Lecture Notes
Chapter #3
Selections

1. Flow Charts

2. Boolean Data Types & Operations

relational (comparison) operators

< <= > >= == !=

int a = 2;
int b = 3;
a < b ➔ TRUE
a == b ➔ FALSE

'a' > 'A' ➔ TRUE

boolean values

TRUE, FALSE

boolean variable

boolean doorOpen = FALSE;

boolean (logical) operators

!	&&	||	^
not	and	inclusive or	exclusive or

```
int number = input.nextInt( );
System.out.println("Is " + number
+ "\n\tdivisible by 2 and 3? " + (number %2 == 0 && number %3 == 0)
+ "\n\tdivisible by 2 or 3? " + (number %2 == 0 || number %3 == 0)
+ "\n\tdivisible by 2 or 3, but not both? "
+ (number %2 == 0 ^ number %3 == 0)
```

mathematics ←→ A < B < C

Java ←→ ((A < B) && (B < C))

boolean values cannot be cast to other types

values of other types cannot be cast to a boolean type

DeMorgan's Law

!(a && b) ←→ !a || !b

!(a || b) ←→ !a && !b

3. **Evaluation of Boolean Expressions**
   - **(a && b) e.g., ((x == y) && (u < v))**
     - **evaluate a, if a is TRUE, evaluate b. if b is FALSE then done**
     - **conditional (short-circuit) AND operator**

   - **(a || b) e.g., ((x == y) || (u < v))**
     - **evaluate a, if a is FALSE, evaluate b. if a is TRUE then done**
     - **conditional (short-circuit) OR operator**

4. **Leap Year Algorithm**

   (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 ==0))

   Leap years are those years that are either divisible by 4 but not by 100 or else are divisible by 400.

```
boolean isLeapYear;
isLeapYear = (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 ==0));
```

5. **Simple Math Program**

```
int answer;
int num1 = (int)(System.currentTimeMillis( ) % 10);
int num1 = (int)(System.currentTimeMillis( ) * 7 % 10);
System.out.print( num1 + " + " + num2 + " == " );
answer = input.nextInt( );
System.out.println("\n" + num1 + " + " + num2 + " == " + answer + " is " +
    (num1 + num2 == answer) );
```

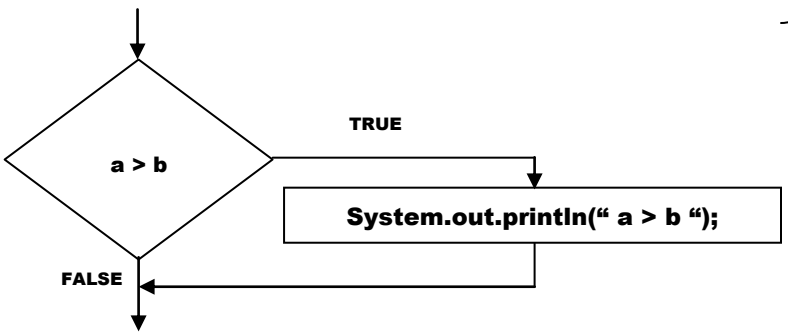
generate random numbers

6. **Selections**

   a. **if (boolean expression) {action }**

```
if ( a > b )
{
   System.out.println(" a > b ");
}
```

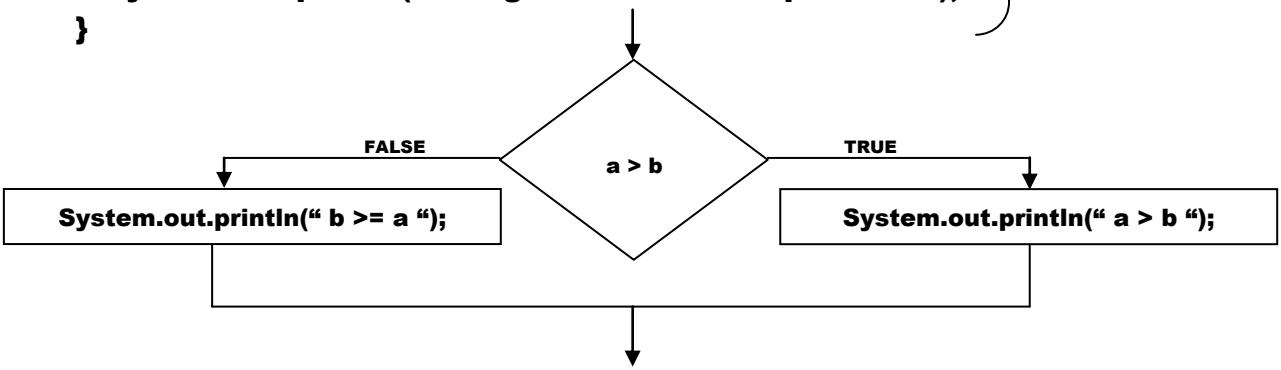
"if then" statement



b. if (boolean expression) { action1 } else { action2 }

```
if ( a > b )
{
   System.out.println(" a is greater than b ");
}
else
{
   System.out.println(" b is greater than or equal to a ");
}
```

"if then, else" statement



c. Nested If Statements

```
if (boolean expression1 )
{
   if (boolean expression2 )
   { action1 }
   else
   { action2 }
}
else { action3 }
```

```
if (boolean expression1 )
{
   if (boolean expression2 )
   { action1 }
   else
   { action2 }
}
else
{
   if (boolean expression3 )
   { action3 }
   else
   { action4 }
 }
```

```
if (score >= 90.0 )
   grade = 'A';
else if (score >= 80.0 )
   grade = 'B';
 else if (score >= 70.0 )
   grade = 'C';
else if (score >= 60.0 )
   grade = 'D';
else
   grade = 'F';
```

An <u>else clause</u> matches the most recent unmatched <u>if clause</u> in the same block

```
boolean even = number %2 == 0;
if (even)
System.out.println("Even");
```

7. **Generation of Random Numbers**
   - **int num1 = (int)(System.currentTimeMillis( ) % 10);**
   - **int num2 = (int)(Math.random( ) * 10);**

   double d;
   d = Math.random( );
   Math.random returns a double value such that
   0.0 ≤ d < 1.0

   int num2 = (int)(Math.random( ) * 10);
   returns an integer number 0 ≤ num2 ≤ 9

8. **Lottery**

```
int lottery = (int)(Math.random( ) * 100);        // two digit lottery number
int guess = input.nextInt( );

if (guess == lottery)
    System.out.println("Exact match you win $10,000");
else if (guess % 10 == lottery / 10  &&  guess / 10 == lottery % 10)
    System.out.println("Match all digits you win $3,000");
else if (guess % 10 == lottery / 10  ||  guess % 10 == lottery % 10) ||
          guess / 10 == lottery / 10  ||  guess / 10 == lottery % 10)
    System.out.println("Match one digit you win $1,000");
else
    System.out.println("You lose!");
```

9. **Page 81-82 §3.3.7 Computing 2002 US Federal Personal Tax Rates**
   Specification and partial code – subject to homework assignment

```
System.exit(0);      // graceful exit from program under error conditions

double tax = 0;
        // avoid potential syntax error if no value is assigned prior to use
```

10. **Testing programs**
    Provide test cases that cover all potential cases; make sure that you cover all normal operating conditions but also consider any unusual or abnormal cases in the test data set

11. Incremental Development & Testing

Write the code in small sections and test it before adding more code
Do not expect to be able to efficiently debug large sections of code!

12. Switch Statements

```
switch (expression)
{
    case value1: { statement1.1, …,  statement1.m; break;}
    case value2: { statement2.1, …,  statement2.n; break;}
    case value3: { statement3.1, …,  statement3.p; break;}
    case value4: { statement4.1, …,  statement4.q; break;}
                …
    case valuek: { statementk.1, …,  statementk.r; break;}
    default: … ; { default statements; break;}
}
```

- expression value types, e.g., value2, etc.
  char, byte, short, int

- expressions must evaluate to constants, i.e., value2 cannot contain variables
- if a break statement is omitted, computation will continue into the next case statement

13. Conditional Expressions

```
if (x > 0)
y = 1;
else
y = -1;

y = (x > 0) ? 1 : -1;

y = ( booleanExpression ) ? expressionIfTrue : expressionIfFalse;


given y = 10 and n = 5
y = ( a > b || c > b ) ? (k * k + n) : (k +k – n) ;
evaluates to
y == 105 if ( a > b || c > b ) evaluates to TRUE
or
 y == 15 if ( a > b || c > b ) evaluates to FALSE
```

14. Formatting Console Output pg 89-90

System.out.printf(format, item1, item2, …, itemk);

format specifiers

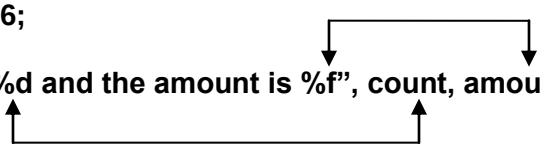
%b	**boolean**
%c	**character**
%d	**integer – base 10 decimal integer**
%f	**floating point – base 10 real number**
%e	**scientific notation – base 10 real number**
%s	**string**

Given

int count = 5;
double amount = 45.56;

then

System.out.printf("count is %d and the amount is %f", count, amount);



produces the following output

count is 5 and the amount is 45.560000

<u>Specifying Width and Precision</u>

%5c	**c='A'**	**➔ _ _ _ _ A**
%6b	**b=TRUE**	**➔ _ _TRUE**
	b=FALSE	**➔ _FALSE**
%5d	**d=15**	**➔ _ _ _ 15**
	d=12345678	**➔ 12345678**
%10.2f	**f=123.45678**	**➔ _ _ _ _123.45**
	f=12345678.12345678	**➔ 12345678.12**
%10.2e	**e=12345678.12345678**	**➔ 1.234567812e+8**
%12s	**s="Welcome"**	**➔ _ _ _ _ _ Welcome**
	s="Welcome to Comp 110"	**➔ Welcome to Comp 110**

15. Operator Precedence and Associativity pg 91

var++, var- -	**i.e., postfix**
+, - (unary plus, unary minus),	**++var, - -var i.e., prefix**
(type)	**i.e., casting**
!(…)	**i.e., not**
***, /, %**	**i.e., multiply, divide, remainder**
+, -	**i.e., addition, subtraction**
<, <=, >, >=	**i.e., comparison operators**
==, !=	**i.e., equality tests**
^	**i.e., exclusive OR, i.e, XOR**
&&	**i.e., AND**
||	**i.e., OR**
=, +=, -=, *=, /=, %=	**i.e., assignment operator**

Assignment operators are RIGHT ASSOCIATIVE a = b += c = 5 ➔ a = (b += (c = 5))
Other binary operators are LEFT ASSOCIATIVE a – b + c - d ➔ ((a - b) + c) - d

7. GUI Confirmation Dialogs pg 92