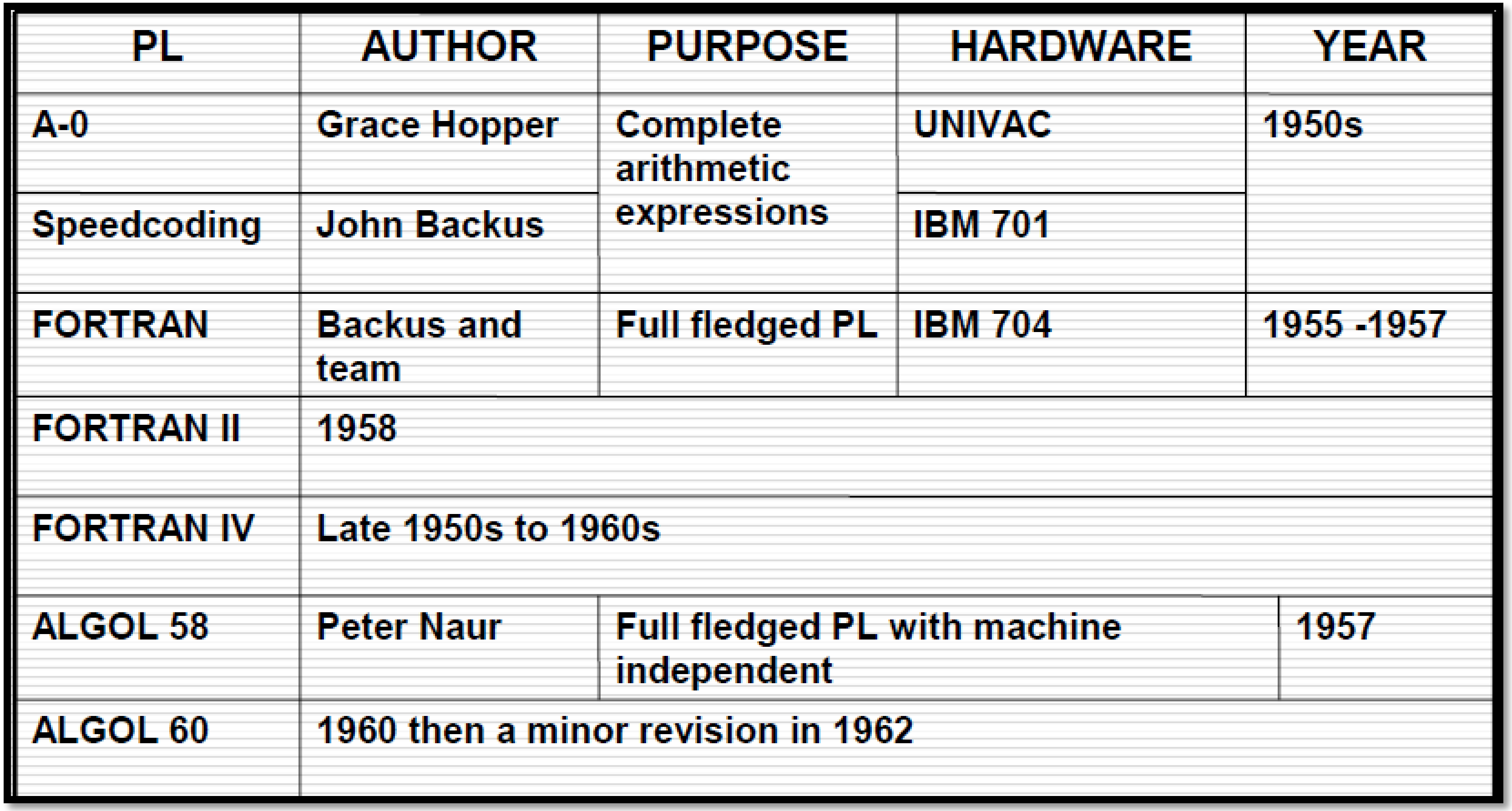
Chapter 1: History & Background

1950-1970

- Numerically based languages
- Business languages
- Artificial intelligence languages
- Systems languages

Chapter 1: History & Background

Numerically Based PL



PL	AUTHOR	PURPOSE	HARDWARE	YEAR
A-0	Grace Hopper	Complete arithmetic expressions	UNIVAC	1950s
Speedcoding	John Backus		IBM 701	
FORTRAN	Backus and team	Full fledged PL	IBM 704	1955 -1957
FORTRAN II	1958			
FORTRAN IV	Late 1950s to 1960s			
ALGOL 58	Peter Naur	Full fledged PL with machine independent		1957
ALGOL 60	1960 then a minor revision in 1962			

Chapter 1: History & Background

Numerically Based PL

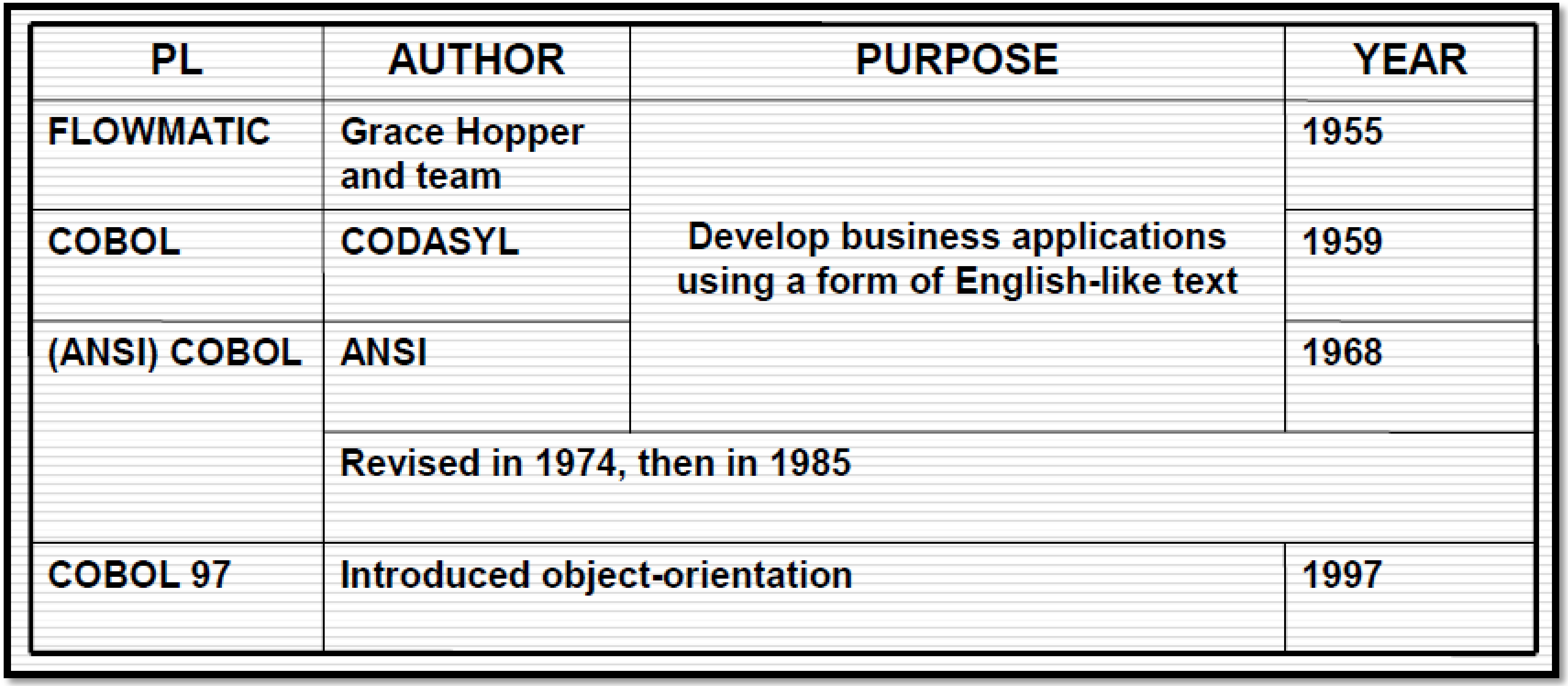
Examples:

1. **FORTRAN (FORmula TRANslation)**
   - 1st successful PL.
   - Designed specifically for scientific & engineering applications.
   - **FORTRAN-I:** Considered a milestone in the history of computing.

2. **ALGOL-60 (ALgorithmic Language)**
   - Aimed to improve FORTRAN
   - Became the basis of almost all block-structured language.

Chapter 1: History & Background

Business PL



PL	AUTHOR	PURPOSE	YEAR
FLOWMATIC	Grace Hopper and team	Develop business applications using a form of English-like text	1955
COBOL	CODASYL		1959
(ANSI) COBOL	ANSI		1968
	Revised in 1974, then in 1985		
COBOL 97	Introduced object-orientation		1997

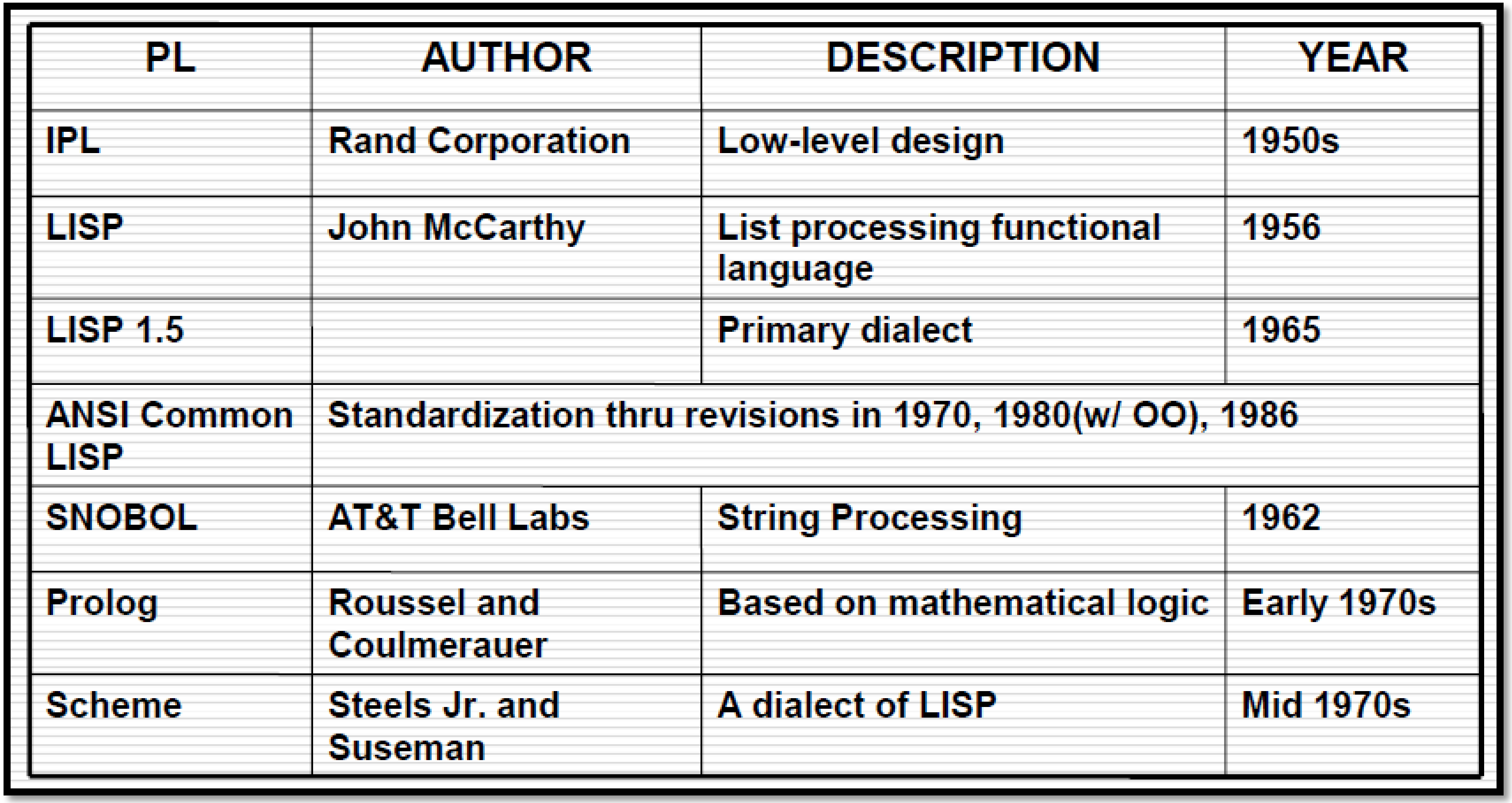
Business PL

Example:

1. **COBOL (COmmon Business Oriented Language)**
   - A fabulous language.
   - Heavily supported by the US government.
   - Heavily structured data definitions that looks like the English language.
   - Self-documenting.
   - Example: COBOL code.

Chapter 1: History & Background

Artificial Intelligence PL



PL	AUTHOR	DESCRIPTION	YEAR
IPL	Rand Corporation	Low-level design	1950s
LISP	John McCarthy	List processing functional language	1956
LISP 1.5		Primary dialect	1965
ANSI Common LISP	Standardization thru revisions in 1970, 1980(w/ OO), 1986		
SNOBOL	AT&T Bell Labs	String Processing	1962
Prolog	Roussel and Coulmerauer	Based on mathematical logic	Early 1970s
Scheme	Steels Jr. and Suseman	A dialect of LISP	Mid 1970s

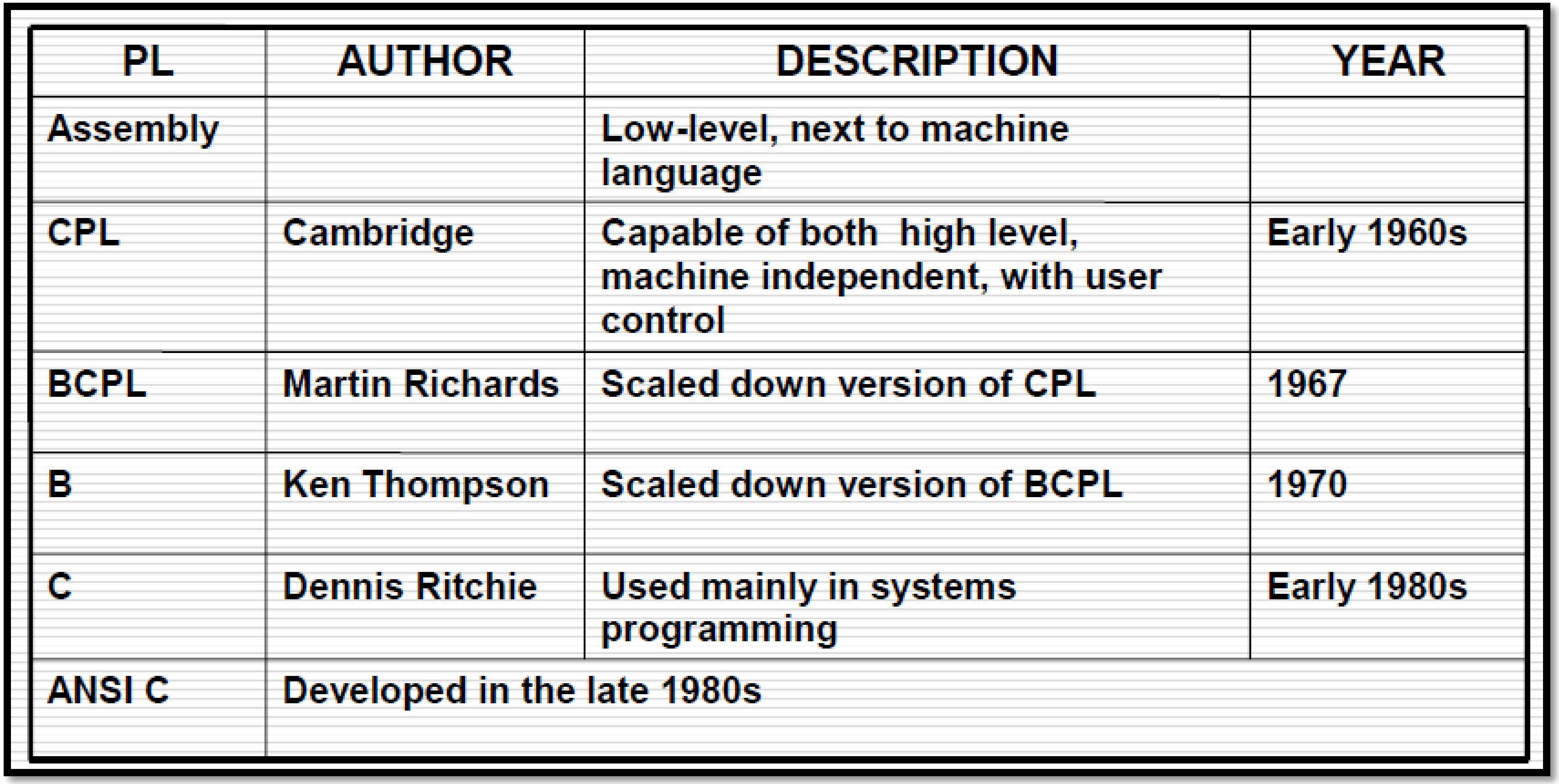
Artificial Intelligence PL

Example:

1. **Lisp**
   - First major language to support list processing.
   - First major language to support recursion.
   - First functional language.

Chapter 1: History & Background

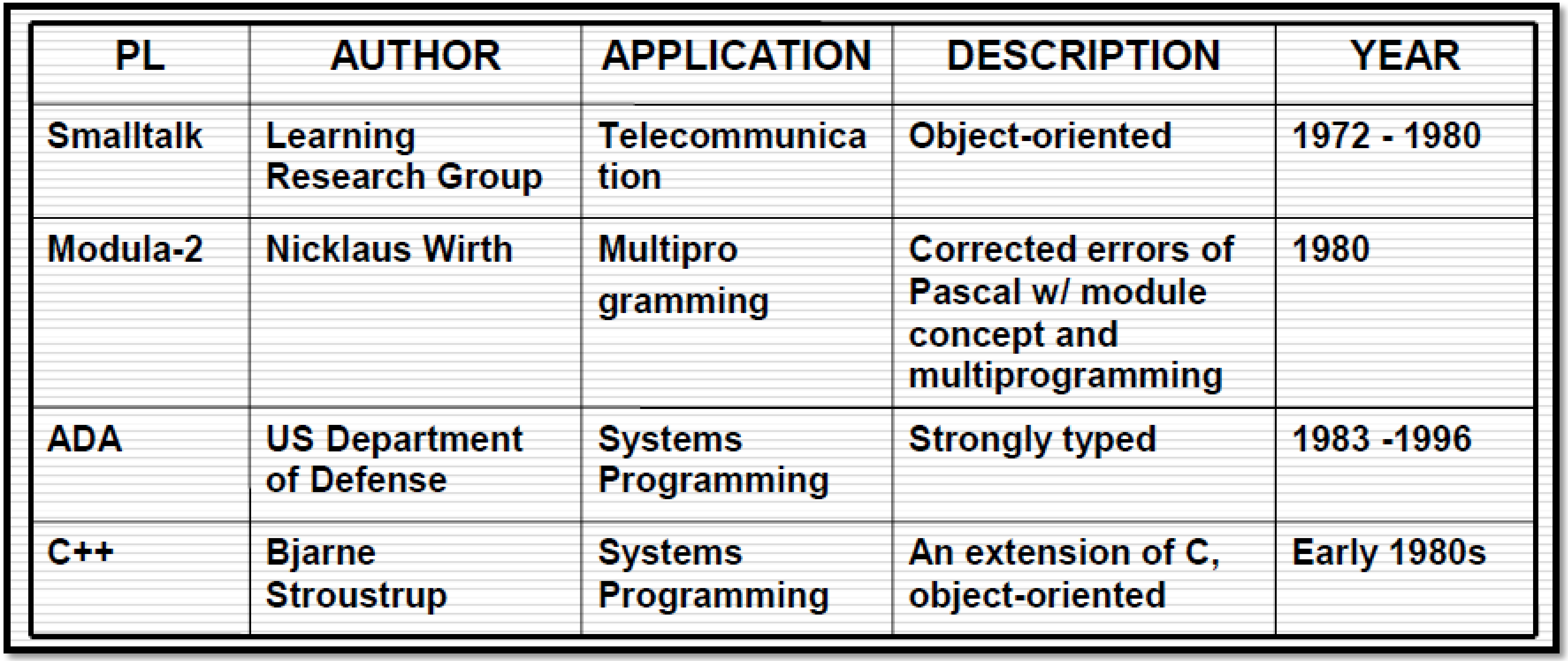
Systems PL



PL	AUTHOR	DESCRIPTION	YEAR
Assembly		Low-level, next to machine language	
CPL	Cambridge	Capable of both high level, machine independent, with user control	Early 1960s
BCPL	Martin Richards	Scaled down version of CPL	1967
B	Ken Thompson	Scaled down version of BCPL	1970
C	Dennis Ritchie	Used mainly in systems programming	Early 1980s
ANSI C	Developed in the late 1980s		

Chapter 1: History & Background

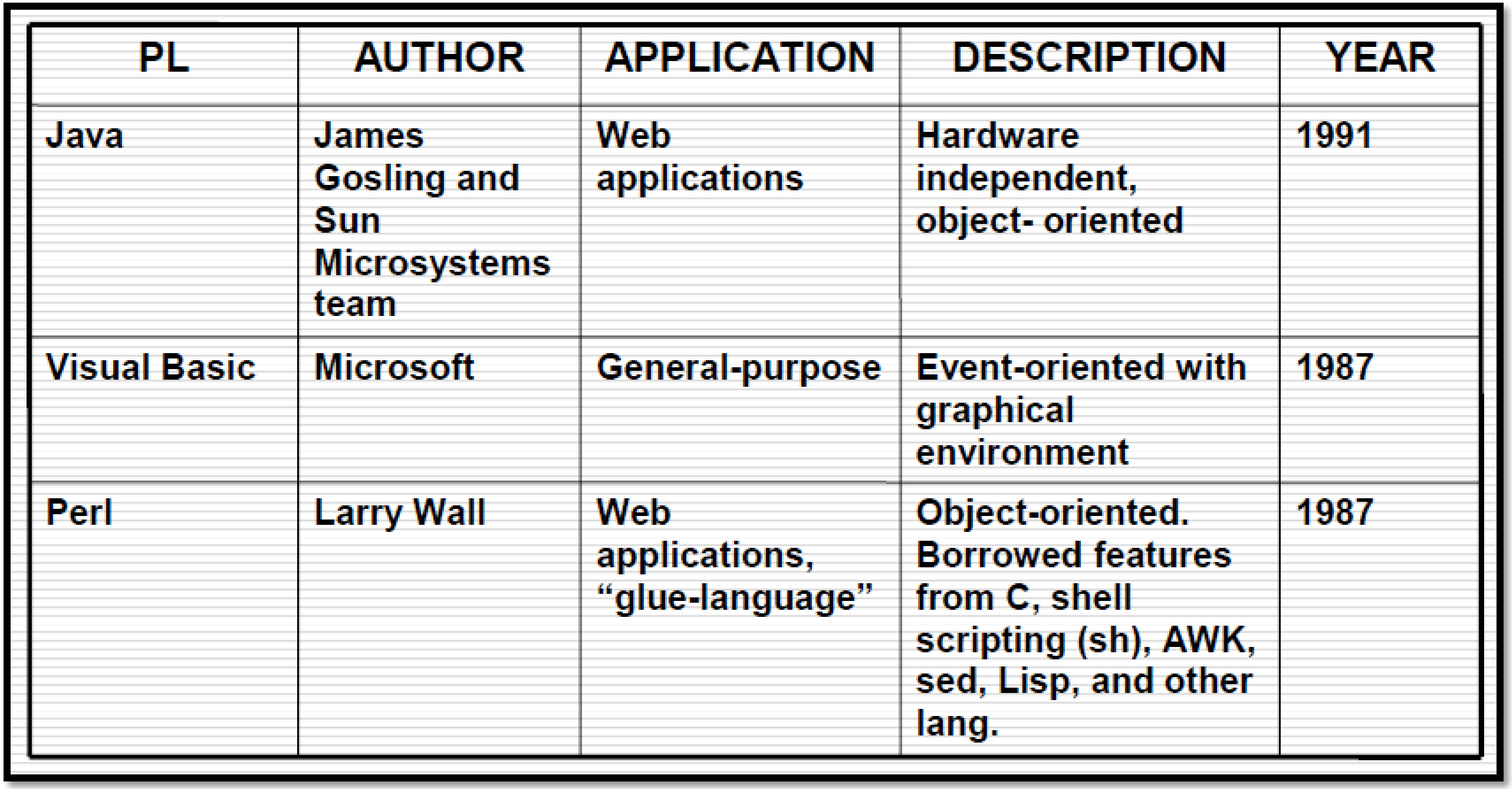
1970-1990



PL	AUTHOR	APPLICATION	DESCRIPTION	YEAR
Smalltalk	Learning Research Group	Telecommunication	Object-oriented	1972 - 1980
Modula-2	Nicklaus Wirth	Multiprogramming	Corrected errors of Pascal w/ module concept and multiprogramming	1980
ADA	US Department of Defense	Systems Programming	Strongly typed	1983 -1996
C++	Bjarne Stroustrup	Systems Programming	An extension of C, object-oriented	Early 1980s

Chapter 1: History & Background

1970-1990



PL	AUTHOR	APPLICATION	DESCRIPTION	YEAR
Java	James Gosling and Sun Microsystems team	Web applications	Hardware independent, object- oriented	1991
Visual Basic	Microsoft	General-purpose	Event-oriented with graphical environment	1987
Perl	Larry Wall	Web applications, “glue-language”	Object-oriented. Borrowed features from C, shell scripting (sh), AWK, sed, Lisp, and other lang.	1987

Programming Domains

- Scientific applications
- Business applications
- Artificial intelligence
- Systems programming
- Scripting languages
- Special-purpose languages

Language Evaluation Criteria

1. **Readability**

   "Is it easy to read & understand a program or a portion of it written in the language?"

Chapter 1: History & Background

Language Evaluation Criteria

2. Writability

"Is it easy to write programs in the language?"

Something to Ponder

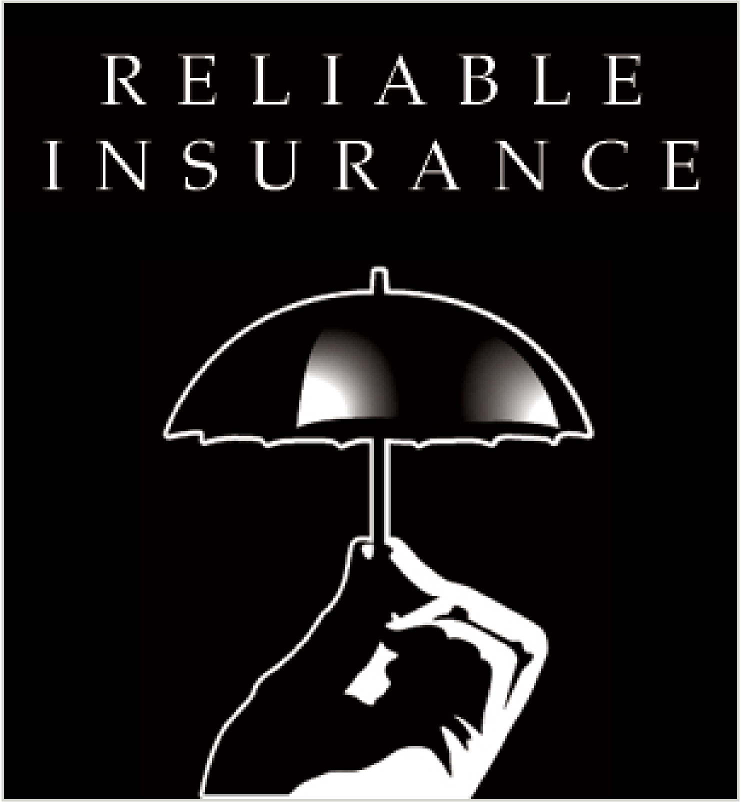
When writability is enhanced, readability suffers.

Something to Ponder

Language Evaluation Criteria

3. Reliability

"Does the program help prevent errors?"



Something to Ponder

“Does the program help prevent errors?” What does that mean?

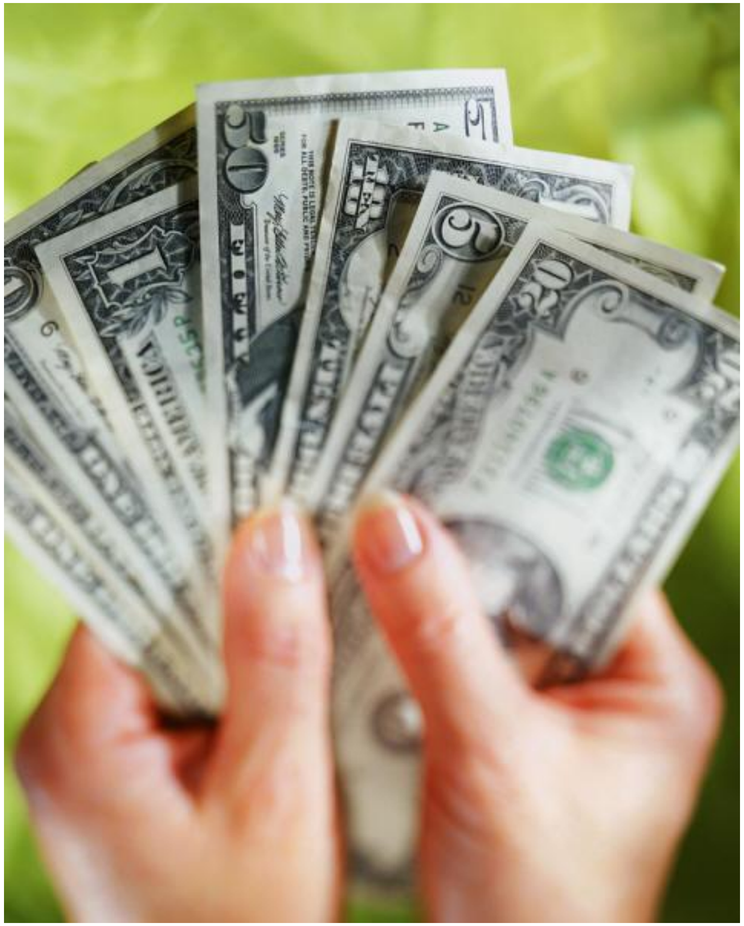
Something to Ponder

Chapter 1: History & Background

Language Evaluation Criteria

4. Cost

"How expensive is it to develop, use, and maintain programs written in the language?"



Chapter 1: History & Background

Programming Languages Classification

- Generations
- Levels of Abstractions
- Paradigms

Chapter 1: History & Background

Programming Languages Classification: Generations

(1) FIRST GENERATION Low-Level Machine Language, Assembly Language	**(2) SECOND GENERATION (early 1960's)** ALGOL-60, BASIC, COBOL, FORTRAN
(3) THIRD GENERATION (late 1960's to present) **Pascal, C, ADA, Java, Eiffel**	**(4) FOURTH GENERATION (domain specific lang.)** **VB, SQL, Access, Excel**

Programming Languages Classification: Levels of Abstraction

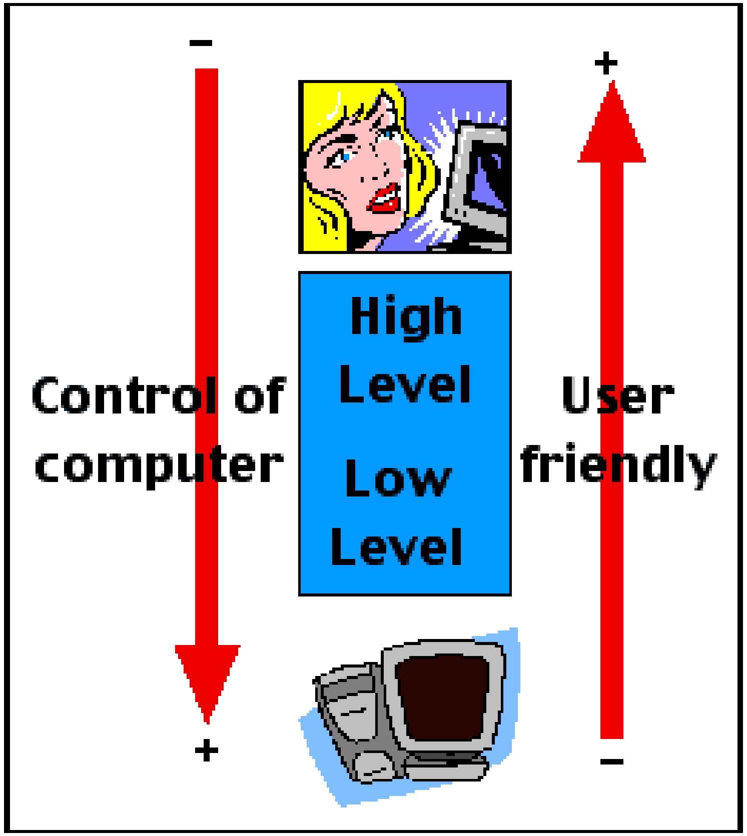
	LOW LEVEL	HIGH LEVEL	VERY HIGH LEVEL
Instructions	Simple machine-like	Expressions and explicit flow of control	Fully abstract machine
Memory Handling	Direct memory access and allocation	Memory access and allocation through operations	Fully hidden memory access and automatic allocation
Examples	Machine, Assembly	C, Java	Logo

Sample LOGO Syntax

```
FORWARD 100
LEFT 90
FORWARD 100
LEFT 90
FORWARD 100
LEFT 90
FORWARD 100
LEFT 90
```

Chapter 1: History & Background

Programming Languages Classification: Levels of Abstraction



Programming Languages Classification: Paradigms

1. Imperative

- "How it is to be achieved"
- To solve a problem, we specify the step-by-step procedure.
- Central features are variables, assignment statements, and iteration

a. Block-Structured

- The procedure is the principal building block of the program.
- Represented by stack
- Examples: Pascal, C

b. Object-Based

- Languages that employ objects.
- An object is a group of procedures that share a state.
- Examples: Java, Modula

Chapter 1: History & Background

Programming Languages Classification: Paradigms

2. Declarative

- "What it is to be achieved"
- Program requires specification of a relation or function.
- Mainly based from math concepts on logic, theory on functions and relational calculus.

a. Logic

- Based on a subset of predicate calculus.
- Axioms and rules are used to deduce new facts.
- Example: Prolog

b. Functional

- Operate only through functions which return one value given a list of parameters.
- Example: Lisp

Something to Ponder

Why were PL’s born?

Why are there so many PL’s out there?

Something to Ponder