# Programming in VISUAL BASIC

# Lecture

Dr Jeff Drobman

website → drjeffsoftware.com/classroom.html

email → jeffrey.drobman@csun.edu

# Index

# Section



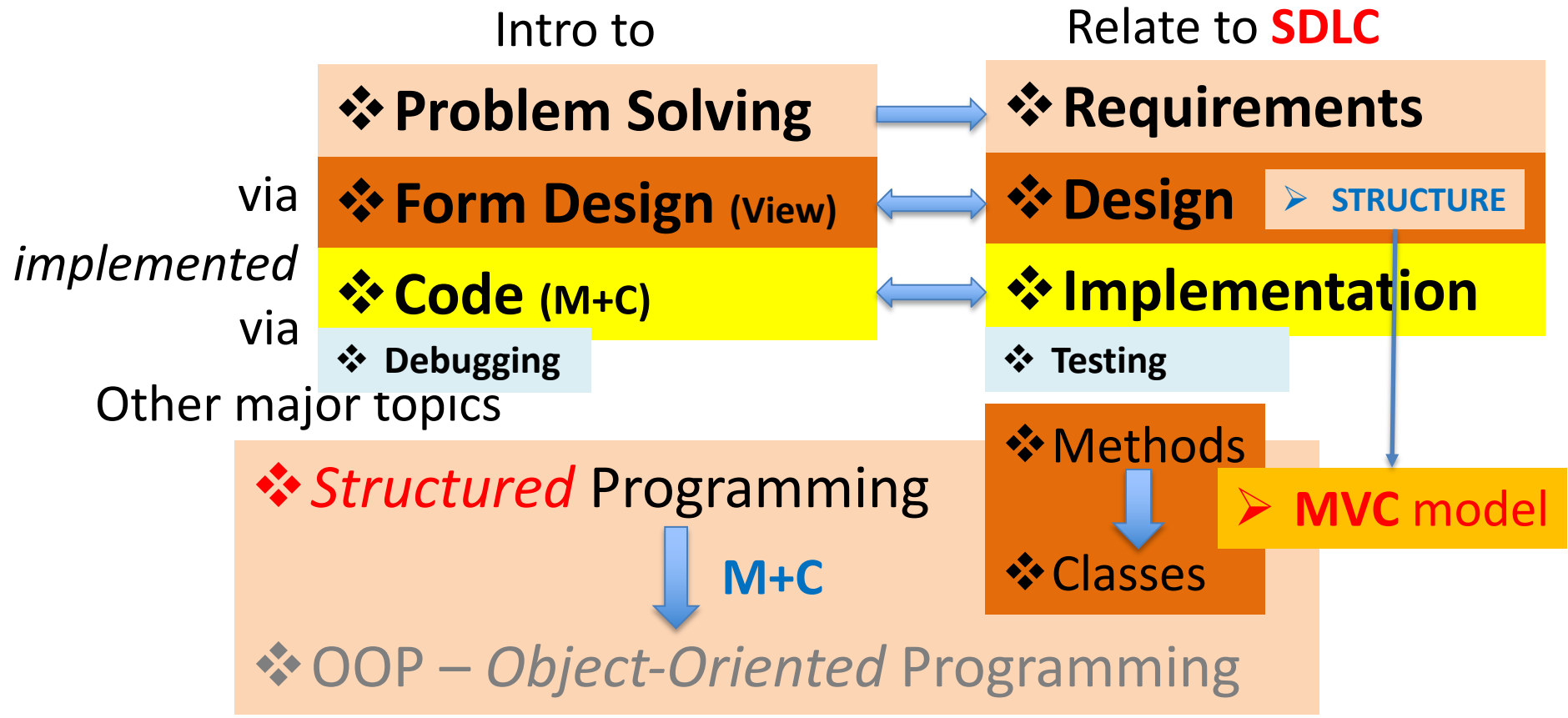
# Course Overview

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP105BAS

# Course Overview





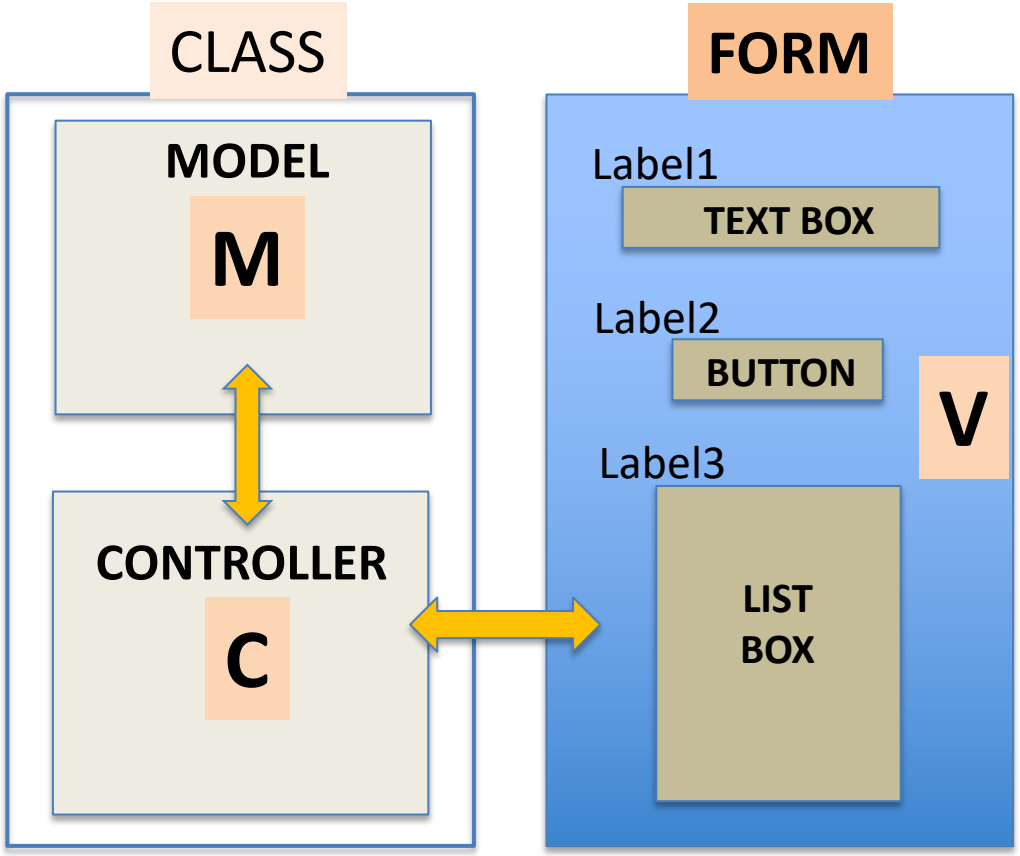

Programming Languages

- ❖ Using **VB.NET**
- ❖ Compare to **Java**, C, C++

# Classes <-> Forms

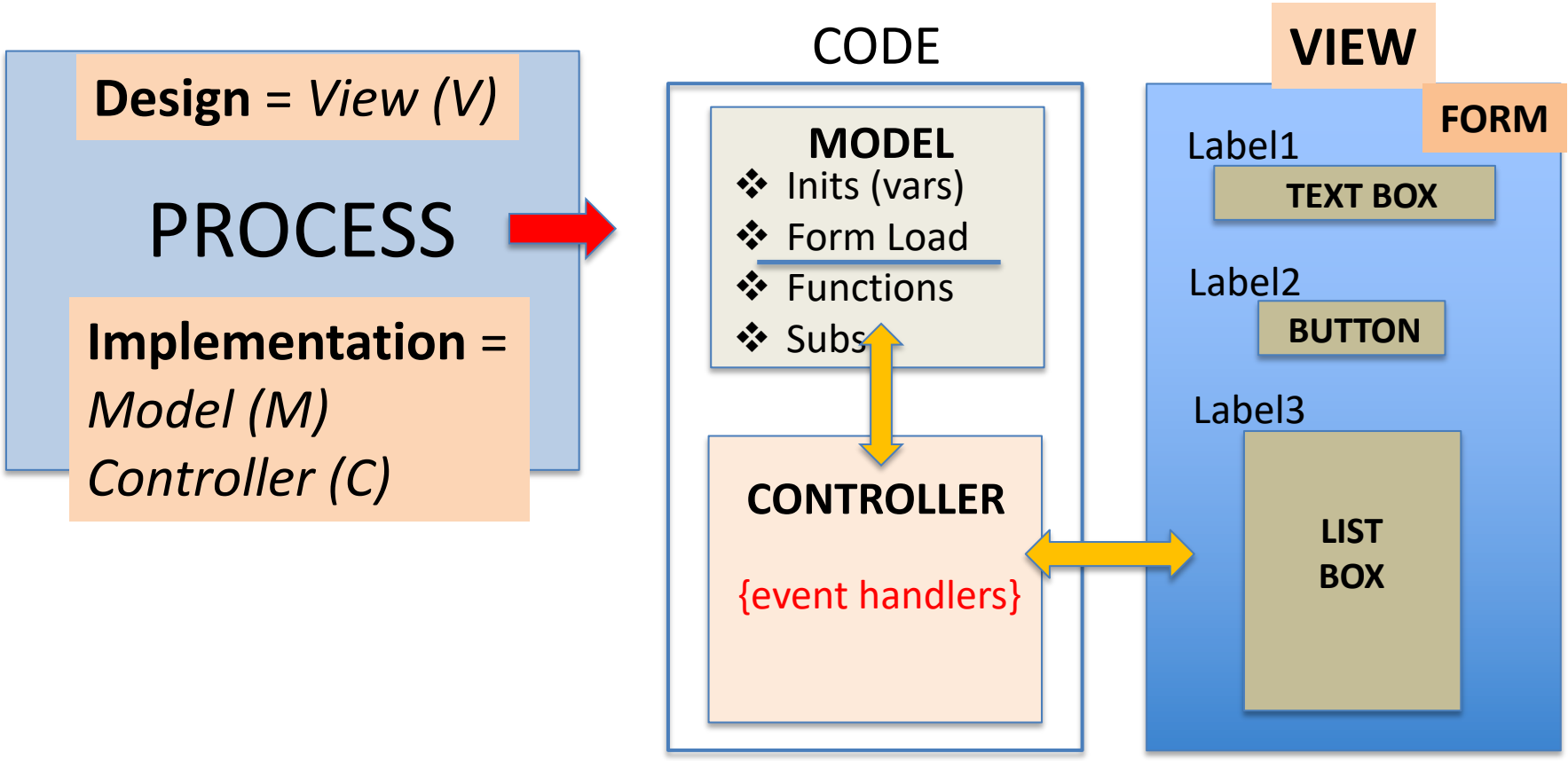


- **MVC** model



# MVC Design Model





# MVC :: Classes



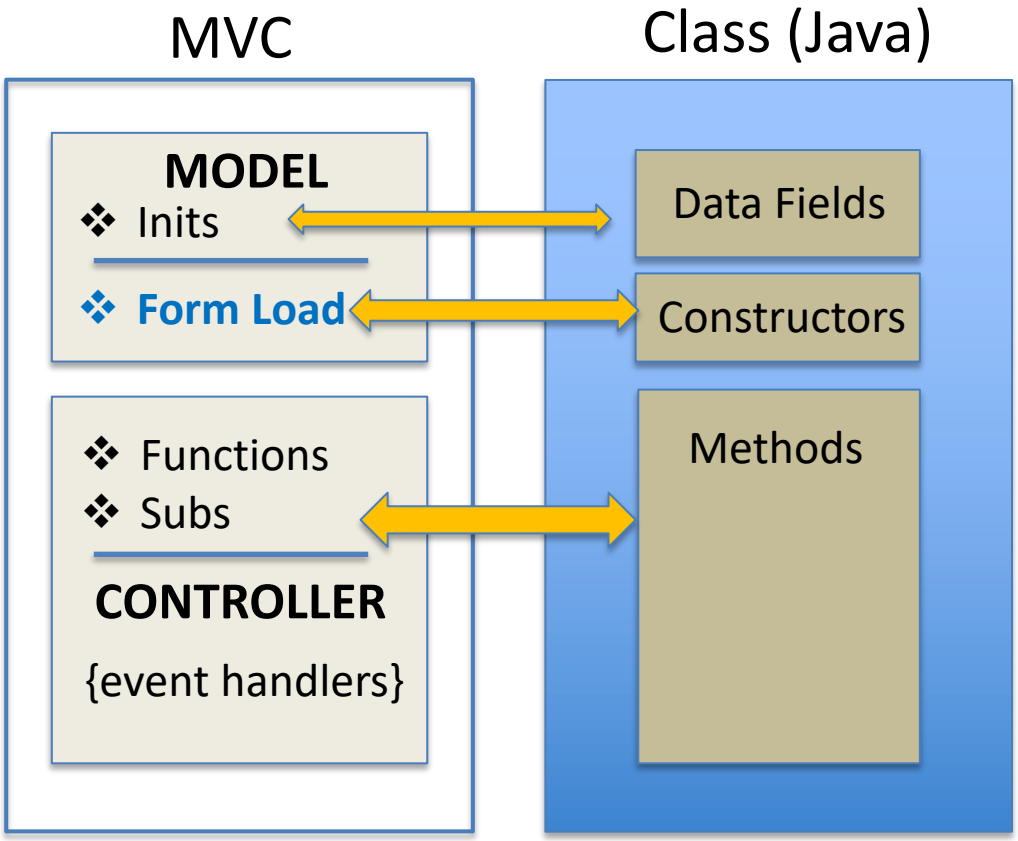
Section headers

'INITS

'FORM LOAD

'MODEL

'CONTROLLER



# Lecture



## Subroutines

# Subroutines-*VB*



VB

```
Private Sub limra(ByRef dd As Int16, ByVal x As Byte)
    < statements >
End Sub
```

Return optional

```
Private Sub abs(ByRef dd As Int16)
    < statements >
End Sub
```

```
Private Function pad0(ByVal dd)
    < statements >
    Return (dd)
End Function
```

Return required

```
Private Sub caller()
    Call abs(x)
    Call pad0(n)
    Call limra(z,m)
End Sub
```

- Call
- Parameter passing
  - By *Value*
  - By *Reference*
- Return
  - Sub -> *void*
  - Function -> *value*

# Section



## Control Structures

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP105BAS

# Control Structures:

## *Conditional & Loops*

VB



- IF-THEN-ELSE

```
If I <= N Then
   <statements>
ElseIf x <> y
   <statements>
Else
   <statements>
EndIf
```

- Case (Switch)
  - Integer values
  - Discrete objects

```
Select N
   Case is 1: <statements>
   Case is 2: <statements>
   Case is 3: <statements>
   Case Else: <statements>
End Select
```

- Loops
  - FOR (count)
  - (DO) WHILE (switch ON)

```
For I = 1 to N
   <statements>
Next
```

```
Do While  I <= N
   <statements>
   I += 1
Continue
```

# Conditional: *IF-THEN-ELSE*



```
If I <= N Then
   <statements>
ElseIf x <> y
   <statements>
Else
   <statements>
EndIf
```

VB


```
if (I <= N)  {
   <statements>}
elseif (x <> y)  {
   <statements>}
else {
   <statements>
}
```

C/C++

```
if (I <= N)
   <statement>
elseif (x <> y)
   <statement>
else {
   <statements>
}
```

```
if Statements

if (condition) {
  statements;
}

if (condition) {
  statements;
}
else {
  statements;
}

if(condition1) {
  statements;
}
else if (condition2) {
  statements;
}
else {
  statements;
}
```

Java

# Conditional: *CASE/SWITCH*



```
Select N
   Case is 1: <statements>
   Case is 2: <statements>
   Case is 3: <statements>
   Case else: <statements>
End Select
```

VB

❖ *NO fall-through*

```
switch (N)  {
   Case 1: <statements>
           break;
   Case 2: <statements>
           break;
   Case 3: <statements>
}
```

C/C++

```
switch Statements

switch (intExpression) {
  case value1:
    statements;
    break;
  ...
  case valuen:
    statements;
    break;
  default:
    statements;
}
```

❖ *fall-through* if no "break"

Java

# Code Blocks – *Loops*



```
For I = 1 to N
   <statements>
Next
```

```
for (I = 1; I<N; I++) {
   <statements>
}
```

```
Do While (I <= N)
   <statements>
   I += 1
Continue
```

```
while (I <= N)  {
   <statements>
   I ++
}
```

```
do  {
   <statements>
   I ++
} while (I <= N);
```

VB

C

```
while (1) { }
```

embedded apps

```
loop Statements

while (condition) {
  statements;
}

do {
  statements;
} while (condition);

for (init; condition;
  adjustment) {
  statements;
}
```

Java

# FOR Loops



- test FIRST

```
For i = 1 to N
   <statements>
Next
```

- executes N times
- i = 1 to N
- i *post* increments

```
For i = 0 to N-1 By 2
   <statements>
Next
```

- executes N times
- i = 0 to N-1
- i *post* increments BY 2

```
For i = 0 to N-1
   <statements>
if ($FLAG) Exit For
Next
```

- Exit → EXIT early

# Type-Classes



## Strings

# Special Char/String Literals



may declare variables for these:

space: Dim sp As char = " "
2x sp: Dim sp2 As String = sp & sp; 'etc for 2..n

```
Dim xc As Char = "a"
CInt(xc)
Microsoft.VisualBasic.AscW(xc)
```

# Convert String ⇔ Numeric



- convert <string> ⇔ <Type>

<Type>.parse<Type>(<string>) Java

**CInt**(<String type>)

VB

**Cstr**(<numeric exp>)

```
xs = CStr(tries + PART)
```

- check out all the converts "C…( )"

# String Handling





## ❖ STRING Data Type (VB)

Arrays vs. Substrings

```
'**STRING variables
Dim wstr, xstr, ystr, zstr, xxstr, alertst, srchstr, matchstr As String

'**STRING expressions, operators, functions
xstr &= ystr & "123" 'concatenate
zstr = substring(xstr, 1, 4)  'substrings (index, length)
wstr = substring(xstr, 5)  'substrings (index)

'**Strings are an option for arrays:  string indices ⇔ array indices
'string of 10 characters vs. array of 10 elements:  example for Y/N votes
Dim xstr As String, xarr(9) As String
xstr = "YNNYYNYYNN"
xarr(0) = "Y" : xarr(1) = "N" : xarr(2) = "N" : xarr(3) = "Y" : xarr(4) = "Y"
Xarr(5) = "N" : xarr(6) = "Y" : xarr(7) = "Y" : xarr(8) = "N" : xarr(9) = "N"
```

```
'load array of 10 from string
For I = 0 To 9
    Xarr(I) = substring(xstr,I,1)
Next
```

# String Methods



Java

**String Class**

```
String s = "Welcome";
String s = new String(char[]);
int length = s.length();
char ch = s.charAt(index);
int d = s.compareTo(s1);
boolean b = s.equals(s1);
boolean b = s.startsWith(s1);
boolean b = s.endsWith(s1);
String s1 = s.trim();
String s1 = s.toUpperCase();
String s1 = s.toLowerCase();
int index = s.indexOf(ch);
int index = s.lastIndexOf(ch);
String s1 = s.substring(ch);
String s1 = s.substring(i,j);
char[] chs = s.toCharArray();
Strings1 = s.replaceAll(regex,repl);
String[] tokens = s.split(regex);
```

can use ”s1 = s2” in **VB**

ch|s; returns (0-n | -1)

ch::= <beginIndex>

I,j::= <beginIndex, endIndex>

regex

length in **VB**

# String Compare




Java

**TABLE 4.8** Comparison Methods for `String` Objects

| *Method* | *Description* |
|---|---|
| `equals(s1)` | Returns true if this string is equal to string `s1`. |
| `equalsIgnoreCase(s1)` | Returns true if this string is equal to string `s1`; it is case insensitive. |
| `compareTo(s1)` | Returns an integer greater than 0, equal to 0, or less than 0 to indicate than, equal to, or less than `s1`. |
| `compareToIgnoreCase(s1)` | Same as `compareTo` except that the comparison is case insensitive. |
| `startsWith(prefix)` | Returns true if this string starts with the specified prefix. |
| `endsWith(suffix)` | Returns true if this string ends with the specified suffix. |
| `contains(s1)` | Returns true if `s1` is a substring in this string. |

# Midterm Examples—Strings



Java

```
41  //**strings
42  static void strings() {
43     String s1, s2, s3;
44     char[] chArr = alpha.toCharArray();
45     s1 = alpha.substring(0,3) + number.substring(5);
46     System.out.println("s1= " + s1);
47     if (alpha.indexOf('d') == number.indexOf('0'))
48        s2 = "then " + alpha.trim();
49     else
50        s2 = "else " + alpha.substring(3,4);
51     System.out.println("s2= " + s2);
52     char chx = chArr[Integer.parseInt(number.substring(2,3))];
53     System.out.println("chx= " + chx);
54     s3 = alpha.substring(number.indexOf(number.charAt(2)),4);
55     System.out.println("s3= " + s3);
56     //
57     String instr = "madam I'm Adam";
58     instr = instr.toLowerCase();
59     for(int i = 0; i < instr.length(); i++) {
60       chx = instr.charAt(i);
61       //if (isLetter(chx)) System.out.print(chx);
62       int ix = alpha.indexOf(instr.charAt(i));
63       if (ix <0) instr = instr.substring(0, i) + instr.substring(i+1);
64       }//end for
65     System.out.println("\n" + instr);
66     System.out.println("----");
67  }//end strings
```

```
----jGRASP exec: ja
starting code
starting Strings...
s1= abc56789
s2= else d
chx= c
s3= cd

madamimadam
```

# Section



## Misc VB

# Control Characters



- vbCr
- vbLf
- **vbCrLf <-> \n**
- vbTab <-> \t

# Ending Programs



- **End** ← Be sure to add a 'comment
- System.exit(0)
- Me.Close( ) or just **Close( )**
- Application.restart( ) ← Ends and Restarts!

# Date & Time



| Property or Function | Description |
|---|---|
| TimeString | This property returns the current time from the system clock. |
| DateString | This property returns the current date from the system clock. |
| Now | This property returns an encoded value representing the current date and time. This property is most useful as an argument for other system clock functions. |
| Hour (*time*) | This function returns the hour portion of the specified time (0 through 23). |
| Minute (*time*) | This function returns the minute portion of the specified time (0 through 59). |
| Second (*time*) | This function returns the second portion of the specified time (0 through 59). |
| Day (*date*) | This function returns a whole number representing the day of the month (1 through 31). |
| Month (*date*) | This function returns a whole number representing the month (1 through 12). |
| Year (*date*) | This function returns the year portion of the specified date. |
| Weekday (*date*) | This function returns a whole number representing the day of the week (1 is Sunday, 2 is Monday, and so on). |

- We use ***DatePicker***

# Dialogs



| Control Name | Purpose |
|---|---|
| OpenFileDialog | Get the drive, folder name, and filename for an existing file |
| SaveFileDialog | Get the drive, folder name, and filename for a new file |
| FontDialog | Let the user choose a new font type and style |
| ColorDialog | Let the user select a color from a palette |
| PrintDialog | Let the user set printing options |
| PrintPreviewDialog | Display a print preview dialog box like Microsoft Word does |
| PageSetupDialog | Let the user control page setup options, such as margins, paper size, and layout |

- File
- Font
- Color
- Print

# Ordinals



Technical ordinals

| | |
|---|---|
| 10^(-24) | yacto |
| 10^(-21) | zepto |
| 10^(-18) | atto |
| 10^(-15) | femto |
| 10^(-12) | pico |
| 10^(-9) | nano |
| 10^(-6) | micro |
| 10^(-3) | milli |
| 10^(-2) | centi |
| 10^(-1) | deci |
| 10^(+1) | deka |
| 10^(+2) | hecto |
| 10^(+3)/2^(10) | kilo |
| 10^(+6)/2^(20) | mega |
| 10^(+9)/2^(30) | giga |
| 10^(+12)/2^(40) | tera |
| 10^(+15)/2^(50) | peta |
| 10^(+18)/2^(60) | exa |
| 10^(+21)/2^(70) | zetta |
| 10^(+24)/2^(80) | yotta |

Gazillions

| | |
|---|---|
| 10^(+6) | million |
| 10^(+9) | billion |
| 10^(+12) | trillion |
| 10^(+15) | quadrillion |
| 10^(+18) | quintillion |
| 10^(+21) | sexillion |
| 10^(+24) | septillion |
| 10^(+27) | octillion |
| 10^(+30) | nonillion |
| 10^(+33) | decillion |
| 10^(+36) | undecillion |
| 10^(+39) | duodecillion |
| 10^(+42) | tredecillion |
| 10^(+45) | quattuordecillion |
| 10^(+48) | quindecillion |
| 10^(+51) | sexdecillion |
| 10^(+54) | septendecillion |
| 10^(+57) | octodecillion |
| 10^(+60) | novemdecillion |
| 10^(+63) | vigintillion |
| 10^(+100) | googol |
| 10^(+303) | centillion |
| 10^(10^(+100)) | googolplex |

| Ordinal | Power of 2 | Power of 10 | Actual |
|---|---|---|---|
| 1K | $2^{10}$ | $10^{3}$ | 1024 |
| 1M | $2^{20}$ | $10^{6}$ | 1,048,576 |
| 1G | $2^{30}$ | $10^{9}$ | $1.074 \times 10^{9}$ |
| 1T | $2^{40}$ | $10^{12}$ | $1.0995 \times 10^{12}$ |

| Name | $2^n$ | M/G | Actual |
|---|---|---|---|
| byte | $2^{8}$ | -- | 256 |
| short | $2^{16}$ | 64K | 65,536 |
| word | $2^{32}$ | 4B | $4.3 \times 10^{9}$ |
| long | $2^{64}$ | 16 Q | $1.84 \times 10^{19}$ |
| IPv6 | $2^{128}$ | 340 uD | $3.4 \times 10^{38}$ |

# Section



## Splash Screens

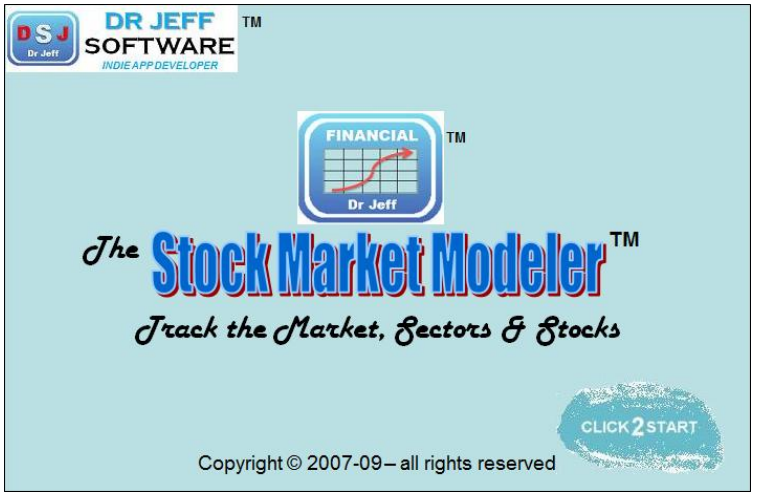
# Splash Screens



# Splash Screens



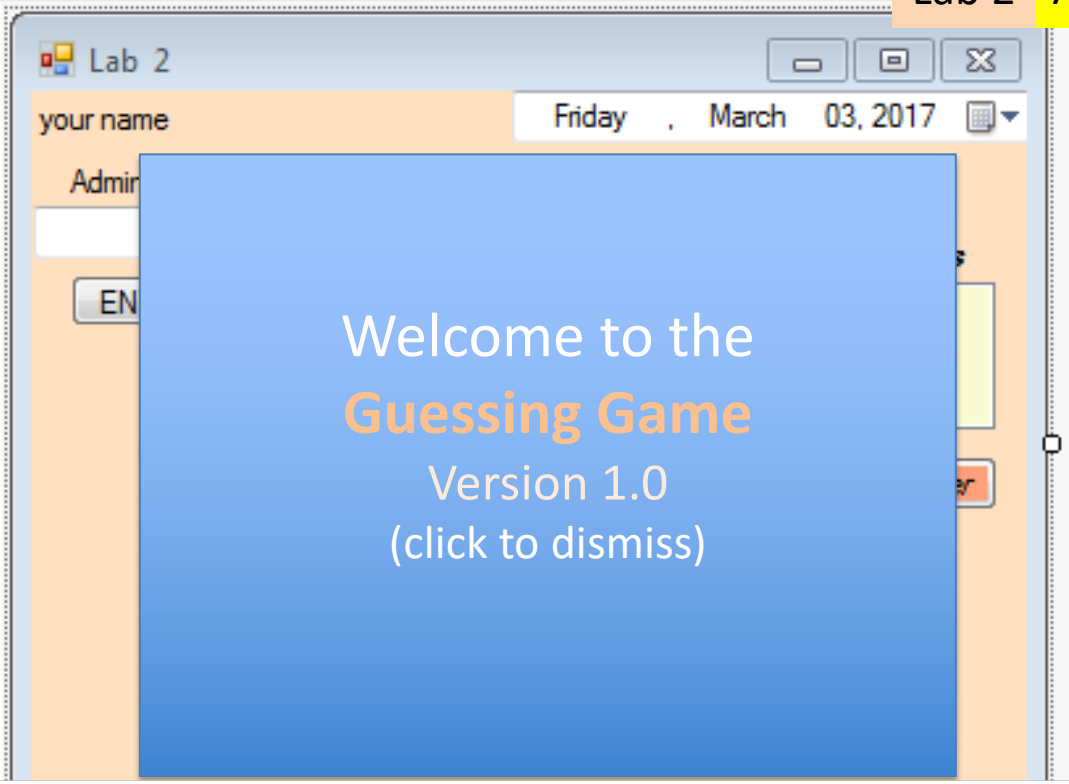

- Picture on Form
  - <name>.visible = True/False
  - Move text items around
  - Not timed:  Click to end
  - Move to Back/Front
- Built-in
  - Project/properties
  - Must create a Form
  - Auto Timed (~2-3 secs)
- Separate Form
  - Set “Launch form” here
  - Not timed:  Click to end
  - Handler Launches Lab 3

# Splash Screen



Lab 2 Add Splash




```
'Controller
0 references
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    splash.Visible = True
    splash.BringToFront()
    Call loadFile()
End Sub
0 references
Private Sub splash_Click(sender As Object, e As EventArgs) Handles splash.Click
    splash.Visible = False
    splash.SendToBack()
End Sub
```

# Chapter 7

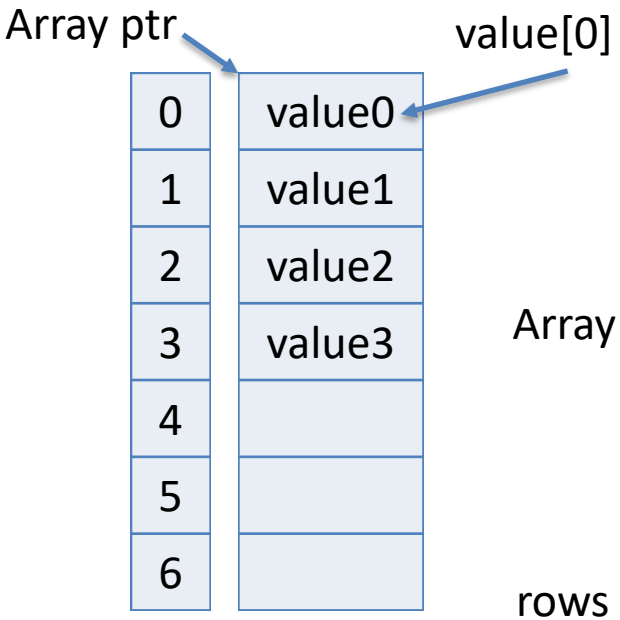


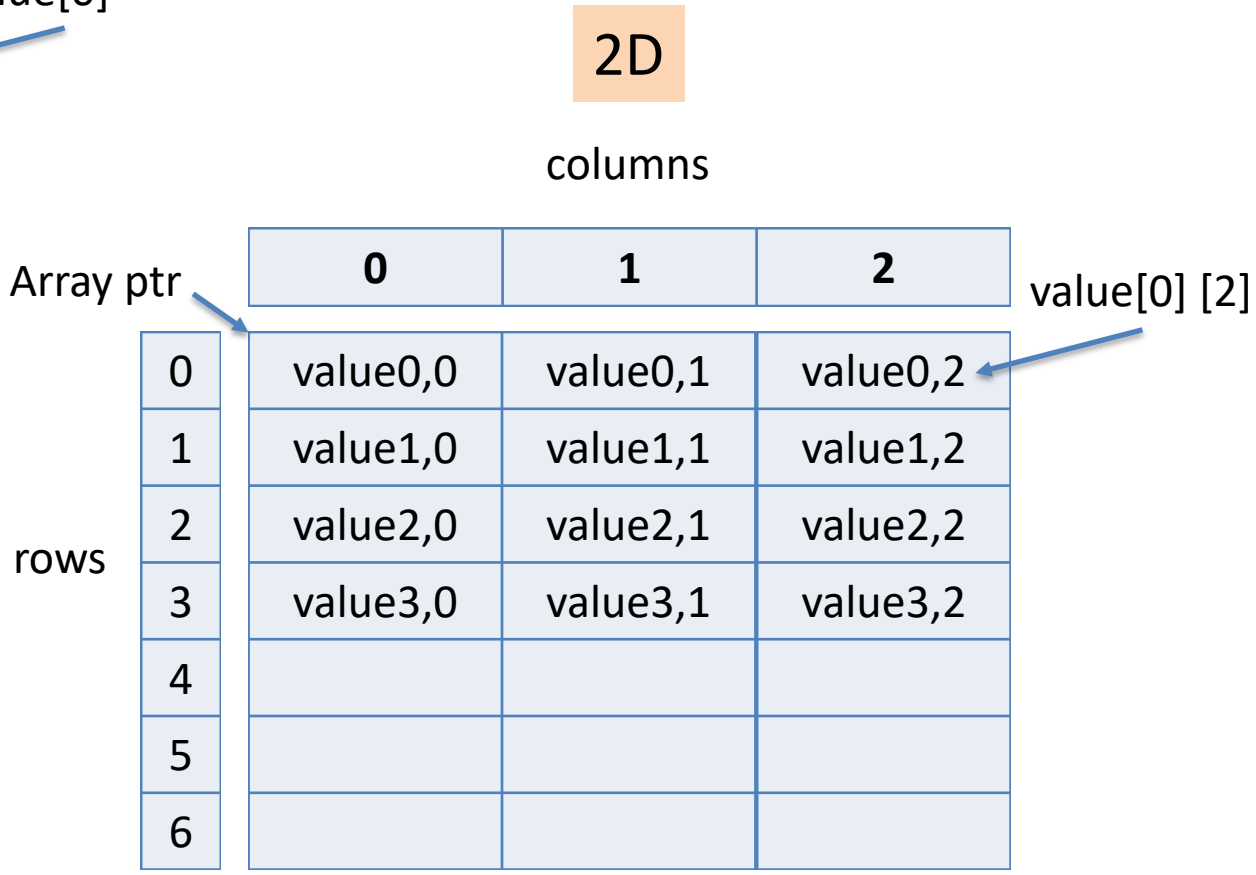
## Arrays

# Arrays



Ch 7

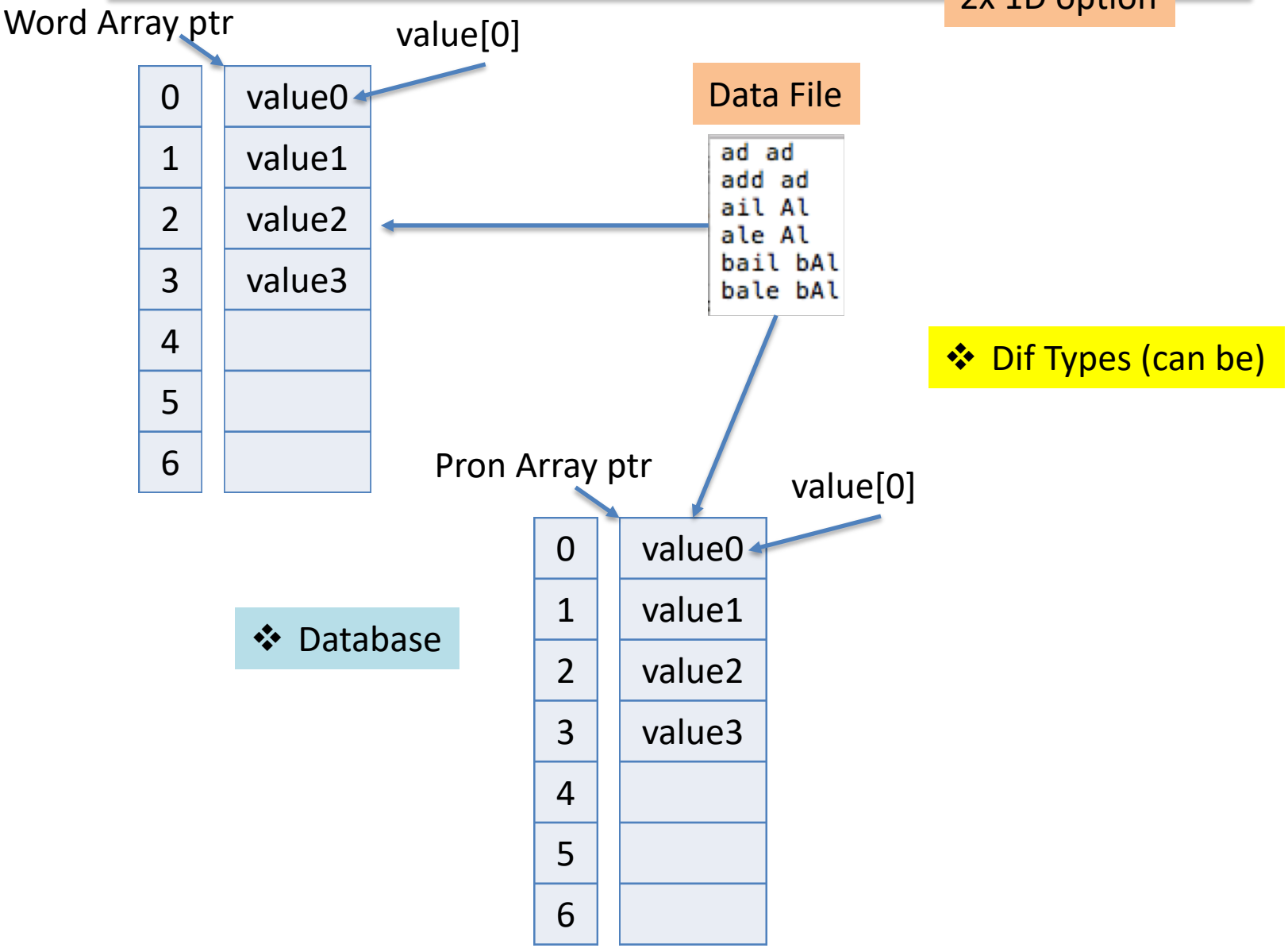


# Arrays



2x 1D option



# Arrays



- Organized collection ("List") of data (all *same* type)
- ***Indexed*** by Integers (Non-negative: 0..N)
- *Associative* arrays in other languages (e.g., PHP) but not Java
- *Single* Dimension ("Linear" array or "vector")
- *Multi* Dimension
  - 2-dimensional: "Matrix" or Table (databases)
  - N-dimensional: abstract

```
Dim myArr(10) As Integer;
```

```
Dim myArr1(10), myArr1(5) As Integer;
```

```
Dim myArr As Integer = {3,21,0,16,0,1};
```

# Section



## Exceptions

# Exception Handling



Java

```
public class cs110Try {
public static void main(String[] args) {
    //*test I/O
    System.out.println("Hello World\n");

    //**test code here
try {
// code here
}
catch(Exception ex) {
//catch code here; "ex" is a parameter
System.out.println(ex);
}
    } //end main & class
}
```

# Try-Catch



Java

```
 9 public class cs110Try {
10 public static void main(String[] args) {
11     //*test I/O
12     System.out.println("Hello World\n");
13
14     //**test code here
15 try {
16 // code here
17 int x = 1/0; //create exception
18 }
19 catch(Exception ex) {
20 //catch code here; "ex" is a parameter
21 System.out.println(ex);
22 }
23 System.out.println("Past Catch block");
24
25     } //end main & class
```

# Exception Handling





- Exception is generic class
- Exception sub-classes
  - ClassNot FoundException
  - IOException

Tables 12.2-3
pp. 456-7

- RuntimeException sub-classes
  - ArithmeticException
  - NullPointerException
  - IndexOutOfBoundsException
  - IllegalArgumentException

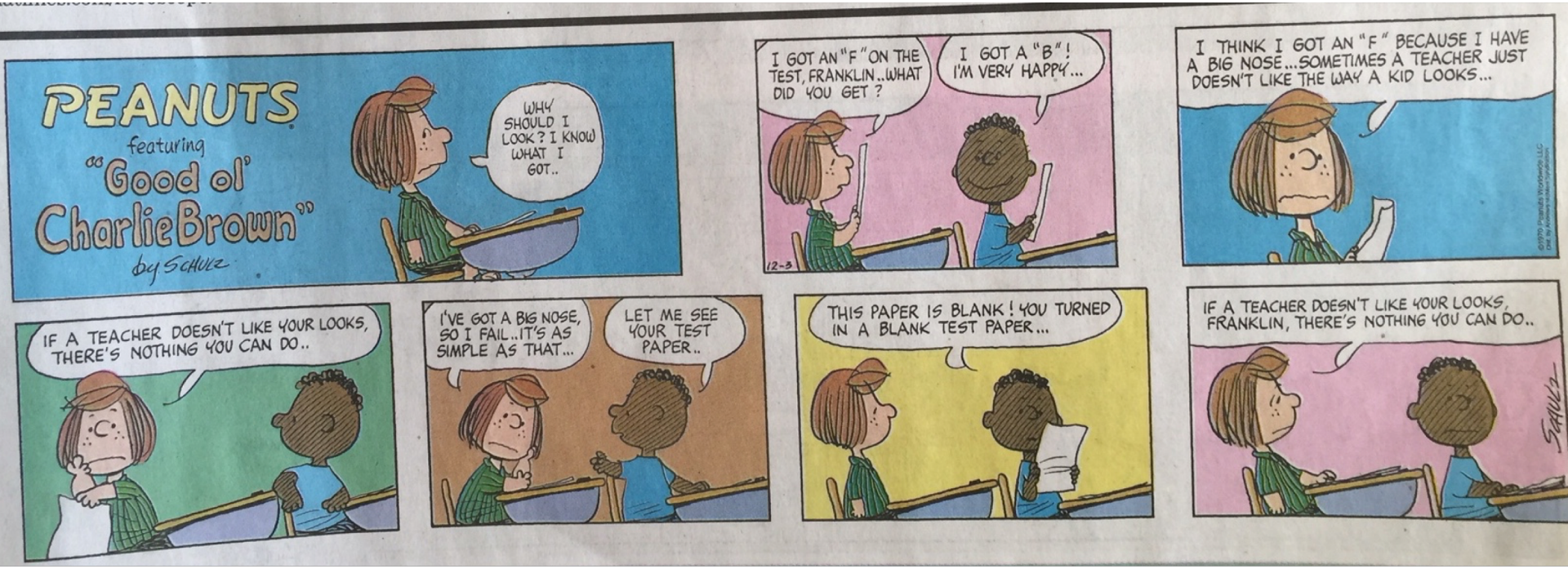
➢ *unchecked* exceptions

➢ *checked* exceptions require declare *throws*
➢ *or use Try-Catch block*

# Section



## Exams



# Exams



- Midterm (8th week)
  - Use Scantron
- Final (Dec 14)
  - Use Scantron
  - Write program segments

| Class | Class Title | Exam Date | Exam Time | Exam Room |
|---|---|---|---|---|
| COMP 105BAS-01 (16220) | COMP PROGRM/BASIC (Laboratory) | 12/14/2018, Friday | 3:00PM - 5:00PM | Jacaranda 1538 |

- Make arrangements for alternative times

# Section



# Cryptography: Blockchains

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP105BAS

# Blockchains

DR JEFF SOFTWARE
INDIE APP DEVELOPER




# Cryptography



- Encryption
  - Scrambles data
  - Makes data *unreadable*
- Hashing
  - Tags data with unique *hash* value
    - Completely *deterministic*
  - Makes data *immutable*
    - Any data corruption is detected

- Protects data:
  - in STORAGE
    - databases (PCI-DSS)
  - in TRANSIT
    - https (uses *TLS*)

- Both use these:
  - Algorithms
  - Keys

# Cryptography



## Encryption

- Used to secure data in storage & *transit*
- Many standards (DES, 3DES, etc.)
- algorithms use sequence of XOR operations
- use public-private key pairs
- replaces each character in situ with a code
- data retains same length
- does not detect tampering

## Hashing

- Used to secure data in storage (only)
- A few standards (MD, SHA)
- algorithms use complex sequence of math operations with key
- use private keys *derived* from random issued words
- does not replace data
- adds a "hash" value to each block of data
- hash value is a fixed 160 bits for SHA
- does detect tampering (*raison d'etre*)

# DES Encryption



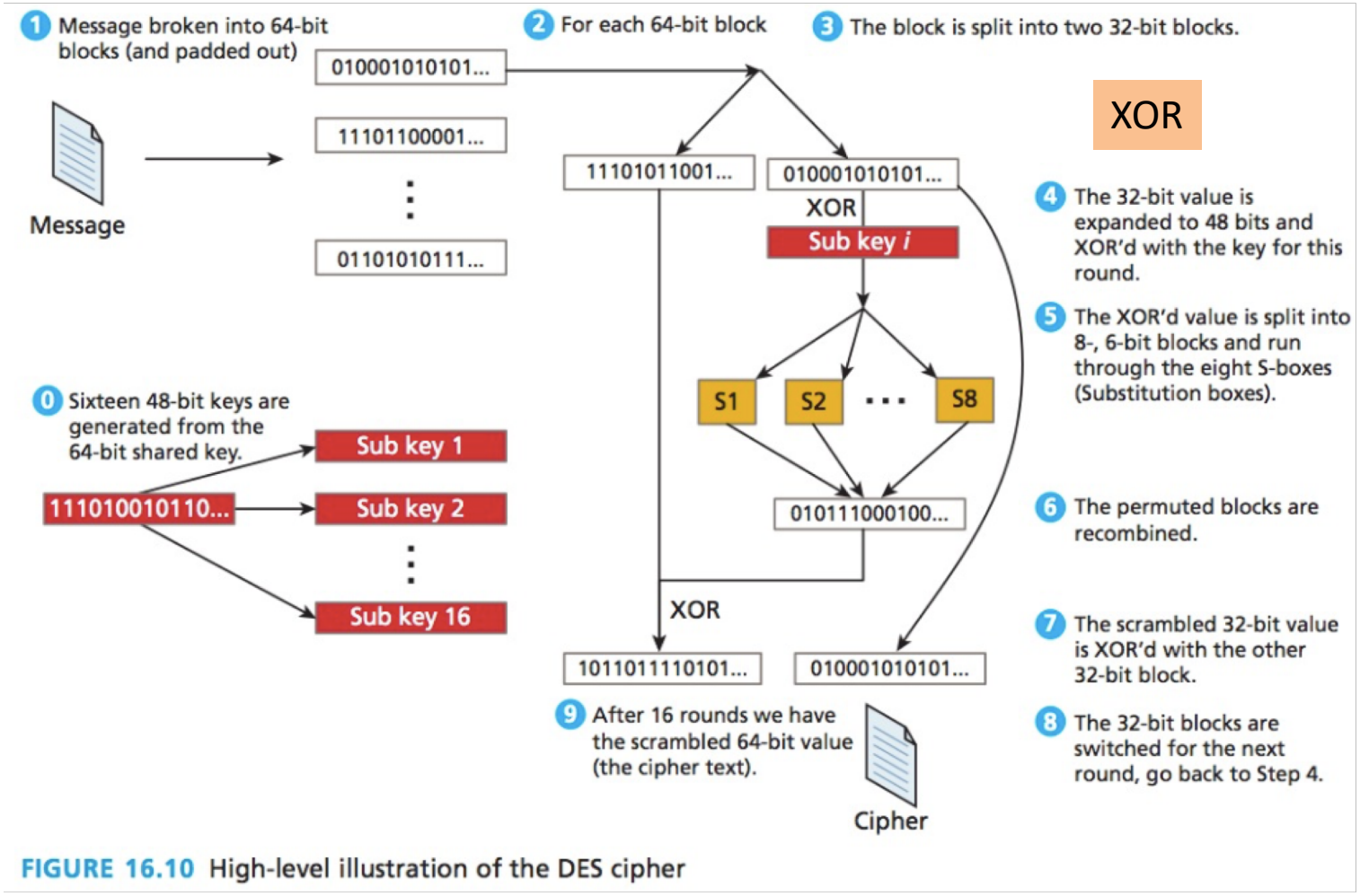



FIGURE 16.10 High-level illustration of the DES cipher

Algorithms + Keys

# SHA Hashing



**Secure Hash Algorithm**


**Concepts**

hash functions · SHA · DSA

**Main standards**

SHA-0 · **SHA-1** · SHA-2 · SHA-3

## SHA-1

From Wikipedia, the free encyclopedia

In cryptography, **SHA-1** (**Secure Hash Algorithm 1**) is a cryptographic hash function which takes an input and produces a 160-bit (20-byte) hash value known as a message digest - typically rendered as a hexadecimal number, 40 digits long. It was designed by the United States National Security Agency, and is a U.S. Federal Information Processing Standard.[3]

Since 2005 SHA-1 has not been considered secure against well-funded opponents,[4] and since 2010 many organizations have recommended its replacement by SHA-2 or SHA-3.[5][6][7] Microsoft, Google, Apple and Mozilla have all announced that their respective browsers will stop accepting SHA-1 SSL certificates by 2017.[8][9][10][11][12][13]

In 2017 CWI Amsterdam and Google announced they had performed a collision attack against SHA-1, publishing two dissimilar PDF files which produced the same SHA-1 hash.[14][15][16]

# SHA Hashing



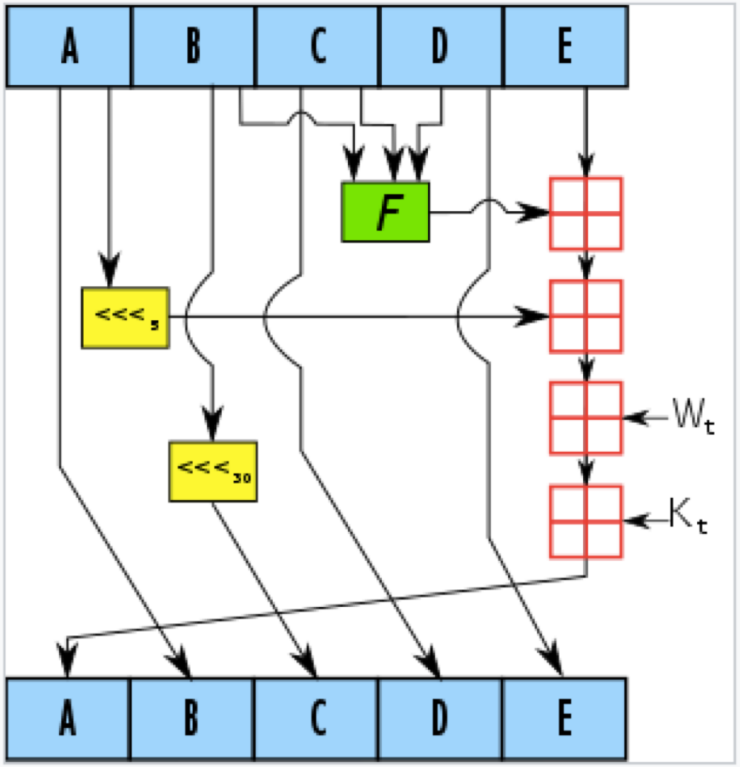
## SHA-1

### General

| | |
|---|---|
| **Designers** | National Security Agency |
| **First published** | 1993 (SHA-0), 1995 (SHA-1) |
| **Series** | (SHA-0), SHA-1, SHA-2, SHA-3 |
| **Certification** | FIPS PUB 180-4, CRYPTREC (Monitored) |

### Cipher detail

| | |
|---|---|
| **Digest sizes** | 160 bits |
| **Block sizes** | 512 bits |
| **Structure** | Merkle–Damgård construction |
| **Rounds** | 80 |




One iteration within the SHA-1 compression function:
A, B, C, D and E are 32-bit words of the state;
*F* is a nonlinear function that varies;
$\lll_n$ denotes a left bit rotation by *n* places;
*n* varies for each operation;
$W_t$ is the expanded message word of round t;
$K_t$ is the round constant of round t;
$\boxplus$ denotes addition modulo $2^{32}$.

# SHA Hashing



**Comparison of SHA functions**

| Algorithm and variant | | Output size (bits) | Internal state size (bits) | Block size (bits) | Max message size (bits) | Rounds | Operations | Security bits (Info) | Capacity against length extension attacks | Performance on Skylake (median cpb)[57] long messages | Performance on Skylake (median cpb)[57] 8 bytes | First Published |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MD5 (as reference) | | 128 | 128 (4 × 32) | 512 | Unlimited[58] | 64 | And, Xor, Rot, Add (mod $2^{32}$), Or | <64 (collisions found) | 0 | 4.99 | 55.00 | 1992 |
| SHA-0 | | 160 | 160 (5 × 32) | 512 | $2^{64} - 1$ | 80 | And, Xor, Rot, Add (mod $2^{32}$), Or | <34 (collisions found) | 0 | ≈ SHA-1 | ≈ SHA-1 | 1993 |
| SHA-1 | | | | | | | | <63 (collisions found[59]) | | 3.47 | 52.00 | 1995 |
| SHA-2 | *SHA-224* | 224 | 256 (8 × 32) | 512 | $2^{64} - 1$ | 64 | And, Xor, Rot, Add (mod $2^{32}$), Or, Shr | 112 | 32 | 7.62 | 84.50 | 2004 |
| | *SHA-256* | 256 | | | | | | 128 | 0 | 7.63 | 85.25 | 2001 |
| | *SHA-384* | 384 | 512 (8 × 64) | 1024 | $2^{128} - 1$ | 80 | And, Xor, Rot, Add (mod $2^{64}$), Or, Shr | 192 | 128 (≤ 384) | 5.12 | 135.75 | |
| | *SHA-512* | 512 | | | | | | 256 | 0 | 5.06 | 135.50 | |
| | *SHA-512/224* | 224 | | | | | | 112 | 288 | ≈ SHA-384 | ≈ SHA-384 | |
| | *SHA-512/256* | 256 | | | | | | 128 | 256 | | | |
| SHA-3 | *SHA3-224* | 224 | 1600 (5 × 5 × 64) | 1152 | Unlimited[60] | 24[61] | And, Xor, Rot, Not | 112 | 448 | 8.12 | 154.25 | 2015 |
| | *SHA3-256* | 256 | | 1088 | | | | 128 | 512 | 8.59 | 155.50 | |
| | *SHA3-384* | 384 | | 832 | | | | 192 | 768 | 11.06 | 164.00 | |
| | *SHA3-512* | 512 | | 576 | | | | 256 | 1024 | 15.88 | 164.00 | |
| | *SHAKE128* | *d* (arbitrary) | | 1344 | | | | min(*d*/2, 128) | 256 | 7.08 | 155.25 | |
| | *SHAKE256* | *d* (arbitrary) | | 1088 | | | | min(*d*/2, 256) | 512 | 8.59 | 155.50 | |

# Blockchains: SHA-1



Main Loop

For example, the main loop of SHA-1 (a cryptographic hash function) has a non-linear step named F that is composed of ANDs, ORs, and XORs, depending on which round of the algorithm you're in (from Wikipedia):

```
Main loop:
    for i from 0 to 79
        if 0 ≤ i ≤ 19 then
            f = (b and c) or ((not b) and d)
            k = 0x5A827999
        else if 20 ≤ i ≤ 39
            f = b xor c xor d
            k = 0x6ED9EBA1
        else if 40 ≤ i ≤ 59
            f = (b and c) or (b and d) or (c and d)
            k = 0x8F1BBCDC
        else if 60 ≤ i ≤ 79
            f = b xor c xor d
            k = 0xCA62C1D6

        temp = (a leftrotate 5) + f + e + k + w[i]
        e = d
        d = c
        c = b leftrotate 30
        b = a
        a = temp
```

SHA-1 is not unique in this regard. Many algorithms based around Feistel ciphers have a non-linear step, and that non-linear step can be realized with AND and OR. That's because the F function in a Feistel cipher round step need

# Blockchains: SHA-1



Body

```
Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit big-endian words w[i], 0 ≤ i ≤ 15

    Extend the sixteen 32-bit words into eighty 32-bit words:
    for i from 16 to 79
        w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1

    Initialize hash value for this chunk:
    a = h0
    b = h1
    c = h2
    d = h3
    e = h4

    Main loop:[3][55]
    for i from 0 to 79
        if 0 ≤ i ≤ 19 then
            f = (b and c) or ((not b) and d)
            k = 0x5A827999
        else if 20 ≤ i ≤ 39
            f = b xor c xor d
            k = 0x6ED9EBA1
        else if 40 ≤ i ≤ 59
            f = (b and c) or (b and d) or (c and d)
            k = 0x8F1BBCDC
        else if 60 ≤ i ≤ 79
            f = b xor c xor d
            k = 0xCA62C1D6

        temp = (a leftrotate 5) + f + e + k + w[i]
        e = d
        d = c
        c = b leftrotate 30
        b = a
        a = temp
```