

A decorative vertical strip on the left side of the slide, featuring a geometric pattern of overlapping triangles in shades of yellow, white, and magenta.

Programming Syntax and Style

David Greenstein
Monta Vista High School

BEAR FACTS

by Burke



When programmers play Scrabble.

Programming Language Syntax

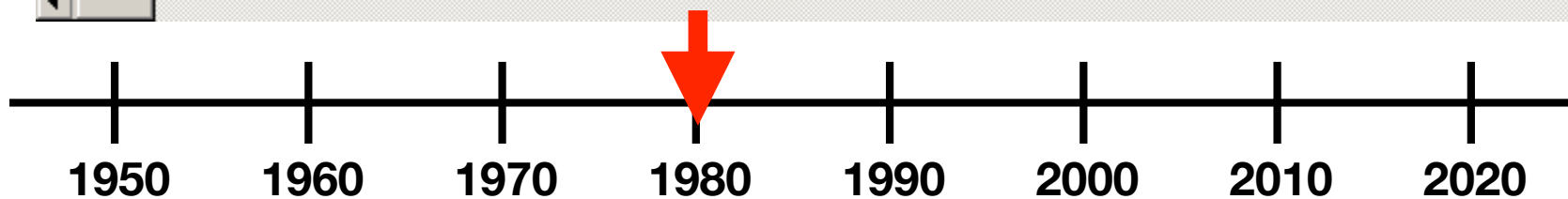
All have:

- Comments
- Programmer-defined Names
- Reserved Words
- Structure

```
/**
 * My First Program
 * @author Mr Greenstein
 * @since 1/1/1976
 */
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!!!");
    }
}
```

Assembly Code (Amiga 68K)

```
CODE:00000048 loc_0_48: ; CODE XREF: start+261j
CODE:00000048 lea aDos_library,a1 ; "dos.library"
CODE:0000004C movea.l (dword_0_1C1C).l,a6
CODE:00000052 moveq #25,d0 ; '%'
CODE:00000054 jsr -$228(a6)
CODE:00000058 movea.l d0,a5
CODE:0000005A tst.l d0
CODE:0000005C beq.w loc_0_17C
CODE:00000060 move.l a5,(dword_0_1C18).l
CODE:00000066 lea $80+var_18(drp),a1
CODE:0000006A moveq #0,d0
CODE:0000006C moveq #18,d1
CODE:0000006E jsr (sub_0_1BA0).l
CODE:00000074 lea aFileMASrelShow,a0 ; "FILE/M/A,SREL=SHOWRELOC/"
CODE:00000078 move.l a0,d1
CODE:0000007A lea $80+var_18(drp),a0
CODE:0000007E move.l a0,d2
CODE:00000080 movea.l (dword_0_1C18).l,a6
CODE:00000086 moveq #0,d3
CODE:00000088 jsr -$31E(a6)
CODE:0000008C tst.l d0
CODE:0000008E beq.w loc_0_15A
CODE:00000092 moveq #0,d7
CODE:00000094 bra.w loc_0_150
```



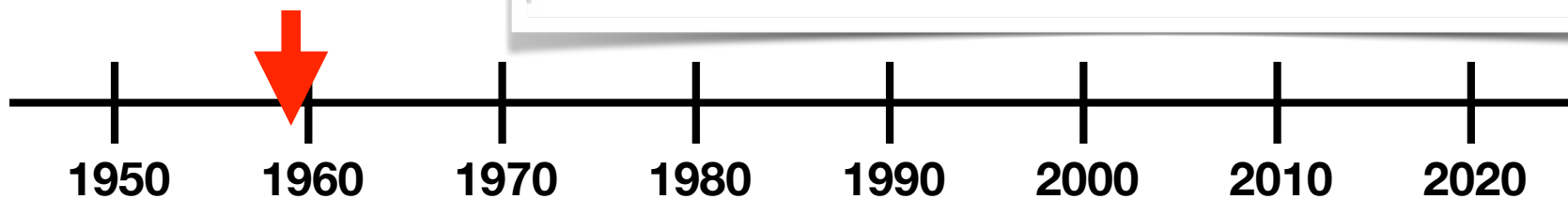
FORTRAN (FORmula TRANslation)

```
PROGRAM FIBONA
C
C PROGRAM TO CALCULATE THE SUM OF THE FIRST TEN FIBONACCI NUMBERS -
C THE RESULT IS HELD IN VARIABLE ANSWER
C
C INTEGER N1, N2, NEW, SUM, ANSWER
C
C N1 = 1
C N2 = 1
C SUM = N1 + N2
C
C DO 10 I=3,10
C     NEW = N1 + N2
C     N1 = N2
C     N2 = NEW
C     SUM = SUM + NEW
10 CONTINUE
C
C ANSWER = SUM
END
```

COBOL

```
method-id button1_Click final private.
local-storage section.
01 ex type Exception.
procedure division using by value sender as object e as type System.EventArgs
    set lblError::Text to " ".
    open input bookfile.
    evaluate ls-file-status
        when "00" continue
        when "35"
            set ex to new Exception("File Not Found")
            raise ex
        when other
            raise new Exception("File Error on bookfile")
    end-evaluate.

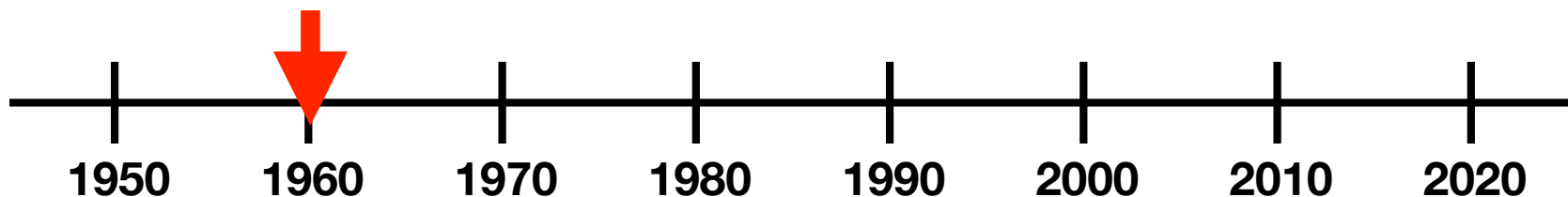
end method.
```



LISP (LISt Processing)

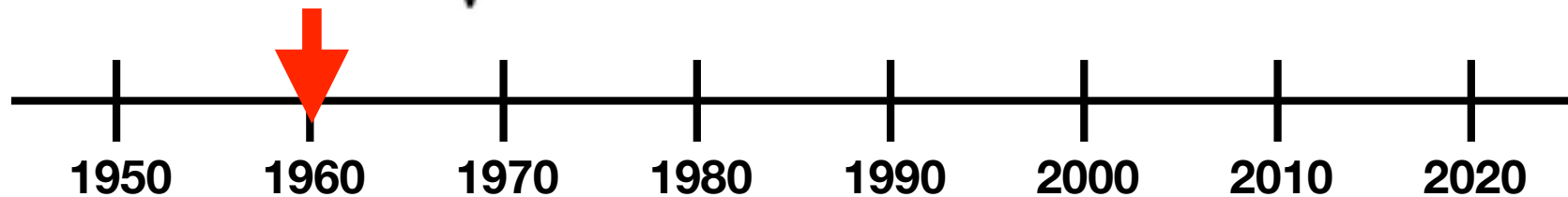
```
MACRO
(((NOTED
  (LAMBDA (FORM)
    ((LABEL PURGE
      (LAMBDA (X)
        (COND
          ((ATOM X) X)
          ((ATOM (CAR X)) (CONS (CAR X) (PURGE (CDR X))))
          ((EQ (CAAR X) (QUOTE NOTE)) (PURGE (CDR X)))
          (T (LIST (PURGE (CAR X)) (PURGE (CDR X))))
        ))
      ))
    (CDR FORM)
  ))))
```

Bobrow, LISP Bulletin No 1, 1969

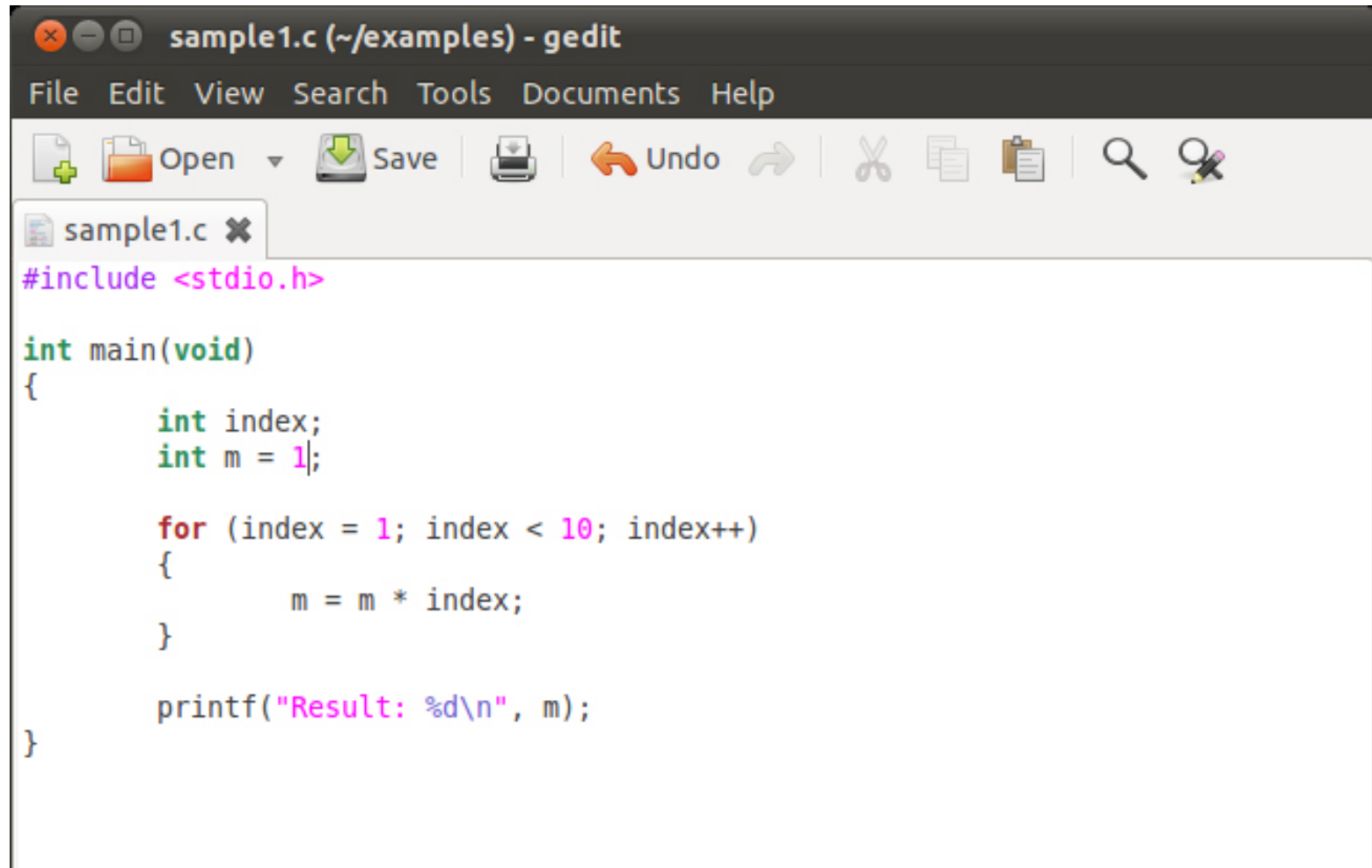


APL (A Programming Language)

```
      ∇DET[□]∇
      ∇ Z←DET A;B;P;I
[ 1 ]      I←□IO
[ 2 ]      Z←1
[ 3 ]      L:P←(|A[;I])∖|A[;I]
[ 4 ]      →(P=I)/LL
[ 5 ]      A[I,P;]←A[P,I;]
[ 6 ]      Z←-Z
[ 7 ]      LL:Z←Z×B←A[I;I]
[ 8 ]      →(0 1 ∨.=Z,1↑ρA)/0
[ 9 ]      A←1 1 ↓A-(A[;I]÷B)∘.×A[I;]
[10 ]      →L
[11 ]      ρEVALUATES A DETERMINANT
      ∇
```



C Language

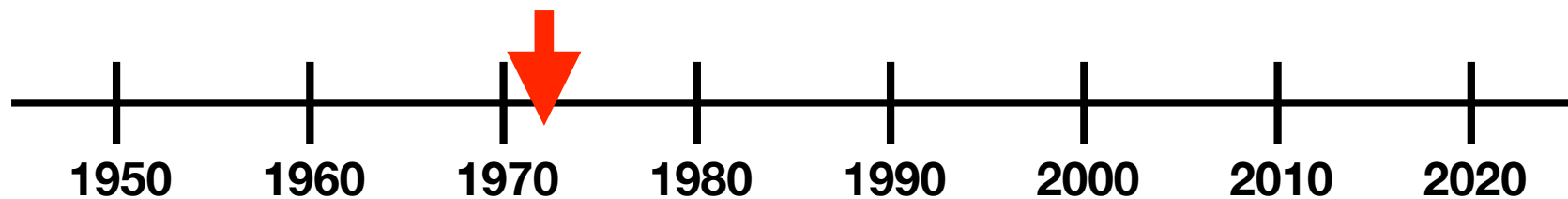


```
sample1.c (~/examples) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
sample1.c x
#include <stdio.h>

int main(void)
{
    int index;
    int m = 1;

    for (index = 1; index < 10; index++)
    {
        m = m * index;
    }

    printf("Result: %d\n", m);
}
```



Pascal

```
program persegi_panjang;

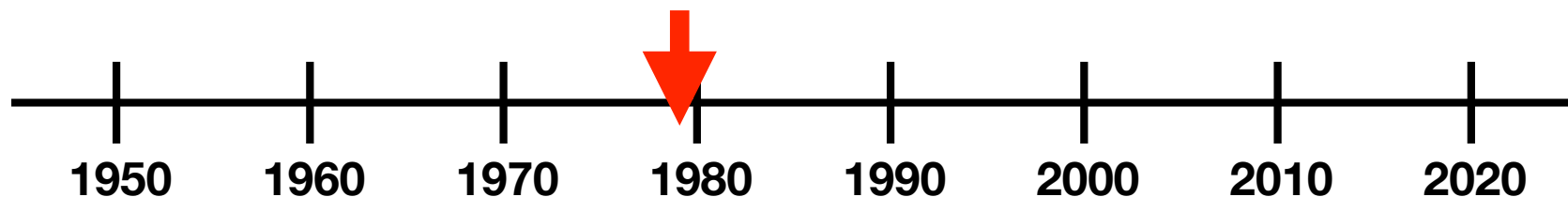
var Ulang : char;
    LS,P,L,K : integer;

begin
    ulang := 'y';
    while (ulang = 'y') do
        begin
            write('panjang = ');readln(p);
            write('lebar   = ');readln(l);
            ls := p * l;
            k  := 2 * (p + l);
            writeln('Luas      =',ls);
            writeln('Keliling =',k);
            write('mengulang (y/n) ? ');readln(ulang);
        end;
    end.
    {algoritma menghitung luas}
```

ADA

```
procedure Test is
    A: array (1 .. 3) of Natural;
    S: Natural;
    K: Natural;
begin
    A := (1,2,3);
    S := 0;
    K := A'First;

    loop
        if K <= A'Last then
            S := S + A(K);
            K := K + 1;
        else
            exit;
        end if;
    end loop;
end Test;
```



C++

Python

```
using namespace std;

class CBaseClass
{
public:

    virtual void SayHello()
    {
        cout << "Hello from CBaseClass\n\n" ;
    }

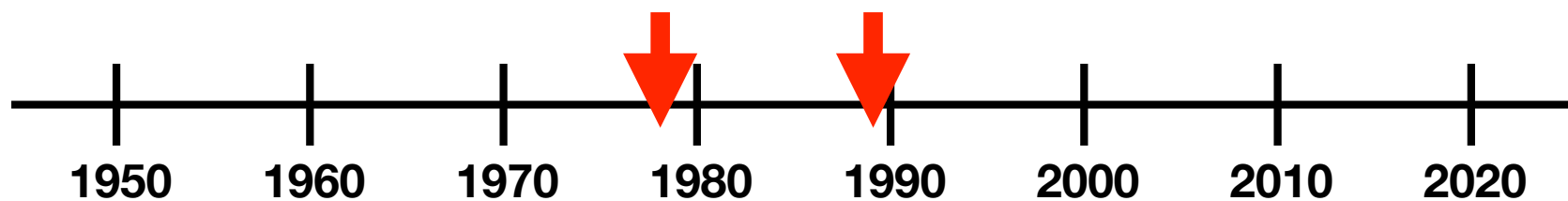
    void SayHi()
    {
        cout << "Hi from CBaseClass\n\n";
    }
};
```

```
def removeVowels(word):
    |
    vowels = 'aeiouAEIOU'
    for j in range(len(word)):

        if str(word[j]) in vowels:
            wordCopy = word.replace(word[j], '_')
            word = wordCopy

    return (wordCopy)

hidden = removeVowels("Lose all of the vowels")
print (hidden)
```



Javascript

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
    <TITLE>Logical operators</TITLE>

    <SCRIPT LANGUAGE = "Javascript">

      var cookies_enabled = true;
      var javascript_enabled = false;

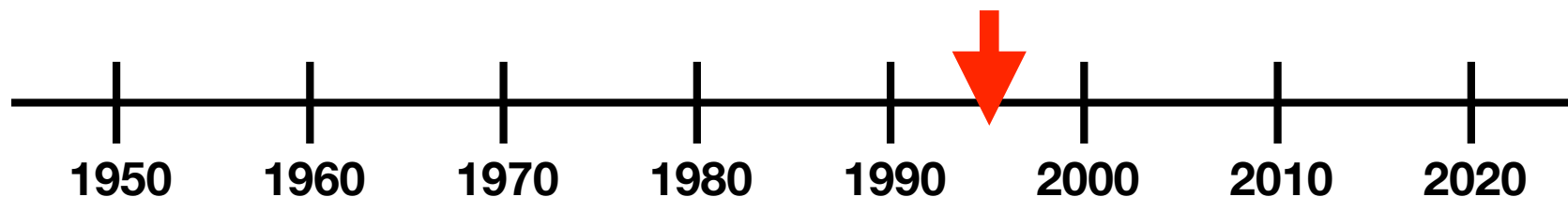
      if (cookies_enabled && javascript_enabled) {
        document.write("ALL OK - continue");
      }
      else {
        document.write("ALL NOT OK - issue warning.");
      }

    </SCRIPT>

  </HEAD>

  <BODY>

    </BODY>
</HTML>
```



A decorative vertical strip on the left side of the slide, featuring a geometric pattern of overlapping triangles in shades of yellow, white, grey, and magenta.

Java

Syntax and Style

Syntax and Style on the Web

GREENSTEIN - CONVENTIONS

[HOME](#)[APCS](#)[GEOMETRY](#)[JAVA](#)[ABOUT](#)[CONTACT](#)[MVHS](#)

Coding Conventions

Coding conventions are important in this course because they make your code more readable, quicker to develop, and easier to grade. The bigger the project, the more benefit software conventions will have on development time.

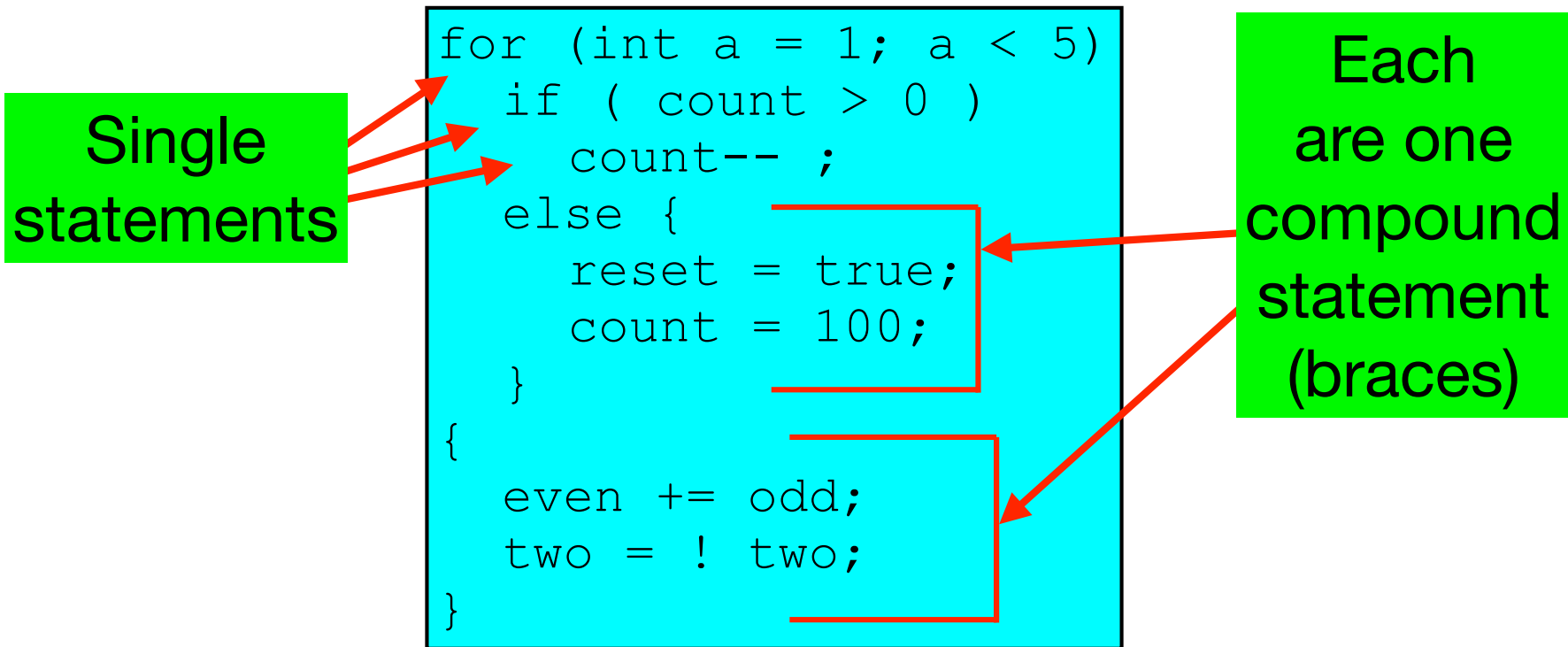
In the real world, good software conventions are valued because the code is easier to maintain. Sun Microsystems, who created the Java language, found:

- 40% to 80% of the lifetime cost of a piece of software goes to maintenance.
- Most software is maintained by people other than the original author.
- Coding conventions improve readability, allowing engineers to understand the code quickly and thoroughly.

Good coding conventions also allow you to revisit your own software months, maybe years later and quickly pick up where you left off.

[NAMING CONVENTIONS](#)[COMMENTING CONVENTIONS](#)[FORMATTING CONVENTIONS](#)[IMPORTING CONVENTIONS](#)

Brace “{ }” Structure



- A pair of braces denote one “compound statement” (even if the braces contain a single statement).
- Statements inside braces are indented.
- Java braces are an example of “block structured” languages

Reserved Words

- In Java, a number of words are reserved for a special purpose.
- Reserved words use only lowercase letters.
- Reserved words include:
 - primitive data types: **int**, **double**, **char**, **boolean**, etc.
 - storage modifiers: **public**, **private**, **static**, **final**, etc.
 - control statements: **if**, **else**, **switch**, **while**, **for**, etc.
 - built-in constants: **true**, **false**, **null**
- There are about 50 reserved words total.

Programmer-Defined Names

- The programmer gives names to his or her classes, methods, fields, and variables.
- In addition to reserved words, Java uses standard names for library packages and classes (APIs):
 - **java.lang.String**
 - **java.io.File**
 - **java.awt.Graphics**
 - **java.util.Scanner**
 - **javax.swing.JFrame**
- Careful! Check to be sure your class names do not conflict with Java's API class names.

Programmer-Defined Names (cont.)

- Syntax: A name can include:
 - upper- and lowercase letters (e.g. camelCase)
 - digits (e.g. july4th)
 - underscore characters (e.g. CARD_CNT)
- Syntax: A name cannot begin with a digit.
4score, 365days
- Style: Names should be descriptive to improve readability.
 - YES: dealtCards; NO: dc
 - Exception: names with limited roles, like names used in loops.
for (int a = 1; a < 5; a++)

Programmer-Defined Names (cont.)

- **Class names**

- The first letter is always uppercase.
- The name should use camel-case.
- The name describes the class and are “noun-like”.

e.g. MyClass, CardDeck, Yahtzee

- **Method names**

- The first letter is always lowercase.
- The name should use camel-case.
- The name should be “verb-like” describing the action.

e.g. setBackground, getText, moveForward

Programmer-Defined Names (cont.)

- **Field names**

- The first letter is always lowercase.
- The name should use camel-case.
- The name should be “noun-like” describing the object.

e.g. `count`, `windowWidth`, `htmlCanvas`

- **Constant names**

- Use ALL_UPPER_CASE for constants separating words with underline characters.
- Java constant fields are declared **`private final`**.
- Make constants out of “magic” numbers. These are numbers that have a significant meaning in your code. For example, the number of cards in a hand.

e.g. `TAX_RATE`, `CARD_HAND`

Comments

- **In-line comment “//”**

- The Java compiler ignores anything on a line to the right of a double-slash “//”.

```
c += d; // comment is end of line
// comment is the whole line
```

- **Multiple-line comment delimiters “/*” and “*/”**

- The Java compiler ignores anything starting with “/*” and ending with “*/”.

```
/*
   This comment can
   last several lines.
*/
```



Great for commenting out large segments of code for debugging!!!

Comments (cont.)

- **JavaDoc comments for documenting** surrounded by **“/**”** and **“*/”**
 - The **javadoc** program automatically reads the comments from your source code and generates API-style HTML.
 - javadoc **annotations**, denoted by a **“@”**, provide key information.

```
/** This method calculates the determinant
    of a matrix.
    Precondition: matrix A must be square
    @param A      matrix A
    @return       the determinant of A
    */
public int determinant(Matrix A) { ...
```

Comments (cont.)

- **Always provide generous amounts of comments**

Comments (cont.)

- Always provide generous amounts of comments
 - Comment important fields and local variables

```
JPanel first;    // First panel shown in the game  
int count;      // Number of cards remaining  
  
double a, b, c;
```

Helper variables
do not need
comments

Important fields in
design need
comments

Comments (cont.)

- **Always provide generous amounts of comments**

- Comment important fields and local variables
- **Comment each method**

A concise description

```
/** Determines number of zombies near robot
    Precondition: robot is on field
    @param bot      the robot
    @param loc      robot's current location
    @return         number of nearby zombies
 */
public int zombieCounter(Robot bot, Location loc) {
```

Describe what is returned

List preconditions and postconditions

List all of the input parameters

Comments (cont.)

- **Always provide generous amounts of comments**
 - Comment important fields and local variables
 - Comment each method
 - **Put a header on each class**

A concise description

```
/** Robot class
 *  A friendly robot that walks around a Field
 *  fighting zombies in its path.
 *  @author Mr Greenstein
 *  @since September 29, 2017
 */
public class Robot implements Comparable { ...
```

Author (you)
and date created

Comments (cont.)

- **Always provide generous amounts of comments**
 - Comment important fields and local variables
 - Comment each method
 - Put a header on each class
- **Missing comments? Serious consequences!**

**LOSE 10% OF
YOUR PROJECT GRADE**

Other Style Issues

- Keep your coding under 80 characters per line.

NO

```
if (item.getItemCost() > money) {  
    System.out.printf("\nNO SALE: Not enough money to buy the item\tChange: $%6.2f\n",  
        item.getItemCost(), money, item.getItemCost() - money);  
    return -1;  
}
```

YES

```
if (item.getItemCost() > money) {  
    System.out.printf("\nNO SALE: Not enough money to buy the " +  
        "item\tChange: $%6.2f\n\n", money, item.getItemCost() - money);  
    return -1;  
}
```

Other Style Issues

- Keep your coding to < 80 characters per line.
- **Names are descriptive, but short (< 15 chars).**

NO `int theLengthOfTheField;`

YES `int fieldLength;`

Other Style Issues

- Keep your coding to < 80 characters per line.
- Names are descriptive, but short (< 15 chars).
- **Names of methods that return a boolean value start with “is” or “has”. For example:**

```
public boolean isOverdrawn ()  
public boolean hasCreditLeft ()
```

Syntax Errors

- The compiler catches syntax errors and generates error messages.
- Text in **comments** and **literal strings** within double quotes are excluded from syntax checking.
- **Braces, brackets, and parentheses** (`{}`, `[]`, `()`) can be on different lines
- **Double quotes** (`""`) must be on the same line

Common Syntax Errors

Spelling

```
public static int abs (int x)
{
    fi (x < 0);
    {
        x = -x
    }
    return x;
```

Extraneous
semicolon

Missing
closing
brace

```
public static int sqrt (int x)
...
}
```

Missing
semicolon

Syntax Errors (cont.)

- Pay attention to and check for:
 - Matching braces { }, parentheses (), and brackets [].
 - Missing or extraneous semicolons.
 - Symbols used correctly for operators +, -, =, <, <=, ==, ++, &&, etc.
 - Spelling is correct for reserved words, library names and programmer-defined names, with special attention to upper/lower case.

Which is a better style?

```
public void act()  
{if(steps< sideLength&&  
canMove()) {move();  
steps++;}else{  
turn();turn(); steps=0;}}
```

```
public void act() {  
    if (steps < sideLength &&  
        canMove()) {  
        move();  
        steps++;  
    }  
    else {  
        turn();  
        turn();  
        steps = 0;  
    }  
}
```

Both compile

- Put spaces between lines.
- Put spaces between words and operators.
- Indent nested code.

A decorative vertical strip on the left side of the slide, featuring a complex geometric pattern of overlapping triangles and polygons in shades of yellow, white, grey, and magenta.

Questions?