### **If-Statements**

* Construct that lets you do conditional execution of statements
* Can specify alternate paths of execution
* Of the form below; if boolean expression evaluates to true, execute the statement(s).

if (*boolean expression*) {

*statement1*;

*statement2*;

*statement3*;

}

* For example:

if (num > 0) {

isPositive = true;

}

### **Parentheses and Braces**

* Consider:

if **(**num > 0**) {**

System.out.println("num is positive");

**}**

* Must have ()'s around boolean expression
* Can omit {}'s if it is a single statement, for example

if (num > 0)

System.out.println("num is positive");

### **If-Else Statements**

* Of the form below; now if boolean expression evaluates to false, execute the statement(s) in the else block.

if (*boolean expression*) {

*statement1*;

*statement2*;

*statement3*;

}

else {

*statement4*;

*statement5*;

}

* For example

if (num > 0) {

System.out.println("num is positive");

}

else {

System.out.println("num is not positive");

}

### **Nested If**

It is possible to you use If in a nested fashion. Consider:

if (num > 0) {

if (num%2 == 0) System.out.println("Even!");

else System.out.println("Odd!");

}

else {

System.out.println("Sorry, negative number!");

}

### **Logical Operators**

* Used to create compound boolean expressions.
* They are ! (not), && (and), || (or) and ^ (exclusive or).
* **NOT** example: !(age > 18)
* **AND** example: (age > 18) && (weight <= 140)
* **OR** example: (age < 35) || (bornInUSA == false)
* **XOR** example: (temp > 100) ^ (ph < 1.0)

### **Switch Statements**

* Execute statements based on a value
* For example:

int incomeTier;

double taxRate;

switch (incomeTier) {

case 0: taxRate = 0;

break;

case 1: taxRate = 0.1;

break;

case 2: taxRate = 0.2;

break;

default: taxRate = 0.3;

}

* Switch expression must be included in parentheses
* Switch expression must be of the same type as values listed next to case
* Use break to exit out of the switch statement
* Use default (optional) as a catch all

### **Precedence Order**

The various operators you have learned about have the following order of precedence:

* post increment and decrement (var++, var--)
* unary +, -; pre increment and decrement (++var, --var)
* casting (e.g. (int)d)
* !
* \*, /, %
* binary +, -
* <, <=, >, >=
* ==, !=
* ^
* &&
* ||
* =, +=, -=, \*=, /=, %=

### **Exiting Program**

You can exit from a program by calling the exit method as below:

if (numEntry < 0)

System.exit(0);

### **Initializing and Conditionals**

Note that the code below may give a compiling error says that myVar needs to be initialized.

int myVar;

int numEntry = input.nextInt();

if (numEntry < 0)

myVar = 5;

System.out.println(myVar);

This is because the compiler can not determine if myVar was set before it is being used as this decision is made at runtime.

### **Pitfalls**

With the use of boolean expressions and conditionals, you may run into unexpected behavior by:

* Mixing == and =

if (i = 0)

System.out.println("i is zero");

* Forgetting Braces (only the first statement will be part of the if)

if (i == 0)

System.out.println("Result:");

System.out.println("i is zero");

* Adding semicolon by mistake at end of a if condition, eg.

if (i == 0)**;**

System.out.println("i is zero");

* Checking for value of a boolean, eg. "if (myBool == true).." is the same as "if (myBool).."
* Dangling else
* Equality test for floating point numbers (covered in quiz)
* Duplicate code in if/else clauses