

AP Computer Science A

Java Programming Essentials [Ver. 2.0]

Unit 2: Structured Programming

WEEK 7: CHAPTER 5 PART 1: LOOPS (BASIC LOOPS)

DR. ERIC CHOU

IEEE SENIOR MEMBER



Objectives

- Basic while-loop, do-while loop and for-loop
- Repetition Loop (While Loop)
- Statistics Loops, Sentinel Loop,
- Tokenization Loop, Menu Selection Loop, Input Validation Loop
- Counting Loop, Indexed Loop and Tabularization Loop (Sine)



While-Loop

LECTURE 1



Motivation

Suppose that you need to print a string (e.g., "Welcome to Java!") a hundred times. It would be tedious to have to write the following statement a **hundred times**:

```
System.out.println("Welcome to Java!");
```

So, how do you solve this problem?



Motivation

100 times

```
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
...  
...  
...  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");
```



Solution to it: while-loop

```
int count = 0;
while (count < 100) {
    System.out.println("Welcome to Java");
    count++;
}
```

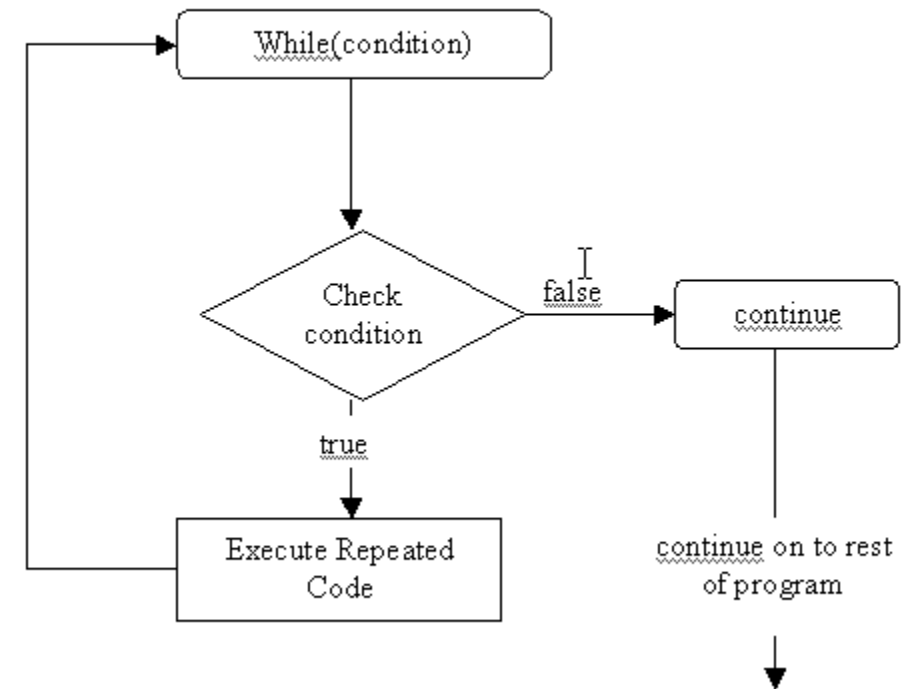


while loop

The syntax for the while loop is:

```
while (loop-continuation-condition) {  
    // loop body  
    Statement(s);  
}
```

Flow Diagram of a while loop





Comparison of if-statement and while-loop

```
int x = 0;           // if-statement
if (x < 10) {
    System.out.println("Welcome to Java.");
}
```

```
int x = 0;           // while-loop
while (x < 10) {
    System.out.println("Welcome to Java.");
    x++;
}
```




Trace while-loop

```
int count = 0;
```

Initialize count

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```



Trace while-loop

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

(count < 2) is true



Trace while-loop

```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```




Print Welcome to Java



Trace while-loop

```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!")  
    count++;  
}
```



Increase count by 1
count is 1 now



Trace while-loop

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

(count < 2) is still true since count
is 1



Trace while-loop


```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```

Print Welcome to Java



Trace while-loop

```
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```



Increase count by 1
count is 2 now



Trace while-loop

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

(count < 2) is false since count is 2
now



Trace while-loop

```
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

The loop exits. Execute the next statement after the loop.



Demonstration Program

REPETITION.JAVA



Basic Statistics

Loops

Sum/Avg/Min

/Max

LECTURE 2



Lab Project: SumAvgMaxMin.java

- Write a program to take in a series of data from user's input and find their sum, average, maximum and minimum. Then, print these data out.
- After finish it for while-loop, re-write it in do-while-loop format.
- A skeleton java program file is in SumAvgMaxMin.java. The sample answer is in SumAvgMaxMinAnswer.java



Lab Project: SumAvgMaxMin.java

- This project is to exercise the basic while loop / do while loop programming skill.
- It will calculate only the sum. Please figure out how to get the average, maximum and minimum of the value.



Pseudo Code

Pseudo Code for Sum and Average:

```
ASSIGN false TO Done
ASSIGN 0    TO Count
ASSIGN 0    TO Sum
WHILE not Done DO
    GET Num
    ASSIGN ( Sum + Num ) TO Sum
    GET User_Input_To_Continue
    IF User_Input_To_Continue THEN ASSIGN false TO Done
                                ELSE ASSIGN true TO Done
    INCREASE Count
END
ASSIGN Sum/Count TO Average
```



Pseudo Code

Pseudo Code for Maximum and Minimum:

```
ASSIGN false TO Done
ASSIGN Integer.MIN_VALUE TO Max
ASSIGN Integer.MAX_VALUE TO Min
WHILE not Done DO
    GET Num
    IF Num > Max THEN ASSIGN Num TO Max
    IF Num < Min THEN ASSIGN Num TO Min
    GET User_Input_To_Continue
    IF User_Input_To_Continue
        THEN ASSIGN false TO Done
        ELSE ASSIGN true TO Done
END
```



In-Class Demonstration Program

SUM.JAVA, AVG.JAVA,
MIN.JAVA, MAX.JAVA



Lab

SUM/AVG/MIN/MAX

Expected Results

1. Counted Loop
2. Taking sum, average, max, min

```
Blue: Terminal Window - Chapter05
Options

Program to find sum, avarage, maximum, and minimum starts ...
Enter an integer: 5

Do you want to quit (Y/N)? n

Enter an integer: 6

Do you want to quit (Y/N)? n

Enter an integer: 7

Do you want to quit (Y/N)? y

Results for while loop ...
Sum:          18.000
Avarage:       6.000
Maximum:       7
Minimum:       5

do while loop starts here ...
Enter an integer: 3

Do you want to quit (Y/N)? n

Enter an integer: 4

Do you want to quit (Y/N)? n

Enter an integer: 5

Do you want to quit (Y/N)? y

Results for do while loop ...
Sum:          12.000
Avarage:       4.000
Maximum:       5
Minimum:       3
```



Sentinel Loop

LECTURE 3



Sentinel Value

- In programming, sentinel value is a special value that is used to terminate a loop.
- The sentinel value typically is chosen so as to not be a legitimate data value that the loop will encounter and attempt to perform with.
- For example, in a loop algorithm that computes non-negative integers, the value "-1" can be set as the sentinel value as the computation will never encounter that value as a legitimate processing output.



Demonstration Program

SENTINELVALUE.JAVA



Do-While Loop

LECTURE 4



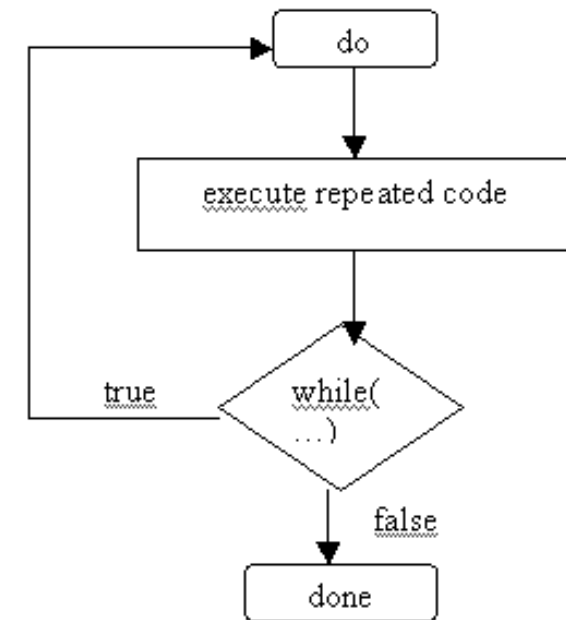
do ... while loop

The do-while loop is a variation of the while loop.

Its syntax is:

```
do {  
    // loop body;  
    statement(s);  
} while (loop-continuation-condition);
```

Flow Diagram of do .. while LOOP





Difference between do-while-loop and while-loop

The difference between a while loop and a do-while-loop is the order in which the loop-continuation-condition is evaluated and the loop body executed. You can write a loop using either the while-loop or the do-while loop. Sometimes one is a more convenient choice than the other.



Demonstration Program

SENTINELVALUE2.JAVA



Demonstration Program

REPEATADDITIONQUIZ.JAVA



Addition Quiz:

- Randomly generate two integer for addition problem and to ask for the user to enter an answer. If he gets it right, the program stop. Otherwise, the program repeats the question for the user.
- New Updates:
 - (1) Use of While Loop.
 - (2) Use of Random Class.
 - (3) Add Difficulty Level for the program:
 - Level 0: 0 – 9 for the operands.
 - Level 1: 0 – 99 for the operands
 - Level 2: -99 to 99 for the operands



Demo Program: FILE I/O

LECTURE 5

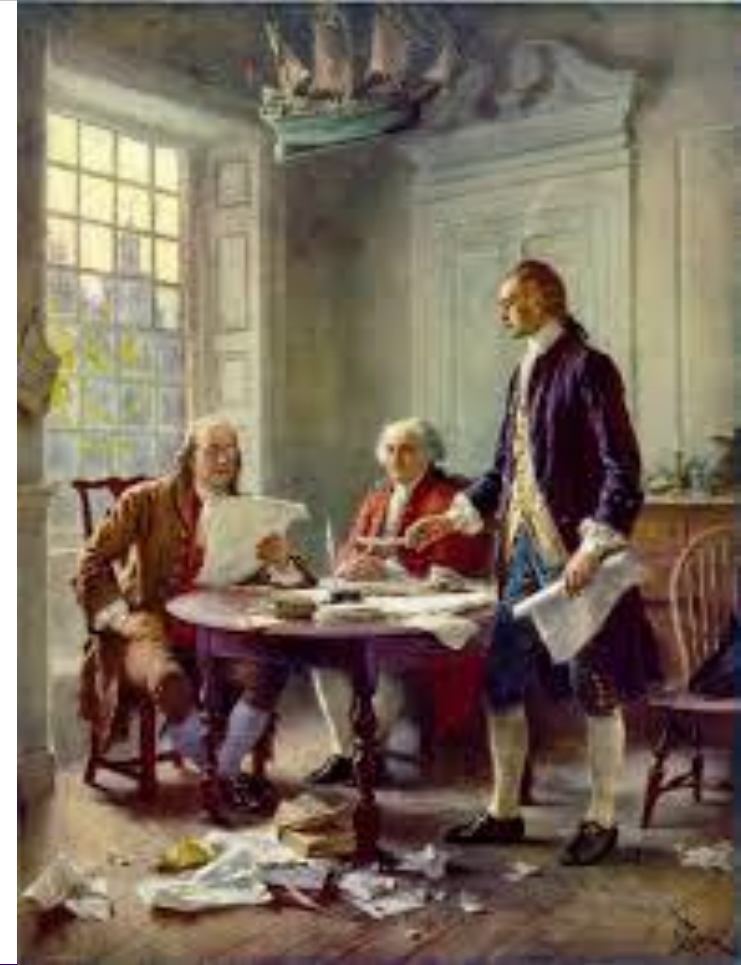


Declaration Of Independence

The Unanimous Declaration of the Thirteen United States of America

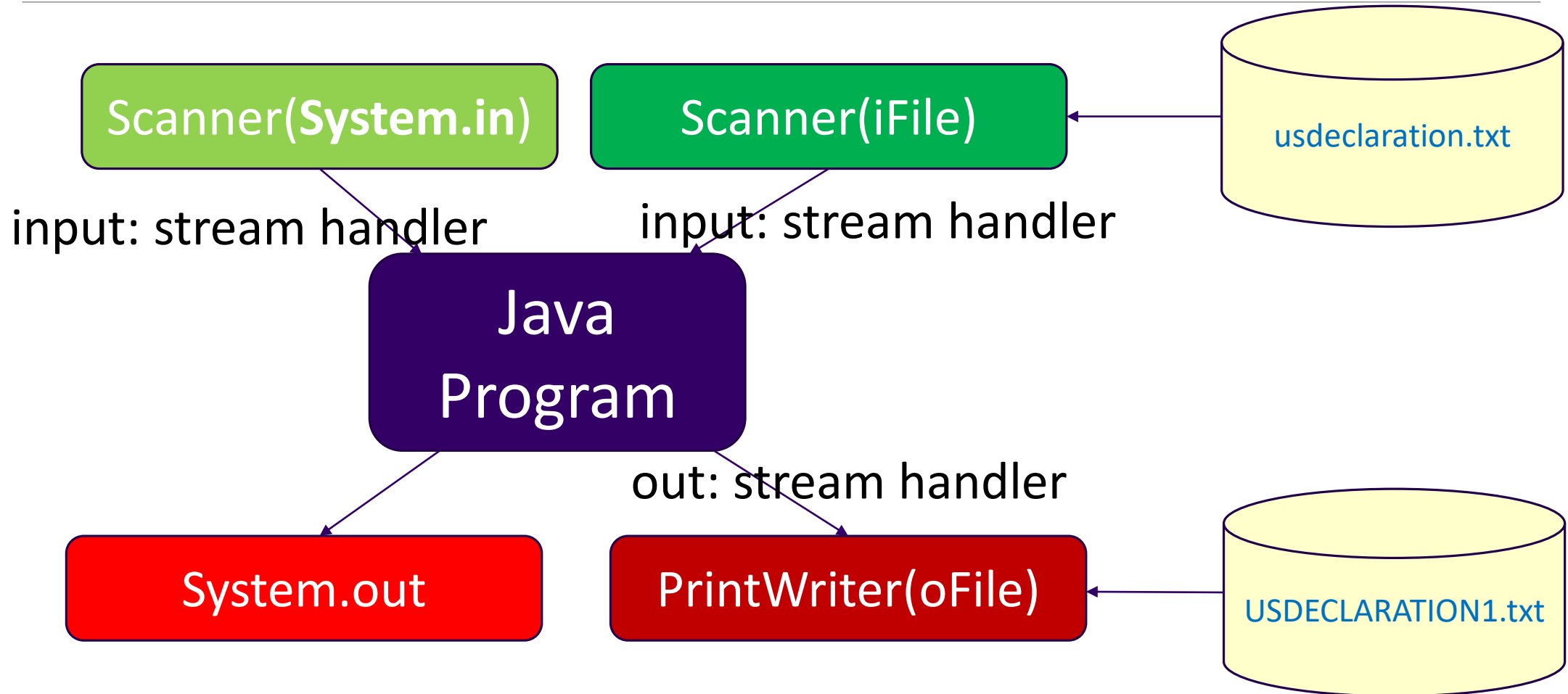
When, in the course of human events, it becomes necessary for one people to dissolve the political bands which have connected them with another, and to assume among the powers of the earth, the separate and equal station to which the laws of nature and of nature's God entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the separation.

...





File I/O





Demo Program: ToUpperCase.java

1. Token by token transcopy from source file to destination file.
2. File Input Scanner in = new Scanner(new File("input_file.name"));
3. File Output PrintWriter out = new PrintWriter(new File("out_file.name"));



Demonstration Program

SENTINELVALUE2.JAVA



Token Counting Loop: Bible Word Count

LECTURE 6



Lab: Word Count Project

Write a program to read in a text file, “bible.txt”, to count for the total words in this file.

Note:

- (1) Combine the techniques of while-loop for summation and text file input in previous demo program or lab project.
- (2) This version is a rough estimation because trimming down whitespace letters and punctuation marks is not required.



Expected Result:

(Sample Answer: [Bible.java](#))

Progress information reporting feature included in the sample answer program.

```
bible - Notepad
File Edit Format View Help
Genesis 1:1    In the beginning God created the heaven and the earth.
Genesis 1:2    And the earth was without form, and void; and darkness was upon the face of the deep. And
the Spirit of God moved upon the face of the waters.
Genesis 1:3    And God said, Let there be light: and there was light.
Genesis 1:4    And God saw the light, that it was good: and God divided the light from the darkness.
Genesis 1:5    And God called the light Day, and the darkness he called Night. And the evening and the
morning were the first day.
Genesis 1:6    And God said, Let there be a firmament in the midst of the waters, and let it divide the
waters from the waters.
Genesis 1:7    And God made the firmament, and divided the waters which were under the firmament from the
waters which were above the firmament: and it was so.
Genesis 1:8    And God called the firmament Heaven. And the evening and the morning were the second day.
Genesis 1:9    And God said, Let the waters under the heaven be gathered together unto one place, and let
the dry land appear: and it was so.
Genesis 1:10   And God called the dry land Earth; and the gathering together of the waters called he Seas:
and God saw that it was good.
```

```
Options
10% done !
20% done !
30% done !
40% done !
50% done !
60% done !
70% done !
80% done !
90% done !
Bible has 858195 words.
```



Menu Selection

LECTURE 7



Coffee Menu

Let's look at the Starbucks menu for Classic Favorites: (In this demo, we only work on **GRANDE** Size coffee)

| | |
|----------------------|------|
| Caffe Latte | 3.75 |
| Cappuccino | 3.65 |
| Caramel Macchiato | 3.95 |
| Caffe Mocha | 3.95 |
| Caffe Americano | 2.40 |
| Cinnamon Dolce Latte | 4.75 |
| Steamer | 2.50 |
| Drip Coffee | 1.95 |



VLOOKUP Explained

1 Scan down to find item

2 Look across to find price

3 Found it!

| CLASSIC FAVORITES | TALL | GRANDE | VENTI |
|----------------------|--------|--------|--------|
| Caffé Latte | \$2.95 | \$3.75 | \$4.15 |
| Cappuccino | \$2.95 | \$3.65 | \$4.15 |
| Caramel Macchiato | \$3.75 | \$3.95 | \$4.25 |
| Caffé Mocha | \$3.25 | \$3.95 | \$4.40 |
| Caffé Americano | \$2.00 | \$2.40 | \$2.75 |
| Cinnamon Dolce Latte | \$3.95 | \$4.75 | \$5.15 |
| Steamer | \$2.25 | \$2.50 | \$2.75 |
| Drip Coffee | \$1.75 | \$1.95 | \$2.05 |



Demo Program: CoffeeShop.java

Demonstrate a program for the coffee shop ordering system. First, display a menu for customer to pick one coffee order (for this version). If the customer has decided the coffee choice, then print out user's choice and enter user's payment amount. Otherwise, repeat the menu for customer to choose.

If the payment is greater than the coffee price, display the change the customer should receive (using Chapter 2's GetChange.java). Otherwise, ask customer for a different payment until payment is greater than price.



Demonstration Program

COFFEESHOP.JAVA

Results:



```
BlueJ: Terminal Window - Chapter05

Options

Welcome to Virtual Coffee Shop ...

CLASSIC FAVORITES      GRANDE
=====
[1] Caffe Latte        3.75
[2] Cappuccino         3.65
[3] Caramel Macchiato  3.95
[4] Caffe Macha        3.95
[5] Caffe Americano    2.40
[6] Cinnamon Dolce Latte 4.75
[7] Steamer           2.50
[8] Drip Coffee        1.95
[0] Need more time ...

Enter your choice: 0

Have you decided? y

Thank you for your business. Please come back soon.
```

```
BlueJ: Terminal Window - Chapter05

Options

Welcome to Virtual Coffee Shop ...

CLASSIC FAVORITES      GRANDE
=====
[1] Caffe Latte        3.75
[2] Cappuccino         3.65
[3] Caramel Macchiato  3.95
[4] Caffe Macha        3.95
[5] Caffe Americano    2.40
[6] Cinnamon Dolce Latte 4.75
[7] Steamer           2.50
[8] Drip Coffee        1.95
[0] Need more time ...

Enter your choice: 3

Have you decided? y

Your Choice: Caramel Macchiato Price: 3.95
Enter your payment: 20

Your change amount 16.05 consists of
    16 dollars
     0 quarters
     0 dimes
     1 nickels
     0 pennies

Thank you for your business. Please come back soon.
```




Input Validation Loop

LECTURE 8

```
public static void main(String[] args){
    System.out.print("\f");
    Scanner input = new Scanner(System.in);
    int x = 0;
    boolean done=false;
    do {
        try{
            System.out.print("Enter an Integer: ");
            x = Integer.parseInt(input.nextLine());
            done = true;
        }
        catch (Exception e){
            System.out.println("Wrong input format");
        }
    } while (!done);
    System.out.println("The number entered: "+x);
}
```

1. do-while loop
2. Try block to detect **InputMismatchException**



Demonstration Program

VALIDATION.JAVA



Lab: Lottery Checking Game

LECTURE 9



Lab Project: 4 digit number lottery

(Lottery.java, Sample Answer: LotteryAnswer.java)

Write a program to allow player to play three different lottery games. All of these three games are based on guess a lottery number of 4 digits (such as 1234, leading 0s are allowed, 0000 is OK).

In each play, computer generates a lottery number and player guesses another 4 digit number.



Lab Project: 4 digit number lottery

(Lottery.java, Sample Answer: LotteryAnswer.java)

Game 1: if all 4 numbers matched, player win \$50 prize.

Game 2: If player's guess number match first two digits, we call it front pair matched, win \$5.

If player's guess number match middle two digits, we call it middle pair matched, win \$5.

If player's guess number match ending two digits, we call it end pair matched, win \$5.

Player may win at most 4 matched pairs to win \$15.

Game 3: If player's guess number match each of the digits, we call it single digit matched, win \$2.

Player may win at most 4 matches to win \$8.



Lab Project: 4 digit number lottery

(Lottery.java, Sample Answer: LotteryAnswer.java)

Player can only enter one game but not 2 or 3 games. (He can choose a game of higher risk higher return or another game of lower risk lower return.)

Print out the prize the player won in this current game.

Then, ask if the player wants to play again?



When You are developing, you may create a debug mode to pre-watch the lottery number.

```
final boolean DEBUG_MODE = true; // turn on the debug  
mode to show lottery number
```

```
// Somewhre, after the lottery number has been generated.  
if (DEBUG_MODE) System.out.println("Lottery: " + lottery);
```

The odds for Game1 to win is only 1/1000. Therefore, it is hard to debug if you just try to guess.



Generation of Lottery Number

Use String, do not use int data type. (integer can not show leading 0s).

```
lottery = "";
```

```
lottery += (char) ((Math.random()*10)+'0'); // generate first digit
```

```
lottery += (char) ((Math.random()*10)+'0'); // generate second digit
```

```
lottery += (char) ((Math.random()*10)+'0'); // generate third digit
```

```
lottery += (char) ((Math.random()*10)+'0'); // generate fourth digit
```

```
// Up to this point, lottery has been created.
```



Rough Pseudo Code

Declare constants DEBUG_MODE;

Declare variables **done**, **play** as flags to control games

Declare sum for total won prize;

Declare lottery and guessed number strings.

Declare input stream from System.in

Declare game code



Rough Pseudo Code

```
do { 1. reset variables for the next game play.  
    2. generate a lottery number.  
       show lottery number if it is in DEBUG_MODE  
    3. Show menus for game choice and, then, ask player to enter a game code  
    4. Ask for player's guessing number  
    5. if (game == 1) check if he won the grand prize $50  
       if (game == 2) check if he won each of the pairs. Sum up the prize won in this game.  
       if (game == 3) check if he won each of the digits. Sum up the prize won in this game.  
    6. Show how much money the player won in current game.  
    7. ask if the player want to play a new game. set the flags (done, play) properly.  
} while (!done)
```



Prize checks

// All matched

```
if (guess.equals(lottery)) {sum += 50;  
    System.out.println("You won the all matched Grand prize $50 !!!");  
}
```

// Front pair matched check

```
if (guess.substring(0,2).equals(lottery.substring(0,2))) {  
    sum +=5;  
    System.out.println("You have front pair matched. Prize: $5.");  
}
```

// check for one digit

```
if (guess.charAt(0)== lottery.charAt(0))  
    {sum+=2; System.out.println("You matched the first number. Prize $2."); }
```



Expected Results. (LotteryAnswer.java)

```
BlueJ: Terminal Window - Chapter05
Options
Lottery: 0433
Welcome to Virtual Casino...
What type of game you want to play ($1/play)?
[1] All matched (All four numbers must match to win. ($50 return)
[2] All matched pairs (front pair MMXX, middle pair XMMX, and end pair XXMM, M: matched, X:missed) $5 return/pair
[3] All individual matched numbers $2/matched number
Enter the game you want to play: 1

Enter your lottery guess (4 digits 1234, leading 0s OK!): 0433

You won the all matched Grand prize $50 !!!
Total prize you have earned: 50

Do you still want to play(Y/N):
```



Expected Results.

```
BlueJ: Terminal Window - Chapter05
Options
Lottery: 8517
Welcome to Virtual Casino...
What type of game you want to play ($1/play)?
[1] All matched (All four numbers must match to win. ($50 return)
[2] All matched pairs (front pair MMXX, middle pair XMMX, and end pair XXMM, M: matched, X:missed) $5 return/pair
[3] All individual matched numbers $2/matched number
Enter the game you want to play: 2

Enter your lottery guess (4 digits 1234, leading 0s OK!): 8517

You have front pair matched. Prize: $5.
You have middle pair matched. Prize: $5.
You have end pair matched. Prize: $5.
Total prize you have earned: 15

Do you still want to play(Y/N):
```



Expected Results.

```
BlueJ: Terminal Window - Chapter05
Options
Lottery: 3113
Welcome to Virtual Casino...
What type of game you want to play ($1/play)?
[1] All matched (All four numbers must match to win. ($50 return)
[2] All matched pairs (front pair MMXX, middle pair XMMX, and end pair XXMM, M: matched, X:missed) $5 return/pair
[3] All individual matched numbers $2/matched number
Enter the game you want to play: 3

Enter your lottery guess (4 digits 1234, leading 0s OK!): 3113

You matched the first number. Prize $2.
You matched the second number. Prize $2.
You matched the third number. Prize $2.
You matched the fourth number. Prize $2.
Total prize you have earned: 8

Do you still want to play(Y/N):
```



for-loop

LECTURE 10



for loop

Syntax of for loop:

```
for (initial_condition; continuation-condition; action-after-each-iteration)
{
```

```
    // loop body:
```

```
    Statement(s);
```

```
}
```

Declaring and Initializing
loop control variable

Checking
condition

Incrementing loop
control variable

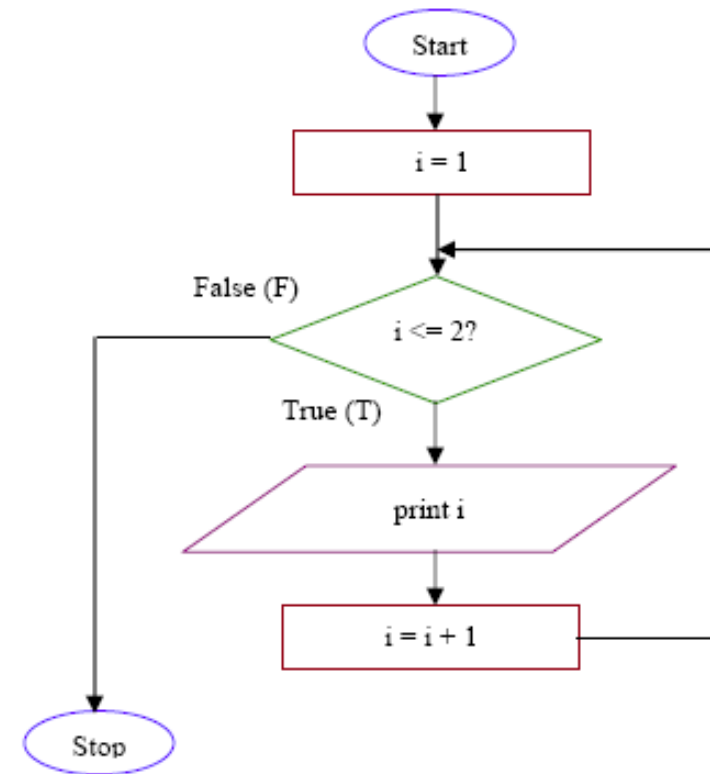
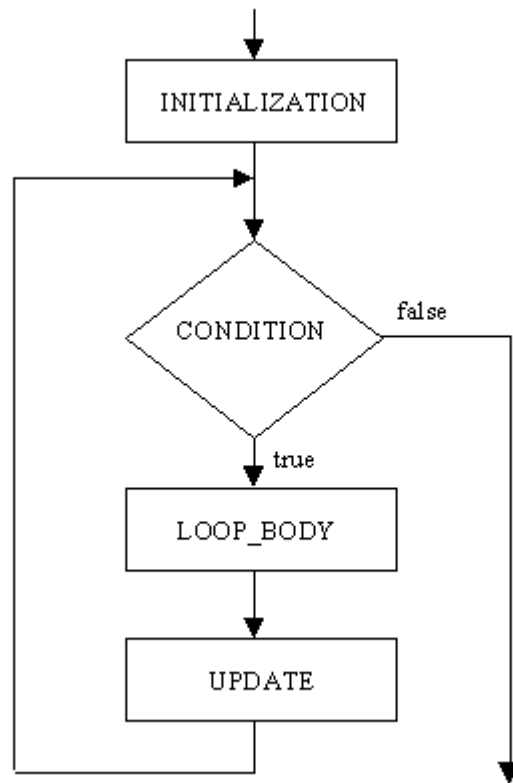
```
for (int i =0; i<10 ; i++) {
```

```
    // Loop statements to be executed
```

```
}
```



Flowchart for for-loop





Counting Loop

LECTURE 11



Trace for Loop

```
int i;
```

Declare i

```
for (i = 0; i < 2; i++) {  
    System.out.println(  
        "Welcome to Java!");  
}
```



Trace for Loop

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println(  
        "Welcome to Java!");  
}
```

Execute initializer
i is now 0



Trace for Loop

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println( "Welcome to Java!");  
}
```

(i < 2) is true
since i is 0



Trace for Loop

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

Print Welcome to Java



Trace for Loop

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

Execute adjustment statement
i now is 1



Trace for Loop

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

(i < 2) is still true
since i is 1



Trace for Loop

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

Print Welcome to Java



Trace for Loop

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

Execute adjustment statement
i now is 2



Trace for Loop

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

(i < 2) is false
since i is 2



Trace for Loop

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java");  
}
```

Exit the loop. Execute the next statement after the loop



Note

The initial-action in a for loop can be a list of zero or more comma-separated expressions. The action-after-each-iteration in a for loop can be a list of zero or more comma-separated statements. Therefore, the following two for loops are correct. They are rarely used in practice, however.

```
for (int i = 1; i < 100; System.out.println(i++));
```

```
for (int i = 0, j = 0; (i + j < 10); i++, j++) {  
    // Do something  
}
```



Note

If the loop-continuation-condition in a for loop is omitted, it is implicitly true. Thus the statement given below in (a), which is an infinite loop, is correct. Nevertheless, it is better to use the equivalent loop in (b) to avoid confusion:

```
for ( ; ; ) {  
    // Do something  
}
```

(a)

Equivalent



```
while (true) {  
    // Do something  
}
```

(b)



Caution

Adding a semicolon at the end of the for clause before the loop body is a common mistake, as shown below:

```
for (int i=0; i<10; i++);  
{  
    System.out.println("i is " + i);  
}
```

Logic Error



Caution, cont.

Similarly, the following loop is also wrong:

```
int i=0;
while (i < 10); ← Logic Error
{
    System.out.println("i is " + i);
    i++;
}
```

In the case of the do loop, the following semicolon is needed to end the loop.

```
int i=0;
do {
    System.out.println("i is " + i);
    i++;
} while (i<10); ← Correct
```



Indexed Loop

LECTURE 12



Indexed Loop

- Each iteration has a number as index.
- Each index access one element in an array.



Demonstration Program

INDEXEDLOOP.JAVA

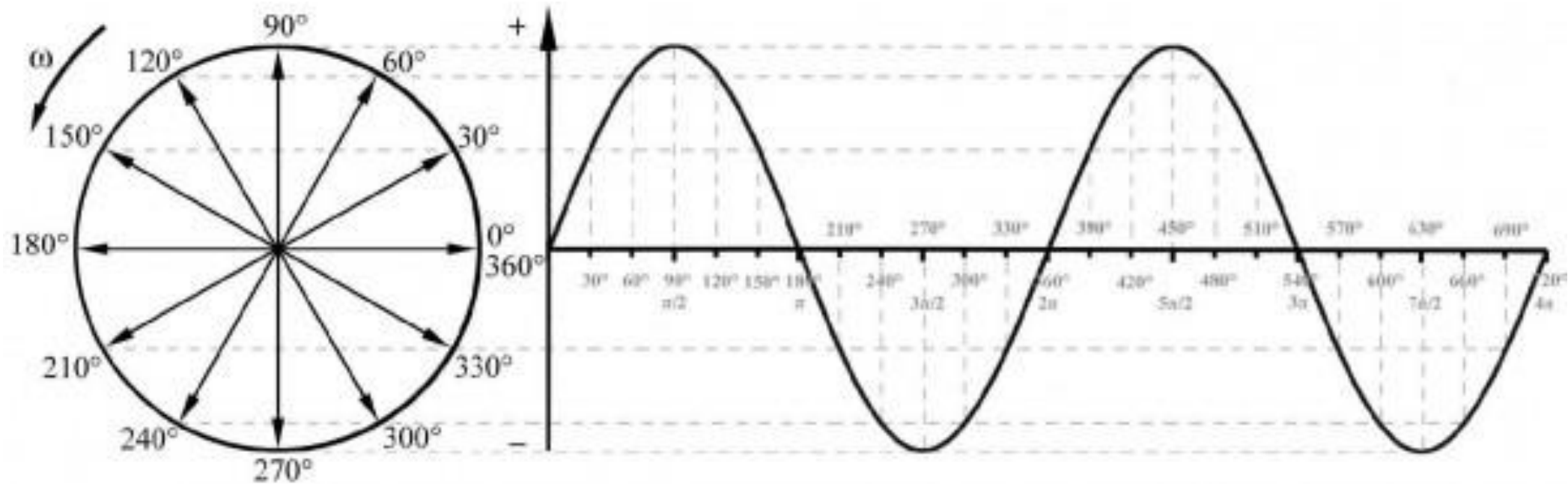


Lab: Sine and Cosine Table of a Unit Circle

LECTURE 13



Unit Circle (Step_Size = 30 degrees)



Step_Size (in radians) = $2\pi / \text{STEPS}$, STEPS = 12 in this case.



Lab: Sine and Cosine Table for Unit Circle

Write a program to generate a table for Sine and Cosine functions over a circle (1 cycle). In the program, please define a constant, STEPS, for the number of angles in the table (Totally, there should be STEPS+1 points, if both 0 and 2π are included.)

Generate the table with proper header and description.

Hint: use for-loop

```
for (int i = 0; i<=STEPS; i++){  
    // Generate sine and cosine of an angle here.  
}
```



Technique to Prevent Accumulation of Round off Errors

```
// Round off double data type to 5 digits  
final int RESOLUTION = 100000;  
double theta = Math.PI * 2.0 / STEPS; // may have round off errors  
sine = Math.sin(theta*i);  
sine = (int)(Math.sin(theta*i)*RESOLUTION)/(double) RESOLUTION ;  
sine = Math.round(Math.sin(theta*i)*RESOLUTION)/(double) RESOLUTION ;  
// shift the sine value left for 5 digits and,  
// then round off errors by quantization  
// and shift right 5 digits in double type
```


Sine/Cosine Table

Sine Cosine

No round off
with println

```

=====
( 0/ 24)*2*PI: 0.0 1.0
( 1/ 24)*2*PI: 0.25881904510252074 0.9659258262890683
( 2/ 24)*2*PI: 0.49999999999999994 0.8660254037844387
( 3/ 24)*2*PI: 0.7071067811865475 0.7071067811865476
( 4/ 24)*2*PI: 0.8660254037844386 0.50000000000000001
( 5/ 24)*2*PI: 0.9659258262890682 0.25881904510252096
( 6/ 24)*2*PI: 1.0 6.123233995736766E-17
( 7/ 24)*2*PI: 0.9659258262890683 -0.25881904510252063
( 8/ 24)*2*PI: 0.8660254037844387 -0.49999999999999998
( 9/ 24)*2*PI: 0.7071067811865476 -0.7071067811865475
(10/ 24)*2*PI: 0.50000000000000003 -0.8660254037844385
(11/ 24)*2*PI: 0.258819045102521 -0.9659258262890682
(12/ 24)*2*PI: 1.2246467991473532E-16 -1.0
(13/ 24)*2*PI: -0.25881904510252035 -0.9659258262890684
(14/ 24)*2*PI: -0.49999999999999997 -0.8660254037844388
(15/ 24)*2*PI: -0.7071067811865471 -0.7071067811865479
(16/ 24)*2*PI: -0.8660254037844385 -0.50000000000000004
(17/ 24)*2*PI: -0.9659258262890681 -0.2588190451025215
(18/ 24)*2*PI: -1.0 -1.8369701987210297E-16
(19/ 24)*2*PI: -0.9659258262890684 0.2588190451025203
(20/ 24)*2*PI: -0.866025403784439 0.49999999999999993
(21/ 24)*2*PI: -0.7071067811865477 0.7071067811865474
(22/ 24)*2*PI: -0.50000000000000004 0.8660254037844384
(23/ 24)*2*PI: -0.25881904510252157 0.9659258262890681
(24/ 24)*2*PI: -2.4492935982947064E-16 1.0

```

Sine/Cosine Table

Sine Cosine

```

=====
( 0/ 24)*2*PI: 0.00000 1.00000
( 1/ 24)*2*PI: 0.25882 0.96593
( 2/ 24)*2*PI: 0.50000 0.86603
( 3/ 24)*2*PI: 0.70711 0.70711
( 4/ 24)*2*PI: 0.86603 0.50000
( 5/ 24)*2*PI: 0.96593 0.25882
( 6/ 24)*2*PI: 1.00000 0.00000
( 7/ 24)*2*PI: 0.96593 -0.25882
( 8/ 24)*2*PI: 0.86603 -0.50000
( 9/ 24)*2*PI: 0.70711 -0.70711
(10/ 24)*2*PI: 0.50000 -0.86603
(11/ 24)*2*PI: 0.25882 -0.96593
(12/ 24)*2*PI: 0.00000 -1.00000
(13/ 24)*2*PI: -0.25882 -0.96593
(14/ 24)*2*PI: -0.50000 -0.86603
(15/ 24)*2*PI: -0.70711 -0.70711
(16/ 24)*2*PI: -0.86603 -0.50000
(17/ 24)*2*PI: -0.96593 -0.25882
(18/ 24)*2*PI: -1.00000 -0.00000
(19/ 24)*2*PI: -0.96593 0.25882
(20/ 24)*2*PI: -0.86603 0.50000
(21/ 24)*2*PI: -0.70711 0.70711
(22/ 24)*2*PI: -0.50000 0.86603
(23/ 24)*2*PI: -0.25882 0.96593
(24/ 24)*2*PI: -0.00000 1.00000

```

No round off
With printfExpected
Result:
(Sine.java)

Math.round()

BlueJ: Termin



Expected
Result:
(Sine.java)

| Options | | |
|-------------------|----------|----------|
| int | | |
| Sine/Cosine Table | | |
| | Sine | Cosine |
| ===== | | |
| (0/ 24)*2*PI: | 0.00000 | 1.00000 |
| (1/ 24)*2*PI: | 0.25881 | 0.96592 |
| (2/ 24)*2*PI: | 0.49999 | 0.86602 |
| (3/ 24)*2*PI: | 0.70710 | 0.70710 |
| (4/ 24)*2*PI: | 0.86602 | 0.50000 |
| (5/ 24)*2*PI: | 0.96592 | 0.25881 |
| (6/ 24)*2*PI: | 1.00000 | 0.00000 |
| (7/ 24)*2*PI: | 0.96592 | -0.25881 |
| (8/ 24)*2*PI: | 0.86602 | -0.49999 |
| (9/ 24)*2*PI: | 0.70710 | -0.70710 |
| (10/ 24)*2*PI: | 0.50000 | -0.86602 |
| (11/ 24)*2*PI: | 0.25881 | -0.96592 |
| (12/ 24)*2*PI: | 0.00000 | -1.00000 |
| (13/ 24)*2*PI: | -0.25881 | -0.96592 |
| (14/ 24)*2*PI: | -0.49999 | -0.86602 |
| (15/ 24)*2*PI: | -0.70710 | -0.70710 |
| (16/ 24)*2*PI: | -0.86602 | -0.50000 |
| (17/ 24)*2*PI: | -0.96592 | -0.25881 |
| (18/ 24)*2*PI: | -1.00000 | 0.00000 |
| (19/ 24)*2*PI: | -0.96592 | 0.25881 |
| (20/ 24)*2*PI: | -0.86602 | 0.49999 |
| (21/ 24)*2*PI: | -0.70710 | 0.70710 |
| (22/ 24)*2*PI: | -0.50000 | 0.86602 |
| (23/ 24)*2*PI: | -0.25881 | 0.96592 |
| (24/ 24)*2*PI: | 0.00000 | 1.00000 |

| Options | | |
|-------------------|----------|----------|
| Sine/Cosine Table | | |
| | Sine | Cosine |
| ===== | | |
| (0/ 24)*2*PI: | 0.00000 | 1.00000 |
| (1/ 24)*2*PI: | 0.25882 | 0.96593 |
| (2/ 24)*2*PI: | 0.50000 | 0.86603 |
| (3/ 24)*2*PI: | 0.70711 | 0.70711 |
| (4/ 24)*2*PI: | 0.86603 | 0.50000 |
| (5/ 24)*2*PI: | 0.96593 | 0.25882 |
| (6/ 24)*2*PI: | 1.00000 | 0.00000 |
| (7/ 24)*2*PI: | 0.96593 | -0.25882 |
| (8/ 24)*2*PI: | 0.86603 | -0.50000 |
| (9/ 24)*2*PI: | 0.70711 | -0.70711 |
| (10/ 24)*2*PI: | 0.50000 | -0.86603 |
| (11/ 24)*2*PI: | 0.25882 | -0.96593 |
| (12/ 24)*2*PI: | 0.00000 | -1.00000 |
| (13/ 24)*2*PI: | -0.25882 | -0.96593 |
| (14/ 24)*2*PI: | -0.50000 | -0.86603 |
| (15/ 24)*2*PI: | -0.70711 | -0.70711 |
| (16/ 24)*2*PI: | -0.86603 | -0.50000 |
| (17/ 24)*2*PI: | -0.96593 | -0.25882 |
| (18/ 24)*2*PI: | -1.00000 | 0.00000 |
| (19/ 24)*2*PI: | -0.96593 | 0.25882 |
| (20/ 24)*2*PI: | -0.86603 | 0.50000 |
| (21/ 24)*2*PI: | -0.70711 | 0.70711 |
| (22/ 24)*2*PI: | -0.50000 | 0.86603 |
| (23/ 24)*2*PI: | -0.25882 | 0.96593 |
| (24/ 24)*2*PI: | 0.00000 | 1.00000 |



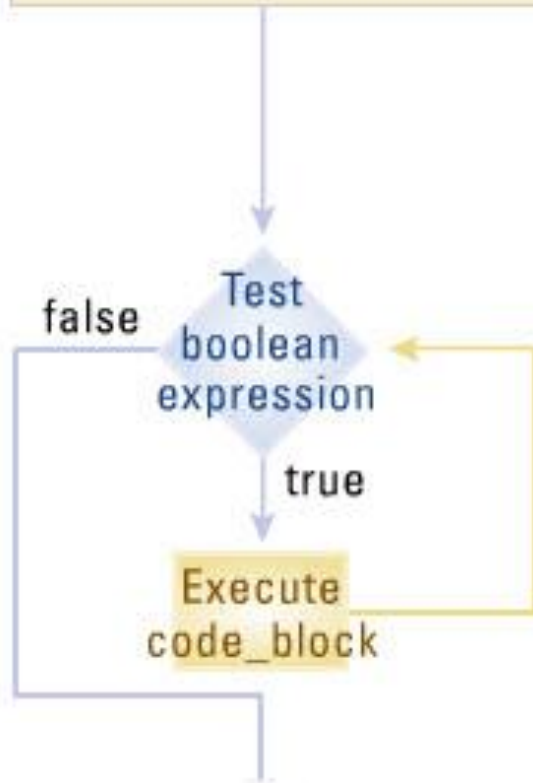
Summary

LECTURE 14



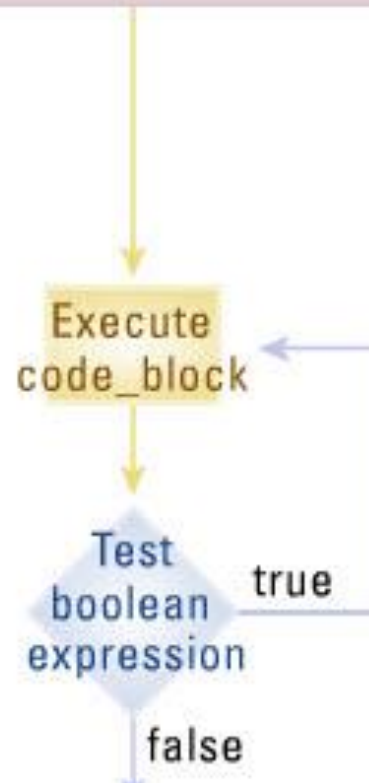
while

- Zero or more iteration
- When total iterations unknown



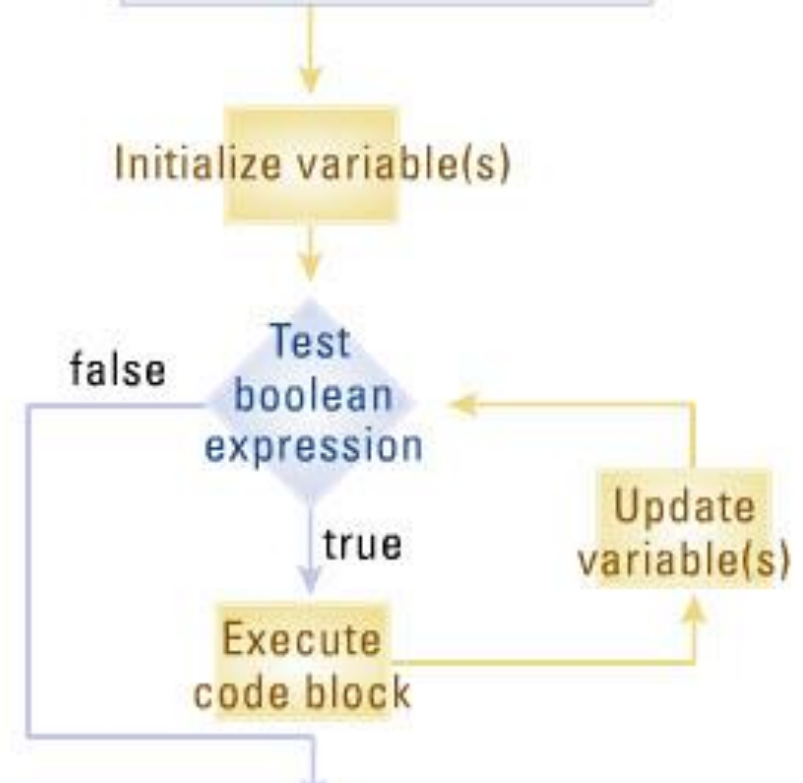
do while

- At least one iteration
- When total iterations unknown



for

- Any number of iteration
- When total iterations known





LOOP Structures Supported By Java

Loops:

- for-loop (later lecture)
- while-loop
- do-while-loop
- for-each-loop (chapter 7/8)

Loop Breaks: (later in other lecture)

- `{}` `/* empty braces as pass function */`
- `Continue` `/* skip the rest of iteration */`
- `Break` `/* skip the rest of loop */`
- `Return` `/* skip the rest of function */`
- `System.exit(0);` `/* skip the rest of program */`