**ITMD\_510 Week 1 Notes –1/10/18 Wed.**

**Java Review – Best practices**

**[ Variables ]**

**Defined: a value stored in memory that can change**

Declarations

Establish good naming conventions!

Ex. **emp\_Rate or empRate empHours, empId**

Scope (variable visibility)

Local vs. Class level (static)

**[Constants]**

**Defined: a value stored in memory that cannot change**

Use of keyword -> **final**

final int ival = 20; //**value needed immediately upon declaration**!

**[ Data Types ]**

|  |  |
| --- | --- |
| **Primitives** | **Reference or Class based examples** |
| * byte | Random |
| * boolean | String |
| * float | Float |
| * double | Double |
| * int | Integer |
| * short | Short |
| * long | Long |
| * char | Character |

Ex.

int j=0, k=1;

double a = 2.0;

double b = 5;

Avoid common pitfall!

int result = 3/4; //0 as truncation occurs

**[ Operators ]**

Know operator precedence and associativities!

Order of operations (highest to lowest)

1. Arithmetic

2. Conditionals ( and **&&**, or **||**, not **!** )

3. Relational ( >,<,>=,<=,!=, == )

Keynote: use shortcut operators in assignment statements

**++**count; //**pre increment** operator (change to variable takes place immediately)

vs.

count**++**; //**post increment** operator (change to variable takes place after assignment ends)

same as

count = count + 1;

or

count+=1;

Avoid common pitfall!

Checking for equality!!

FOR NUMERIC TYPES USE

int a = 12;

(5 == a)

FOR STRING TYPES USE **EQUALS** METHOD!

String name1 = “Joe”, name2 = “Jack”;

(name1.**equals**(name2))

**[ Conditionals ]**

if statements

if(a>b)

//if true do something

else

//if false do something else

Make use of curly bracies { } to combine statements

Ex.

if(a>b) **{**

result = a;

System.out.print(“Value = “ + result);

**}**

else **{**

result = b;

System.out.print(“Value = “ + result);

**}**

Other forms- multi branched, nested

if alternative – the CEO (**C**onditional **E**xpression **O**perator)

Known as a ternary operator - ( **? : )**

*format*

conditon **?** *true* expression **:** *false* expression

Ex.

int result = a>b ? a : b;

if statement equivalent

if(a>b)

result = a;

else

result = b;

**[ Conditional ~ Truth tables]**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| And **&&**  **Expression- a>b && b>c**  Cond. Result | | | Or **||**  **Expression- a>b || b>c**  Cond. Result | | | Not **!**  **Expression- !(a>b)** | |
| **T && T** | **T** | **T || T** | | **T** | **!T = F** | |
| F && T | F | **F || T** | | **T** | **!F= T** | |
| T && F | F | **T || F** | | **T** |  | |
| F && F | F | F || F | | F |  | |

**[ Loops ]**

while, do-while, for, for enhanced

ex. while (some *condition* holds true)

Ex.

int count = 0;

**while** (count<10) {

//do something

count++; //increment counter variable

}

**[ Arrays ]**

* Index or subscript based
* Holds a series or list of values
* Can be fixed or dynamic

Chief advantages

Can be sorted, searched, averaged, find max/min values, printed…

Any disadvantages???

Es.

int array[] = {1,2,3,4,5};

array[2]=14; //update array at subscript 2, element #3

|  |  |  |
| --- | --- | --- |
| Element number | Subscript number (or index) | Value at subscript position |
| 1 | 0 | 1 |
| 2 | 1 | 2 |
| 3 | 2 | 14 |
| 4 | 3 | 4 |
| 5 | 4 | 5 |

Cycle thru array up to array length - 1

for (int i = 0; i < array.**length**; i++)

System.out.print(array[i] + “ “);