Java Notes Chapter 05 – Making Decisions

**IF STATEMENT:**

* When you want to take action if a Boolean expression is true
* Java IF statement always includes ‘( )’ with bool expression inside
* Any expression that evaluates as true or false:
* Simple Boolean variable
* Call to a method that returns a bool value

**PITFALLS:**

* Putting ‘;’ a the end of the first line
* Using assignment operator (=) instead of equivalency (==)

\*Can store the Boolean expression’s value in a Boolean variable

* Comparing objects using relational operators
* Can use ‘==’ & ‘=’ 🡪 compares object memory addresses instead of their values

\*\*To compare values of objects, write specialized methods that compare methods & variables of object

**IF ELSE STATEMENTS:**

* When you want to take an action if a Boolean expression is true and a different action if its false
* Only one action takes place. ‘;’ at the end of each statement
* ‘{ }’ required for multiple actions / statements

**NESTING IF & IF ELSE STATEMENTS:**

* Useful when 2+ conditions must be met before an action takes place
* \*Multiple nested if.. else statements are paired in FILO

**USING LOGICAL AND & OR OPERATORS:** combines Boolean tests into a single expression

* AND: ( && ) creates a compound bool expression that’s true when both of its operands are true
* OR: ( || ) creates a compound bool expression that’s true when at least one of its operands are true
* Must include a complete Boolean expression on each side
* Short-Circuit Evaluation: expressions on each side of the && and || operators are only evaluated as far as necessary to determine whether the entire expression is true or false

**MAKING ACCURATE & EFFICIENT DECISIONS:**

* Making Accurate Range Checks: \*most efficient to ask the question that’s most likely to be true\*
* Using && and || Appropriately:

**SWITCH STATEMENT:**

* Alternative to using a series of nested if statements
* Convenient when there are several alternative courses of actions that depend on a single integer, character, or string value
* Uses 4 keywords:
* Switch: starts statement & immediately followed by a test expression enclosed in ‘ ( ) ’
* Case: followed by one of the possible values for the test expression and a ‘ : ’
* Break: optionally terminates a switch statement at end of each case
* Default: optionally used prior to any action that should occur if the test variable does not match any case
* Not required to list case values in ascending order
* \*\*Intentionally omit the break statements if you want all subsequent cases to execute after the test variable is matched
* Don’t need to write code for each case in a switch statement if multiple cases have the same outcome

**USING THE CONDITIONAL & NOT OPERATORS:**

* Conditional: abbreviated version of the if else statement

**-**Requires 3 expressions separated with a ‘ ? ’ and a ‘ : ’

testExpression ? trueResult : falseResult;

* Not: ( ! ) negates the result of any Boolean expression

-True expression 🡪 False and vice versa

**OPERATOR PRECEDENCE:**

* The order in which you use operators makes a difference ( && higher than || )
* Use parenthesis to change precedence or make your intentions clearer

**ADDING DECISIONS & CONSTRUCTORS TO INSTANCE METHODS:**

* Whether values are assigned to objects via constructors or by mutator methods, you often will need to use decisions to restrict the values assigned to fields
* Ex. Employee class constructor accepts params to set two field values & contains if statements to ensure the values for the fields are within the MIN and MAX
* Could include set methods with similar if statement rules 🡪 call methods from constructor