**04.01: NUMBER SYSTEMS**

Decimal, octal, hex, and binary

**04.02: PRIMITIVE DATA TYPES: BOOLEANS**

* **Conditionals** – a statement involving a binary decision
* **Relational operators** – an operator used to compare two values, variables, or expressions

if(average >= 90)  
   System.out.println ("Grade = A")

if(age <= 10)  
   admissionFee = 4.50;

if(hours > 40)  
   overtimePay = (hours - 40) \* rate \* 1.5;

if(balance < 0)  
   fee = 24;

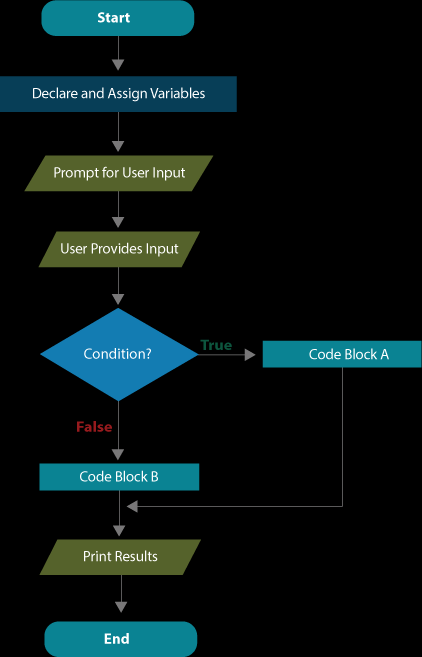
if(side1 != side2)  
   System.out.println("Not a square.");

if(tempF == 32)  
   tempC = 0;

* Relational operator symbols
  + > Greater than
  + < Less than
  + >= Greater than or equal to
  + <= Less than or equal to
  + == Equal to
  + != Not equal to
* Involve a branch of a program’s **flow of control** – the order in which statements are executed
* Boolean is a primitive data type
  + “boolean”
  + Can be “true” or “false”

**04.03: CONDITION STATEMENTS: IF**

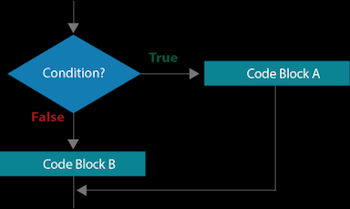
* “if”
  + Always boolean



* + If the statement inside the parentheses of “if()” is true, it executes the code inside its brackets (or just a single line if that’s all that is put).
  + Else, it will continue with the code that comes after
* Never ends with ;

**04.04: CONDITION STATEMENTS: IF-ELSE**

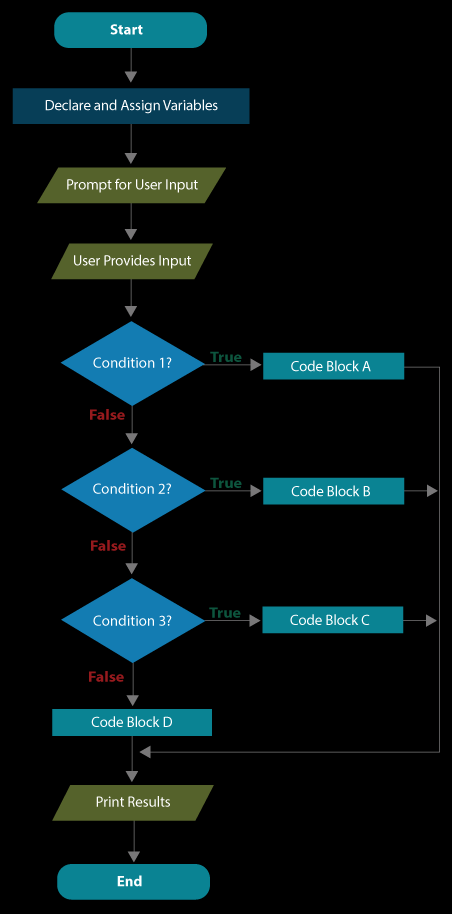
* **Block of code –** a segment of code located between opening and closing curly braces



* if(bool condition), do this
* else, do that

**04.05: CONDITION STATEMENTS: IF-ELSE-IF**

* Java’s if-else-if structure is designed for conditions that have multiple alternatives



* if(boolean Condition 1)

execute Code Block A;

else if(boolean Condition 2)

execute Code Block B;

else if(boolean Condition 3)

execute Code Block C;

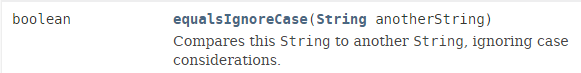
else

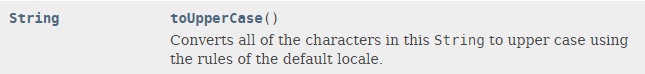
execute Code Block D;

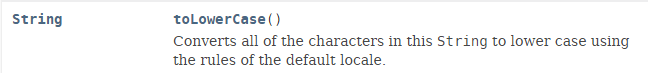
**4.06: COMPARING STRINGS**

* **Identity equality** – refers to whether two objects are actually the same object
* **Content equality** – indicates whether two objects contain equal values.
* The simplest way to evaluate the equality of String objects is with the equals() method
  + boolean       **equals(Object** anObject)  
                         Compares this string to the specified object
* Strings are objects! Always use the String methods to compare their values. If relational operators are used, you may get some unexpected results. Try running the code below. You'll discover str1 and str2 both have a content value of "A", but comparison with the relational operator yields false and the comparison with the String method yields true.
* String str = "APCS";  
  String str1 = "" + str.charAt(0);  
  String str2 = "" + str.charAt(0);  
  System.out.println("The string values: " + str1 + " and " + str2);  
  System.out.println(" str1 == str2 : " + (str1 == str2) );  
  System.out.println(" str1.equals(str2) : " + str1.equals(str2) );
* Some of the most useful String methods include the following:

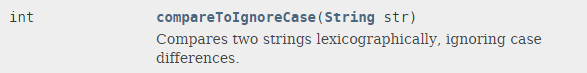












* **Lexicographically** - similar to the ordering that one might find in a dictionary.
  + -1: before
  + 0: at
  + 1: after

equals()

equalsIgnoreCase()

toUpperCase()

toLowerCase()

compareTo()

compareToIgnoreCase()

**4.07: LOGICAL OPERATOR**

* Java’s logical operators
  + && (AND)
  + || (OR)
  + ! (NOT)
* Pretty self-explanatory… (Check EEL 2880 notes for questions)