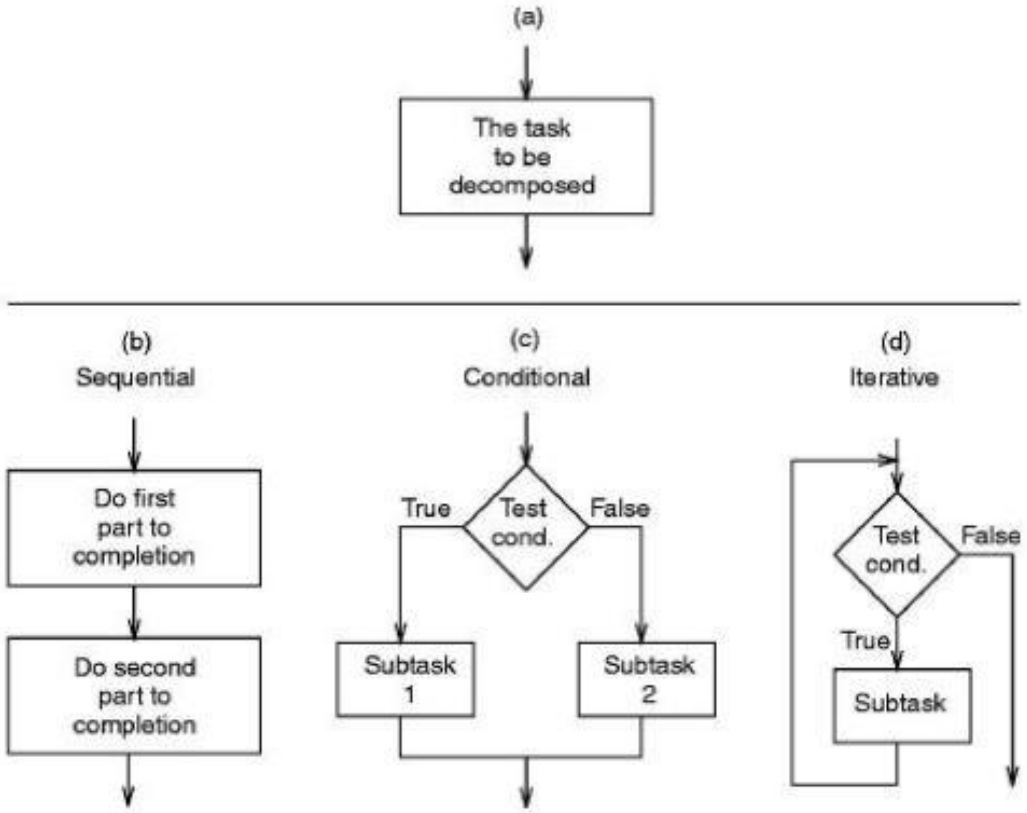
WEEK THREE

Acknowledgements: Slides created based off material provided by Dr. Travis Doom

CONTROL STRUCTURES

- Sequential
  - Default
  - Do A -> B -> C -> …
- Selective/Conditional
  - Decision/choice
  - Do A if some condition, otherwise do B
- Iteration
  - Loops
  - Do A repeatedly until a condition is met



RELATIONAL OPERATORS

Relational Operator	Meaning
>	is greater than
<	is less than
>=	is greater than or equal to
<=	is less than or equal to
==	is equal to
!=	is not equal to

CONDITIONAL CONTROL: IF STATEMENT

```
int age = 21;

if (age < 16)
{
        System.out.println("You can't drive!");
}
```

conditional statement

Code inside curly braces only executes if the conditional statement evaluates to true otherwise it is skipped

= if statement syntax

CONDITIONAL CONTROL: ELSE STATEMENT

```
if (age < 16)
{
        System.out.println("You can't drive!");
}
else
{
        System.out.println("You can drive!");
}
```

Executes if conditional evaluates to true

catch-all

must be preceeded by an if statement (or an else if)

always alone (no conditional statement)

Executes if none of the previous conditional statements evaluate to true

CONDITIONAL CONTROL: ELSE IF STATEMENT

```
if (age < 16) {
    System.out.println("You can't drive!");
}
else if (hasLicense == false) {
    System.out.println("You can't drive!);
}
else {
    System.out.println("You can drive!");
}
```

if the conditional evaluates to true / false

if the conditional evaluates to true / false

else if:

- allows for more than two options
- must be preceeded by an if statement
- must have its own conditional statement

NESTED IF STATEMENTS

```
if (age < 16) {
    if (hasPermit == true) {
        System.out.println("You can drive!");
    }
    else {
        System.out.println("You can't drive!");
    }
}
else if (hasLicense == false) {
    System.out.println("You can't drive!);
}
else {
    System.out.println("You can drive!");
}
```

- Nested if statements allow us to do a check only if some prior condition was first met.

LOGICAL OPERATORS

Operator	Meaning	Effect
&&	AND	Connects two boolean expressions. Both expressions must be true for the overall expression to be true.
||	OR	Connects two boolean expressions. Either one or both expressions must be true for the overall expression to be true.
!	NOT	The ! operator reverses the truth of a boolean expression.

IN CLASS ACTIVITY

1. Prompt the user for their grade on their first assignment (0-100)
2. Prompt the user for their grade on their second assignment (0-100)
3. Prompt the user for their grade on their third assignment (0-100)
4. Print to the user their letter grade for each assignment
   - A (100-90), B (89-80), C (79-70), D (69-60), F (59-0)
5. Print out the average of those 3 grades to two decimal places

LOGICAL OPERATORS CONTINUED

A	B	!A	!B	A && B	A || B	!A && !B	!(A && B)
false	false	true	true	false	false	true	true
false	true	true	false	false	true	false	true
true	false	false	true	false	true	false	true
true	true	false	false	true	true	false	false

ORDER OF PRECEDENCE

Order of Precedence	Operators	Description
1	!	Unary negation, logical NOT
2	* / %	Multiplication, Division, Modulus
3	+ -	Addition, Subtraction
4	< > <= >=	Less-than, Greater-than, Less-than or equal to, Greater-than or equal to
5	== !=	Is equal to, Is not equal to
6	&&	Logical AND
7	||	Logical OR
8	=	Assignment operator.

LOGICAL OPERATOR EXAMPLES

```
int x = 15; int y = 5; int z = 6;
```

- `if ((x > y) && (y > z))` false
- `if (x > y && y > z)` false
- `if ((x > y) || (y > z))` true
- `if !(x == 4)` true
- `if !(x < y)` true
- `if (!(!(x < y)))` false

SHORT CIRCUITING

- Compiler will not necessarily perform both conditional checks for AND and OR operations
  - AND: if the first condition evaluates to false, the compiler will not check the second condition
  - OR: if the first condition evaluates to true, the compiler will not check the second condition

```
boolean this = false;

boolean that = true;

if (this && that)
```

since "this" = false, compiler won't check "that"

```
if (that || this)
```

since "that" = true, compiler won't check "this"

OTHER NOTES

- Don’t mix up the assignment operator (=) with the equality operator (==)
  - Using = instead of == will result in an incompatible type error
  - required: boolean, found int
- Strings *cannot* be compared with the equality operator (==)
  - String is not a primitive type
  - You must use a String method called .equals() or .equalsIgnoreCase()
  - Ex:
```
String str = “ABC”;
Boolean result = str.equals(“abc”);
```
- Make sure to use && and || instead of & and |
  - &: bitwise AND
  - |: bitwise OR
- boolean values can be very useful with conditional statements
  - Can store the result of a conditional
  - Can represent a current state

THE SWITCH

- Switch conditional must evaluate to a non-floating point (decimal) primitive
  - Usually char, or int
  - Exceptions include String and some wrapper classes
- Each case has a literal value
- Break keyword
  - Exits the case statement
  - If it's removed, the compiler will continue to execute the code in the next case
- Default case is a catch-all for input that doesn't match any other case

```
char cmd;
switch (cmd) {
        case 'o':
                System.out.println("open");
                break;
        case 'c':
                System.out.println("close");
                break;
        default:
                System.out.println("bye");
                break;
}
```

TERNARY OPERATOR

- Used to replace if/else conditional
- Syntax:

```
datatype variableName = conditional ? value if true : value if false
```

- Ex: `String temperature = (degrees > 60) ? “warm” : “cool”;`
- For clear style, avoid use of this operator

IN CLASS ACTIVITY: CALCULATOR

- Read in a simple (2 operand) addition or subtraction problem from the user
- If either of the operands is negative
  - Print that you can't perform the operation
- If either of the operands is more than two digits
  - Print that you can't perform the operation
- Otherwise perform the operation the user typed in
  - Print the result to the user

IN CLASS ACTIVITY

- Take in an integer and a character from the user
- If integer is greater than 100 and the character is a vowel
  - Print to the user “yeehaw”
- If the integer is negative and the character is an o
  - Print to the user “-o”
- If the character is an “y”
  - Print to the user “why”

IN CLASS CODE (SECTION 03)

1. Prompt the user to enter an even number
   - Check if the number is even, if it is, add a point to their score
2. Prompt the user to enter a decimal number between 0.5 and 5.5
   - If the number is in the range, add a point
3. Prompt the user for a word with the word cat in it
   - The word must be longer than cat
   - The word must contain cat
   - If the word meets these conditions, add a point
4. Prompt the user for a word as long as the first number they entered
   - If the word is the same length as their even number from step 1, add a point
5. Print a random number to the user and ask them to repeat it
   - If the number is the same, add a point
6. Print the score to the user
7. Ask the user to say "bye"
   - If they type "bye" (with any capitalization), print "Goodbye!"

IN CLASS CODE (SECTION 04)

- Create a number guessing game

1. Generate a random number
2. Ask if 2 or 3 players are going to play
   - If the user enters any number besides 2 or 3, notify them and then end the program
3. Prompt each player for a number
4. Indicate which player was closer
5. If there is a tie, generate another number and ask again
   - If there is another tie, just print “It’s a tie!”

IN CLASS CODE (SECTION 02)

- Higher or lower game

1. Generate a random integer (default from 1-10 inclusive)
   - Allow user to select range
2. Have the user guess the number
   - If their guess is less than the correct number, print “higher”
   - If their guess is greater than the correct number, print “lower”
3. Have the user guess two more times with the same print rules
4. If the user guesses the number correctly, end the game and print their score
5. Score will be based on how many attempts they had
   - If they guess it right the first time: 3 pts, second time: 2 pts, third time: 1 pt, otherwise 0 pt
6. Modify game to work with characters

IN CLASS CODE (SECTION 01)

- Higher or lower game

1. Generate a random character
2. Have the user guess the character
   - If their guess is less than the correct character, indicate which way they need to guess
   - If their guess is greater than the correct character, indicate which way they need to guess
3. Have the user guess two more times with the same print rules
4. If the user guesses the character correctly, end the game and print their score
5. Score will be based on how many attempts they had
   - If they guess it right the first time: 3 pts, second time: 2 pts, third time: 1 pt, otherwise 0 pt

From the ASCII table…

Symbol	Decimal	Binary
A	65	01000001
B	66	01000010
C	67	01000011
D	68	01000100
E	69	01000101
F	70	01000110
G	71	01000111
H	72	01001000
I	73	01001001
J	74	01001010
K	75	01001011
L	76	01001100
M	77	01001101
N	78	01001110
O	79	01001111
P	80	01010000
Q	81	01010001
R	82	01010010
S	83	01010011
T	84	01010100
U	85	01010101
V	86	01010110
W	87	01010111
X	88	01011000
Y	89	01011001
Z	90	01011010

Symbol	Decimal	Binary
a	97	01100001
b	98	01100010
c	99	01100011
d	100	01100100
e	101	01100101
f	102	01100110
g	103	01100111
h	104	01101000
i	105	01101001
j	106	01101010
k	107	01101011
l	108	01101100
m	109	01101101
n	110	01101110
o	111	01101111
p	112	01110000
q	113	01110001
r	114	01110010
s	115	01110011
t	116	01110100
u	117	01110101
v	118	01110110
w	119	01110111
x	120	01111000
y	121	01111001
z	122	01111010