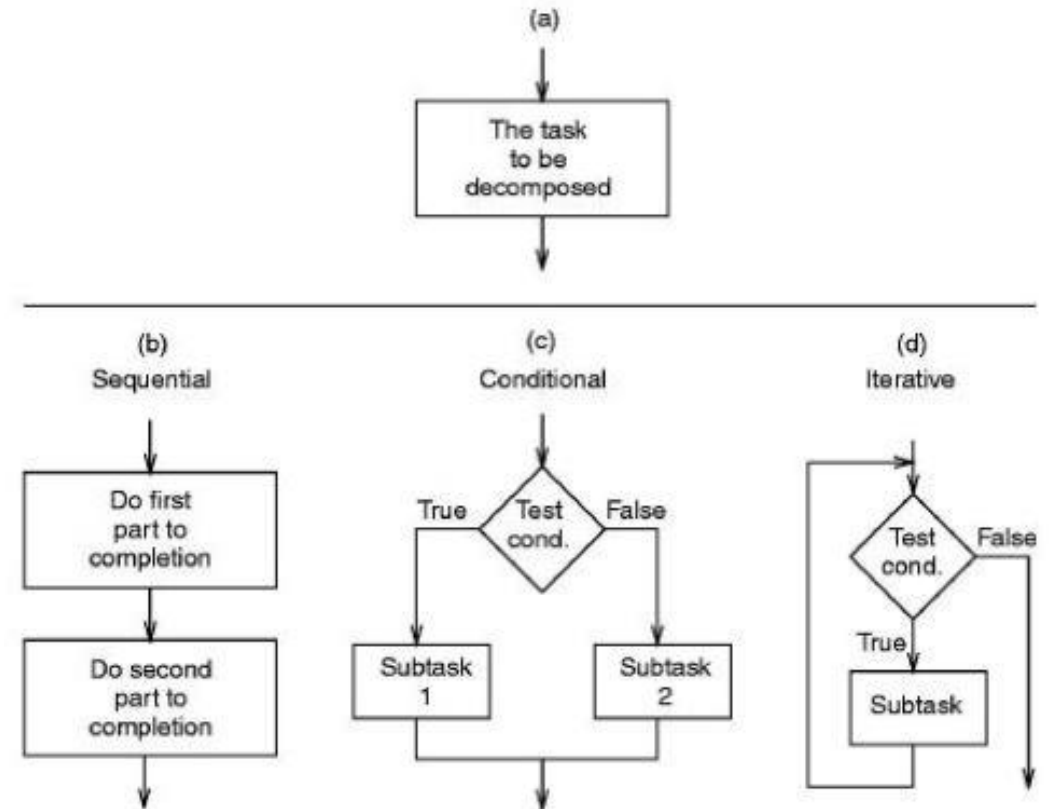

WEEK THREE

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) conditional statement
```

```
{
```

```
    System.out.println("You can't drive!");
```

```
}
```

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

 = if statement syntax

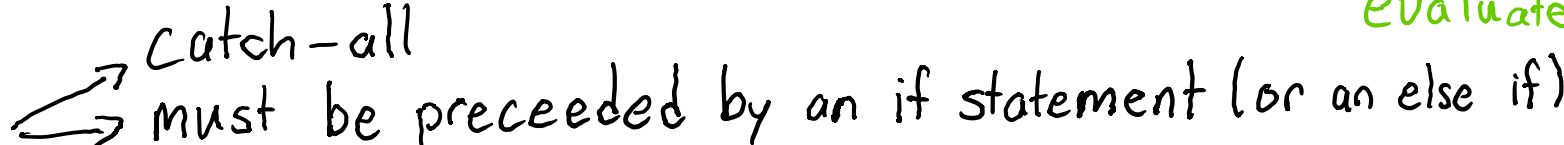
CONDITIONAL CONTROL: ELSE STATEMENT

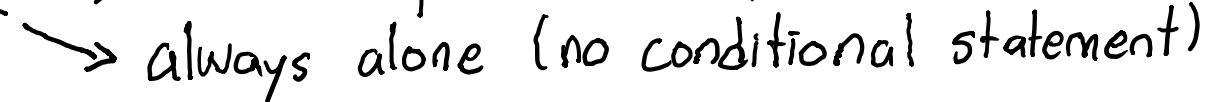
```
if (age < 16)
```

```
{
```

```
    System.out.println("You can't drive!"); Executes if conditional  
evaluates to true
```

```
}
```

else  Catch-all
must be preceded by an if statement (or an else if)

 always alone (no conditional statement)

```
{
```

```
    System.out.println("You can drive!"); Executes if none of the previous  
conditional statements evaluate  
to true
```

```
}
```

CONDITIONAL CONTROL: ELSE IF STATEMENT

if (age < 16) { if the conditional evaluates to *true* / *false*

+ → System.out.println("You can't drive!");
}

f → else if (hasLicense == false) { if the conditional evaluates to *true* / *false*

+ → System.out.println("You can't drive!");
}

f → else {
 System.out.println("You can drive!");
}

else if:

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

NESTED IF STATEMENTS

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