

# AP Computer Science A

Java Programming Essentials [Ver.4.0]

Unit 2: Selection and Iterations

CHAPTER 9: LOOP PROCESSING

DR. ERIC CHOU

IEEE SENIOR MEMBER



# AP Computer Science Curriculum

- Implementing String Algorithms (T 2.10)
- Nested Iteration (T 2.11)
- Informal Run-Time Analysis (T 2.12)

# Objectives:

- Nested Loops
- 2-D Index Space for Loops
- Loop Processing 1: Prime Number, GCD, Compounding, and Palindrome.
- Loop Processing 2: Reverse of String, Decimal to Hexadecimal, Sum of the Digits, and Histogram.
- Lab Projects: Two Dice Project, Strong Password Project



# Nested Loop

Lecture 1

# Nested Loop

- Nested loops consist of an outer loop and one or more inner loops. Each time the outer loop is repeated, the inner loops are reentered and started anew.
- Nested while-loop might have multiple control variables (done, play, found, ...)
- Nested for-loops have multiple control index variables.



# Demonstration Program Multiplication Table

MultiplicationTable.java

# Nested Loop

## MultiplicationTable.java

- Problem: Write a program that uses nested for loops to print a multiplication table.
- Outer Loop: first multiplier as row control variable
- Inner Loop: second multiplier as column control variable
- first multiplier \* second multiplier = result and
- print it out.
- end inner loop.
- End outer loop.

# MultiplicationTable.java

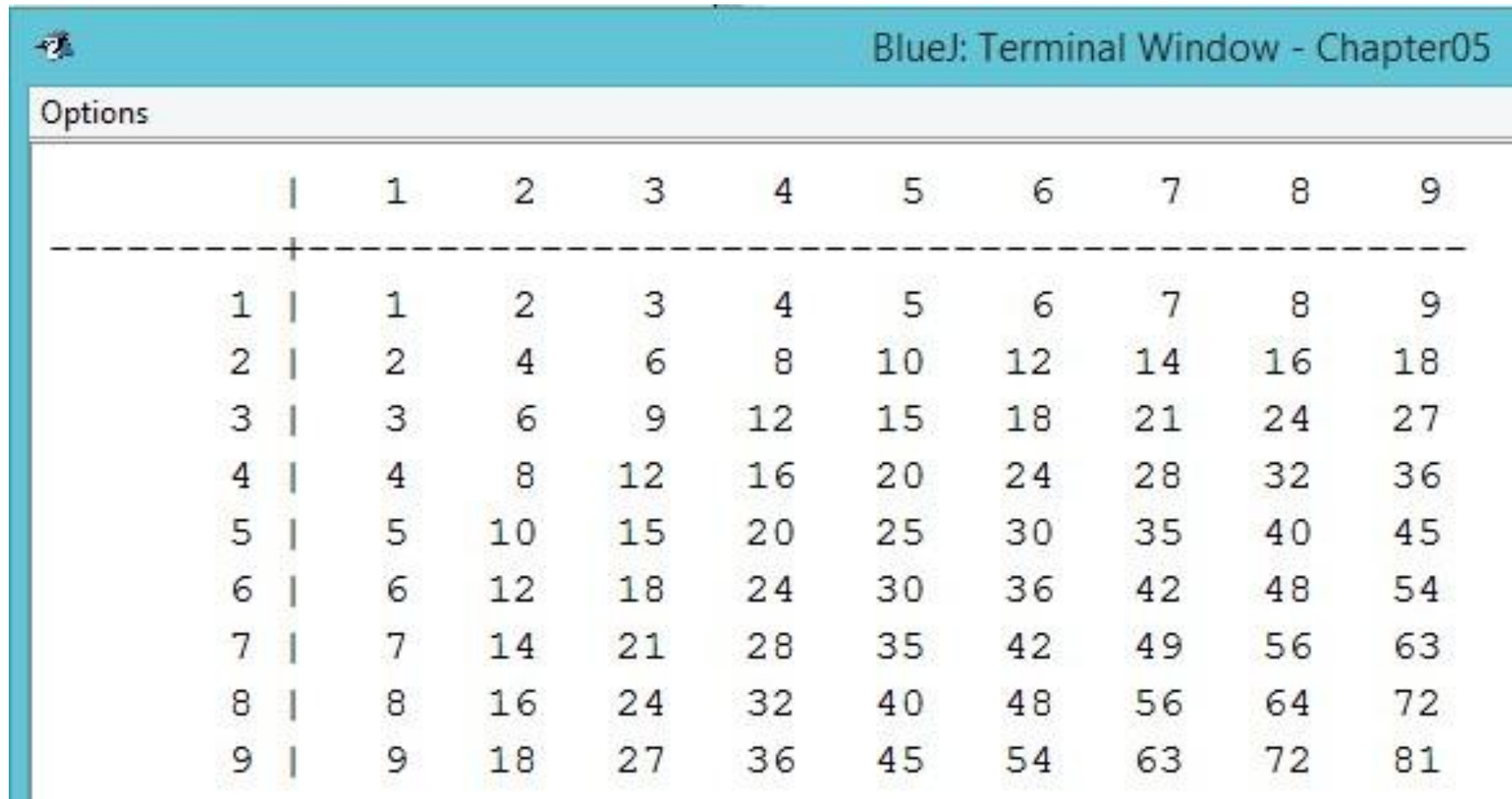
```
public static void main(String[] args){
    // column index line
    System.out.printf("%8s |", " ");
    for (int i=1; i<=9; i++){
        System.out.printf(" %3d ", i);
    }
    System.out.println();
    // header divider
    System.out.print("-----+");
    for (int i=1; i<=9; i++){
        System.out.printf("-----");
    }
    System.out.println();

    // multiplication table
    for (int i=1; i<=9; i++){
        // row index
        System.out.printf("%8d |", i);
        // multiplication cell
        for (int j=1; j<=9; j++){
            System.out.printf(" %3d ", i*j);
        }
        System.out.println();
    }
}
```



# Result:

(download **MultiplicationTable.zip** and try)



BlueJ: Terminal Window - Chapter05

Options

		1	2	3	4	5	6	7	8	9
1		1	2	3	4	5	6	7	8	9
2		2	4	6	8	10	12	14	16	18
3		3	6	9	12	15	18	21	24	27
4		4	8	12	16	20	24	28	32	36
5		5	10	15	20	25	30	35	40	45
6		6	12	18	24	30	36	42	48	54
7		7	14	21	28	35	42	49	56	63
8		8	16	24	32	40	48	56	64	72
9		9	18	27	36	45	54	63	72	81



# 2-D Index Space

Lecture 2

# Index Space Chart (Rectangular Shaped)

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5								

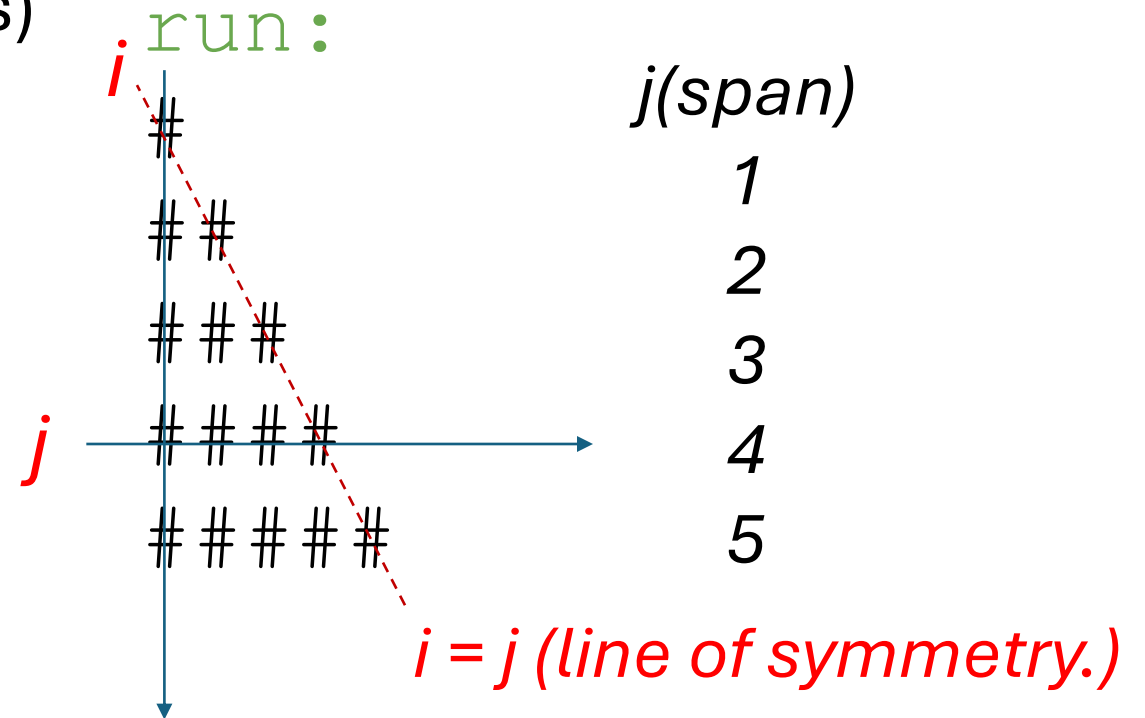
# Print Pyramids

(Good Example for Index Chart)

Index Sweeping Line  
Index Ending Line  
Span, boundary

```
public class Pyramids {  
    public static void main(String[] args)  
    {  
        for (int i = 1; i <= 5; i++) {  
            for (int j = 1; j <= i; j++) {  
                System.out.print("#");  
            }  
            System.out.println("");  
        }  
    }  
}
```

Output of this program :



# Print Pyramids

## (Good Example for Index Chart)

```
public class Pyramid2 {
    public static void main(String[] args) {
        for (int i = 5; i >= 0; i--) {
            for (int s = 1; s < i; s++) {
                System.out.print(" ");
            }
            for (int j = 5; j >= i; j--) {
                System.out.print("#");
            }
            System.out.println("");
        }
    }
}
```

Output of this program :

run:

	<i>i</i>	<i>j</i>	<i>s(span)</i>	<i>j(span)</i>
#	5	5	4	1
# #		5	3	2
# # #		5	2	3
# # # #		5	1	4
# # # # #		5	0	5
# # # # # #	0	5	0	6

*s=i-1*

*s=0*

*s=0*

# Print Pyramids

## (Good Example for Index Chart)

```
public class Pyramid3 {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 5; i++) {  
            for (int s = 1; s < i; s++) {  
                System.out.print(" ");  
            }  
            for (int j = 5; j >= i; j--) {  
                System.out.print("#");  
            }  
            System.out.println("");  
        }  
    }  
}
```

## Output of this program :

run: *s (span)* *j (span)*

<i>i</i>	#####	0	5
	#####	1	4
	#####	2	3
	#####	3	2
	#####	4	1



# Print Pyramids

## (Good Example for Index Chart)

```
public static void main(String[] args){  
    for (int i = 1; i <= 5; i++) {  
        for (int s = 5; s > i; s--) {System.out.print(" ");}  
        for (int j = 1; j < i; j++) {System.out.print("#");}  
        for (int j = 1; j < i; j++) {System.out.print("#");}  
        System.out.println("");  
    }  
    for (int i = 1; i <= 5; i++) {  
        for (int s = 1; s < i; s++) {System.out.print(" ");}  
        for (int j = 5; j > i; j--) { System.out.print("#");}  
        for (int j = 5; j > i; j--) {System.out.print("#");}  
        System.out.println("");  
    }  
}
```

<i>Span(" ")</i>	<i>Span("#")</i>	<i>Span("#")</i>
4	0	0
3	1	1
2	2	2
1	3	3
0	4	4

<i>Span(" ")</i>	<i>Span("#")</i>	<i>Span("#")</i>
0	4	4
1	3	3
2	2	2
3	1	1
4	0	0

# Print Pyramids

(Good Example for Index Chart)

Output of this program :

```

      # #
    _# #
  _# # #
_# # # #
# # # # #
# # # # #
_# # #
  _# #
    _#
      #
  
```

<i>Span(" ")</i>	<i>Span("#")</i>	<b><i>Span("#")</i></b>
4	0	<b>0</b>
3	1	<b>1</b>
2	2	<b>2</b>
1	3	<b>3</b>
0	4	<b>4</b>

<i>Span(" ")</i>	<i>Span("#")</i>	<b><i>Span("#")</i></b>
0	4	<b>4</b>
1	3	<b>3</b>
2	2	<b>2</b>
3	1	<b>1</b>
4	0	<b>0</b>



# Number Ladder

```
public static void main(String [] args){  
    for (int i=1; i<=9; i++){  
        System.out.println();  
        for (int j=1; j<=i; j++){  
            System.out.print(j);  
        }  
    }  
    System.out.println();  
}
```

<i><b>i</b></i>	<i><b>j(first)</b></i>	<i><b>j(last)</b></i>	<i><b>span</b></i>
1	1	1	1
2	1	2	2
3	1	3	3
4	1	4	4
5	1	5	5
6	1	6	6
7	1	7	7
8	1	8	8
9	1	9	9

# Number Ladder

1

12

123

1234

12345

123456

1234567

12345678

123456789

<i><b>i</b></i>	<i><b>j(first)</b></i>	<i><b>j(last)</b></i>	<i><b>span</b></i>
1	1	1	1
2	1	2	2
3	1	3	3
4	1	4	4
5	1	5	5
6	1	6	6
7	1	7	7
8	1	8	8
9	1	9	9

**$(i, j)$**  is an index-pair (or ***discrete vector***)  
to access some 2-D array or create values (multiplication table)

- 1. index space chart show people what is the region of array cells or index region got accessed by the nested loop.
- 2. create value such as  $i + j$ ,  $i - j$ ,  $i * j$ ,  $i / j$ ,  $i^j$ , ...
- 3. access 2-D array cells ***ary[i][j]***



# Loop Processing I

Lecture 3



# Demonstration Program Prime Number

PrimeNumber.java

# PrimeNumber.java

## Displaying Prime Number

- An integer greater than 1 is a prime if its only positive divisor is 1 or itself.
- The problem is to display the first 50 prime numbers in five lines, each of which contains ten numbers. The problem can be broken down into the following tasks:
  - Determine whether a given number is prime.
  - For number = 2, 3, 4, 5, 6, ..., test whether it is prime.
  - Count the prime numbers.
  - Display each prime number, and display ten numbers per line.

The prime number count is initially 0. It is updated whenever it is updated. When it reaches 50, the loop terminates.

# Algorithm for PrimeNumber.java

- Set the number of prime numbers to be printed as a constant `NUMBER_OF_PRIMES`;
- Use `count` to track the number of prime numbers and set an initial count to 0;
- Set an initial number to 2;
- while (`count` < `NUMBER_OF_PRIMES`) {
  - Test whether `number` is prime;
  - if number is prime {Display the prime number and increase the `count`; }
  - increase the `number` by 1;
- }



# Demonstration Program GCD Loop

GCD.java



# Problem: GCD.java

## Finding the Greatest Common Divisor

- Problem: Write a program that prompts the user to enter two positive integers and finds their greatest common divisor.
- Solution: Suppose you enter two integers 4 and 2, their greatest common divisor is 2. Suppose you enter two integers 16 and 24, their greatest common divisor is 8. So, how do you find the greatest common divisor? Let the two input integers be  $n1$  and  $n2$ . You know number 1 is a common divisor, but it may not be the greatest common divisor. So you can check whether  $k$  (for  $k = 2, 3, 4$ , and so on) is a common divisor for  $n1$  and  $n2$ , until  $k$  is greater than  $n1$  or  $n2$ .
- Also, try Euclidean Algorithm.
- `while (m != 0) {`
- `n, m = m, n%m;`
- `}`



# Demonstration Program Future Value (Compounding)

FutureTuition.java

# FutureTuition.java

## Problem: Predicting the Future Tuition

- Problem: Suppose that the tuition for a university is \$10,000 this year and tuition increases 7% every year. In how many years will the tuition be doubled?

```
double tuition = 10000; int year = 0 // Year 0
```

```
tuition = tuition * 1.07; year++;    // Year 1
```

```
tuition = tuition * 1.07; year++;    // Year 2
```

```
tuition = tuition * 1.07; year++;    // Year 3
```

```
...
```



# Demonstration Program Palindrome

Palindrome.java

# Demo Program: Palindrome.java

- Given a string, write a c function to check if it is palindrome or not.
- A string is said to be palindrome if reverse of the string is same as string. For example, “abba” is palindrome, but “abbc” is not palindrome.

Input : ABCDCBA

Output : Yes

If we read string from end to begin, it is same as begin to end.

Input : GeeksforGeeks

Output : No



# Loop Processing II

Lecture 4



# Demonstration Program Reverse of Strings

ReverseOfString.java



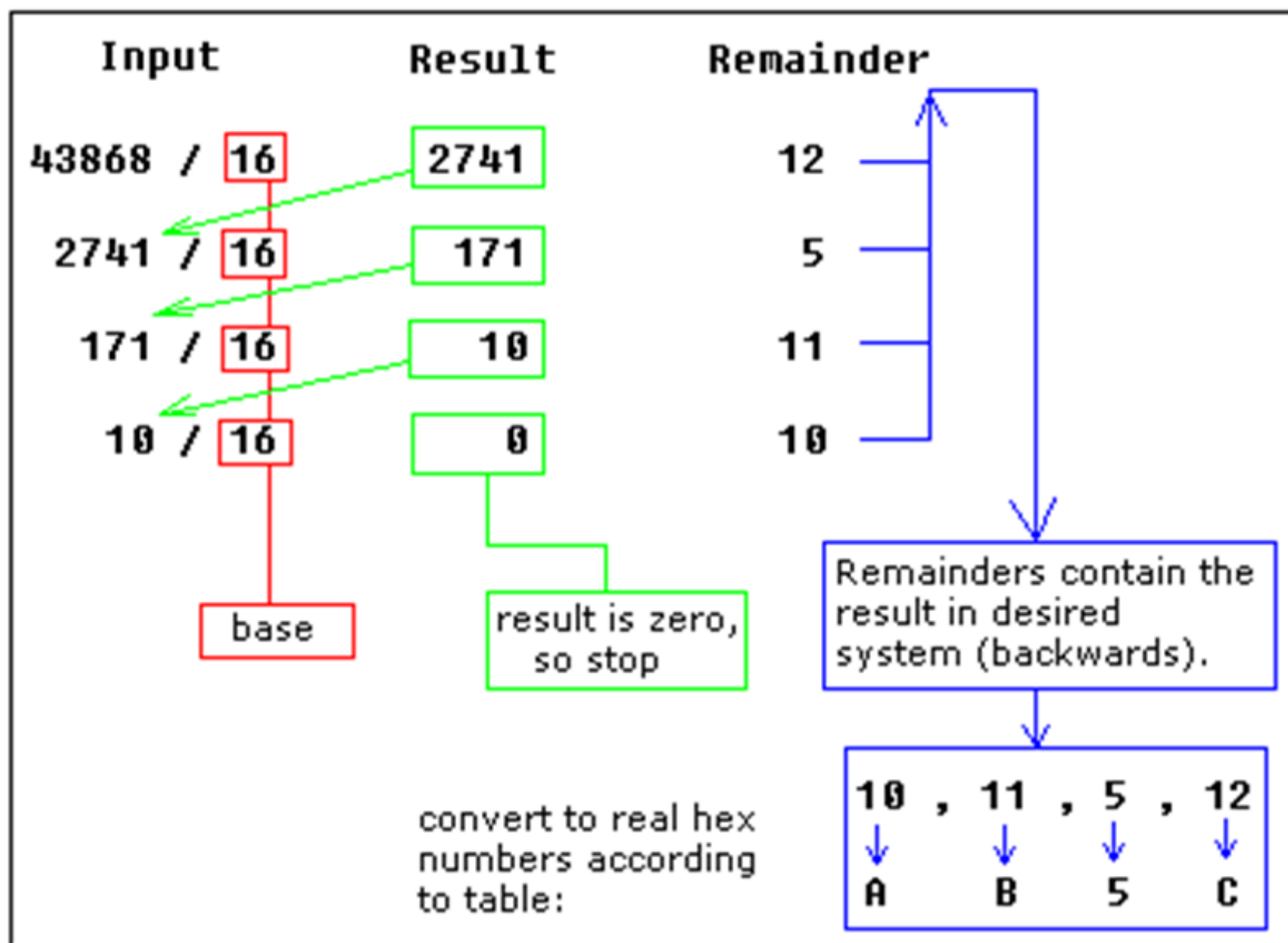
# Demonstration Program Decimal to Hexadecimal

Dec2Hex.java



# Dec2Hex.java

- Hexadecimals are often used in computer systems programming. How do you convert a decimal number to a hexadecimal number? To convert a decimal  $d$  to a hexadecimal number is to find the hexadecimal digits  $d \rightarrow h_n, h_{n-1}, h_{n-2}, \dots, h_2, h_1, h_0$
- $d = h_n \times 16^n + h_{n-1} \times 16^{n-1} + \dots + h_2 \times 16^2 + h_1 \times 16^1 + h_0 \times 16^0$
- `q = d; /* Pseudo code version */`
- `while (q != 0) {`
- `q = d // 16;`
- `r = d % 16; /* get a digit for h = r */`
- `}`





# Demonstration Program Sum of the Digits of a Number

SumofDigits.java

# Sum of Digits

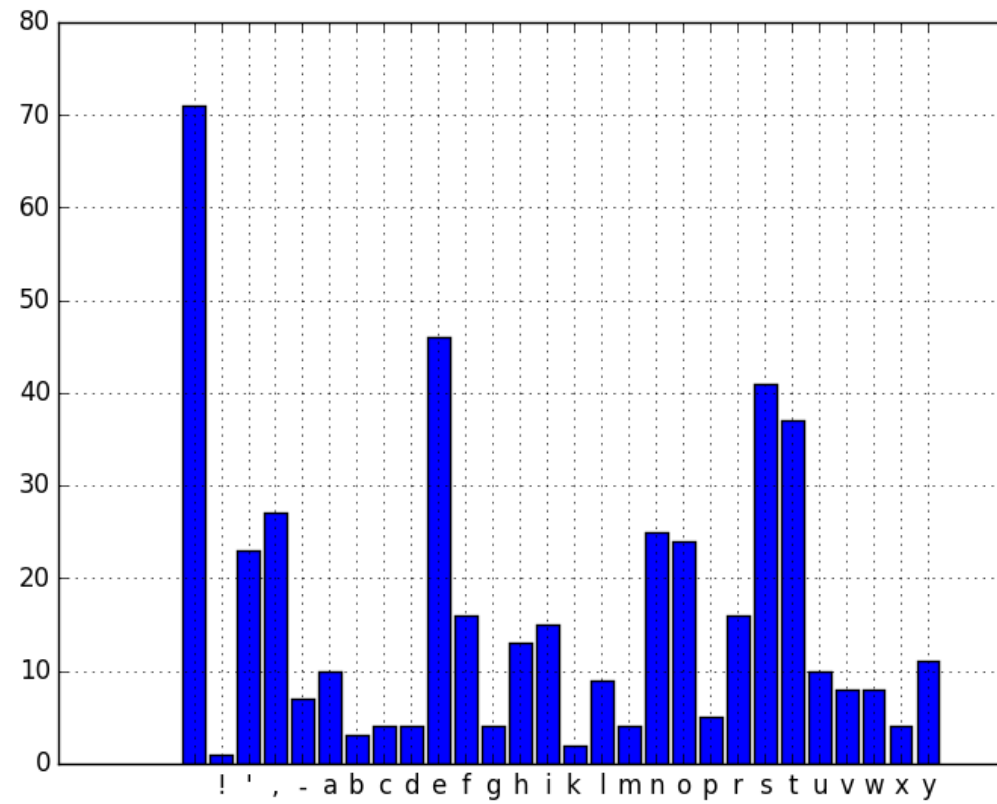
- How do you calculate the sum of the digits for a decimal number?  
To convert a decimal  $d$  to a hexadecimal number is to find the hexadecimal digits  $x \rightarrow d_n, d_{n-1}, d_{n-2}, \dots, d_2, d_1, d_0$
- $x = d_n \times 10^n + d_{n-1} \times 10^{n-1} + \dots + d_2 \times 10^2 + d_1 \times 10^1 + d_0 \times 10^0$
- $q = d;$  /\* Pseudo code version \*/
- while ( $q \neq 0$ ) {
- $d = q \% 10;$
- $\text{sum} += d;$
- $q = q // 10;$
- }



# Demonstration Program Histogram

Histogram.java

# Demo: Histogram.java



```
public static void main(String[] args){  
    System.out.print("\f");  
    for (int i=0; i<s.length(); i++){  
        if (Character.isLetter(s.charAt(i))){  
            freq[Character.toUpperCase(s.charAt(i))-'A']++;  
        }  
    }  
    for (int i=0; i<26; i++){  
        System.out.printf("Character %c occurs %d times.%n", (char) ('A'+i), freq[i]);  
    }  
}
```



# Chapter project 1: Two dice Sum

Lecture 5



# Chapter Project: TwoDice.java

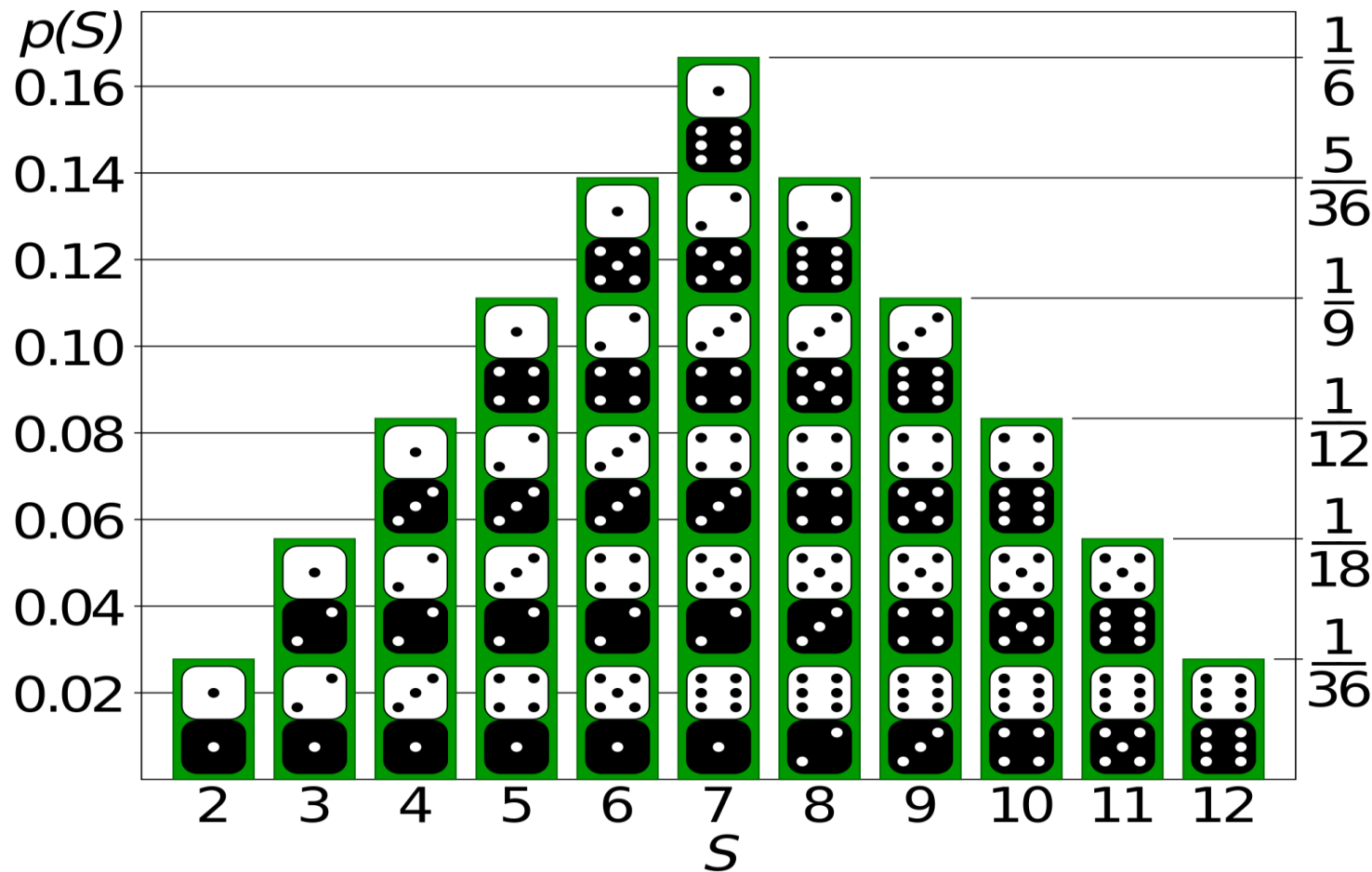
- Using nested loops, cycle through the possible sums of two dice.
- **Outer loop:** loop index variable on all the possible sums of two dice.
- **Inner loop:** loop index each individual two dice combination for totally **rolls** (a variable input by user) rolls.
- sum up the number of rolls that match the sum of the two dice of a roll to the outer loop index variable.
- Then, calculate the probability of having the specific sum of the two dice by dividing the sum of the two dice over **rolls**. Then, this probability times 100, we get the percentage of getting the specific sum over **rolls**.

# Chapter Project: TwoDice.java

- At the end, print out the percentage (probability) of each sum.

# Background Information:

## Theoretical probability distribution of sum of a pair of dice.



The probability by Monte Carlo Simulation should be approaching these number when the number of **rolls** increase to be more than 1000. The larger number of **rolls**, the closer the probability should be.

# Chapter Project: TwoDice.java

- Write a program to ask user to input the **sides** of a die, and the total number of **rolls** for each possible sum of a two dice combination.
- Then, find out the probability of each possible sum of two dice by a Monte Carlo Simulation Method which is to use random number generators (Random Class required), to simulate for **rolls** rolls.
- Calculate the probability of the sum by dividing the number of cases, which has the sum of two random dice equals to the sum of current iteration, over the total number of **rolls**.
- Then, print out all of the probabilities of each possible sums of two dice.

# Expected Results:

## (6 sides, start from this case first)

```
BlueJ: Terminal Window - Chapter05
Options
*****
Using nested loops, cycle through the possible sums of the dice.
Roll the dice the given number of times for each sum.
Count how many times the sum of the dice match the current sum being looked for.
*****
Please enter the number of Rolls: 1000

Please enter the number of sides on a die: 6

Sum of Dice      Probability
2s:              2.100%
3s:              6.600%
4s:              8.100%
5s:              12.300%
6s:              13.100%
7s:              16.100%
8s:              13.700%
9s:              9.800%
10s:             9.500%
11s:             5.000%
12s:             2.800%
```

# Expected Results: more rolls (6 sides, start from this case first)

```
Blue: Terminal Window - Chapter05
Options
*****
Using nested loops, cycle through the possible sums of the dice.
Roll the dice the given number of times for each sum.
Count how many times the sum of the dice match the current sum being looked for.
*****
Please enter the number of Rolls: 10000000

Please enter the number of sides on a die: 6

Sum of Dice      Probability
2s:              2.785%
3s:              5.554%
4s:              8.338%
5s:              11.131%
6s:              13.890%
7s:              16.672%
8s:              13.898%
9s:              11.112%
10s:             8.330%
11s:             5.557%
12s:             2.777%
```

Options

```
*****
Using nested loops, cycle through the possible sums of the dice.
Roll the dice the given number of times for each sum.
Count how many times the sum of the dice match the current sum being looked for.
*****
Please enter the number of Rolls: 100000
```

```
Please enter the number of sides on a die: 8
```

Sum of Dice	Probability
2s:	1.549%
3s:	3.204%
4s:	4.725%
5s:	6.158%
6s:	7.796%
7s:	9.319%
8s:	10.972%
9s:	12.447%
10s:	10.738%
11s:	9.516%
12s:	7.893%
13s:	6.131%
14s:	4.585%
15s:	3.125%
16s:	1.562%

Try different number of sides  
(8 in this case)

# Pseudo Code (for your reference)

- Get a seed number from `System.currentTimeMillis();`
- Declare input stream
- Declare random number input stream
- Declare match as number of match for the sum of dice
- Declare double type prob as the probability of the matches over total rolls.
- Declare int data type for two dice die1 and die2.
- Print out header information.
- Ask user to enter number of **rolls** and **sides** of a die.



# Pseudo Code (for your reference)

- for (sum = 2; sum<=sides \* 2; sum++) { // from sum = 2 to sides \* 2 are the possible outcomes
- reset math to 0; // reset match so that the number match has a new start for a new outcome.
- for (int i = 0; i < **rolls**; i++) { // for each possible outcome, rolls for **rolls** times.
  - 1. generate two dice value and sum them up to sumDie
  - 2. if (sumDie == sum) then increase match number
  - }
- calculate the probability = match / rolls
- print out the outcome and its simulated probability in



# Chapter project 2: Strong Password

Lecture 6

# Microsoft Strong Password Test

<https://www.microsoft.com/es-xl/security/pc-security/password-checker.aspx>

Passwords are the first line of defense against break-ins to your online accounts and computer, tablet, or phone. Poorly chosen passwords can render your information vulnerable to criminals, so it's important to make your passwords strong.

**Another site:**

<https://www.microsoft.com/security/pc-security/password-checker.aspx>

# Create strong passwords

To help you create strong passwords, follow the same network security guidelines:

- Strong passwords are phrases (or sentences) at least eight characters long—longer is better—that include at least **three** of the following: **uppercase** and **lowercase** letters, **numerals**, **punctuation marks**, and **symbols**.
- Give passwords the thought they deserve, and make them memorable. One way is to base them on the title of a favorite song or book, or a familiar slogan or other phrase.

# Generation of a random letter (symbol)

```
passLower = (char) ((int)(Math.random()*26)+ (int) 'a'); // 97-122
```

```
passUpper = (char) ((int)(Math.random()*26)+ (int) 'A'); // 65-90
```

```
passNumber = (char) ((int)(Math.random()*10)+ (int) '0'); // 48-57
```

```
// passPunc for punctuation Marks (unbiased randomness)
```

```
punc = ((int) (Math.random()*32));
```

```
if (punc < 15) {passPunc = (char)(punc+(int) '!'); } // region 1 for punctuation 33 to 47
```

```
else if (punc >=15 && punc < 22) { passPunc = (char) ((punc-15) + (int) ':'); }
```

```
// region 2 for punctuation 58 - 64
```

```
else if (punc >=22 && punc < 28) { passPunc = (char) ((punc-22) + (int) '['); }
```

```
// region 3 for punctuation 91 - 96
```

```
else { passPunc = (char) ((punc-28) + (int) '{'); } // region 4 for punctuation 123-126
```

# Random punctuation marks and symbols

- The 4 regions for random symbols can be further divided into 4 independent regions if finer password generator is needed or a system do not allow certain one sub-region among these four.
- In this project, we assume there is nothing like this required.

# ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

# Rough Pseudo Code (part 1)

do {

1. display menu for user to pick one of the password format.
2. ask user to input password length. If not long enough for all combinations

set

longEnough to be false, otherwise, true. Whether the password length is longEnough

depends on the selection of password format.

3. Enter into a for loop:

for (i=0; i< length; i++) {

// each time generate a symbol and append it to the resulting password

3.1 generate random candidate symbols for uppercase, lowercase, numerals, and symbols cases.



# Rough Pseudo Code (part 2)

```
    3.2 if selection one (only lowercase) then use the lowercase symbol.  
        if selection two (lowercase and uppercase): The first letter must be  
        Uppercase.  
            The last letter must be lowercase. Other letters randomly pick among upper  
            and lower.  
        if selection three (lowercase, uppercase and numerals): The first letter must  
        be  
            Uppercase. The second letter must be lower. The last one numeral. Others,  
            random pick.  
        if selection four (lower, upper, numbers and symbols): The first letter must be  
            upper. Second, lower. The one before last, numeral. The last one, symbol.  
            Others, random pick.  
    3.3 append the letter symbol pick to the password  
    }  
} while (!done)
```

# Password format

In this program, if the password length is entered properly.  
The program will generate a password in the following format.

- \* option 1: **LLLLLL**
- \* option 2: **URRRRL**
- \* option 3: **ULRRRN**
- \* option 4: **ULRRNP**

Note: L:Lowercase, U:Uppercase, N:Numbers, P:Punctuation,  
R:Random Pick at the specific strength level.

# Chapter Project:

- Write a program to generate passwords for 4 different optional formats (in a selection menu):
  - (1) lower case letters only
  - (2) upper case and lower case letters.
  - (3) uppercase, lowercase, and numerals
  - (4) uppercase, lowercase, numerals and symbols
  - (5) quit
- Password length allowed (1-100). Format requirements as mentioned in previous slide.

# Expected Result:

```
Password Generation Menu
[1] Lowercase Letters
[2] Lowercase & Uppercase Letters
[3] Lowercase, Uppercase, and Numbers
[4] Lowercase, Uppercase, Numbers, and Punctuation
[5] Quit
Enter Selection(1-5): 1

Password Length(1-100): 20

password: pvputgbnlvkoxtixsrIt
```

```
Password Generation Menu
[1] Lowercase Letters
[2] Lowercase & Uppercase Letters
[3] Lowercase, Uppercase, and Numbers
[4] Lowercase, Uppercase, Numbers, and Punctuation
[5] Quit
Enter Selection(1-5): 2

Password Length(1-100): 20

password: UqSTDJML00eWCZawupFg
```

# Expected Result:

```
Password Generation Menu
[1] Lowercase Letters
[2] Lowercase & Uppercase Letters
[3] Lowercase, Uppercase, and Numbers
[4] Lowercase, Uppercase, Numbers, and Punctuation
[5] Quit
Enter Selection(1-5): 3

Password Length(1-100): 20

password: Am4bb4LTb40Z435e1li9
```

## How strong is your password?

Type a password into the box.

Password:

Strength:



```
Password Generation Menu
[1] Lowercase Letters
[2] Lowercase & Uppercase Letters
[3] Lowercase, Uppercase, and Numbers
[4] Lowercase, Uppercase, Numbers, and Punctuation
[5] Quit
Enter Selection(1-5): 4

Password Length(1-100): 20

password: XtBO8=5Qb4tRB-2Vi20=
```