Chapter 3

Decisions

**3.1 The if Statement**

* Syntax 3.1

if (condition)

{

statements;

}

else (condition) The else statement doesn’t have to be there

{

statements;

}

**3.2 Comparing numbers and strings**

* Relational operators:

|  |  |  |
| --- | --- | --- |
| Java | Math Notation | Description |
| > | > | Greater than |
| >= | ≥ | Greater than or equal |
| < | < | Less than |
| <= | ≤ | Less than or equal |
| == | = | Equal |
| != | ≠ | Not equal |

* To test whether two strings are equal to each other, you must use the method called equals:
  + If (string1.equals(string2))…
  + DO NOT use == operator to compare strings

**3.3 Multiple Alternatives**

* For multiple if statements you can use if, if else… if else, else.

**3.4 Nested Branches**

* Nested if: An if statement inside and if statement

**3.5 Problem solving flowcharts**

****

**3.6 Problem**

**solving: Test Cases**

**---Skip---**

**3.7 Boolean Variables and Operators**

* To store a condition that can be true or false use a Boolean variable
  + Ex: Boolean failed = true;
* An operator that combines Boolean conditions is called a Boolean operator
* Java has 2 Boolean operators
  + The && operator (called and) yield true only when both conditions are true
    - Ex: if (temp > 0 && temp < 100)

{System.out.println(“liquid”);}

* + The || operator (called or) yields the result true if at least of’ the condition is true
    - Ex: if (temp <= 0 | | temp >= 100)

{System.out.println(“Not Liquid”);}

* The ! operator takes single condition and inverses it (not Boolean operator)
  + Ex: if (! Frozen) {System.out.println(“Not Frozen”);}
* 1(A && B) is the same as !A || !B
* !(A | | B) is the same as !A && !B

**3.8 Application: Input Validation**

* Whenever your program accepts user input, you need to make sure it is valid, like maybe no letters or negative numbers etc
* To set parameters you can use
  + *If (floor <= 0 | | floor > 0)*

{

*System.out.println(“Error: The floor must be between 1 and 20.”)*

*}*

* Dealing with an input that is not a valid integer (like letters) is more serious to solve, because a run-time exception occurs and the program is terminated
  + Use *hasNextInt* method, which checks whether the program is an integer
    - *If (in.hasNextInt())*

{

*Int floor = in.nextInt();*

*}*

*else*

*{*

*System.out.println(“Error: Not an integer.”);*

*}*