AP Computer Science – Chapter 3 Overview

**Boolean Data Type**

* holds a value of either true or false.
* Declaring a boolean variable *boolean lightsOn = true;*
* By default, a boolean variable is set to false;

**Logical Operators**

In the following table, assume: *boolean a = true; boolean b = false; boolean c = true;*

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Symbol** | **Purpose** | **Example** |
| NOT | ! | Reverses the value of the boolean variable | !a would be false!b would be true |
| AND | && | Both sides of the AND expression must be true for the AND expression to be true | a && b = falsea && c = true |
| OR | || | Only one side of an OR expression needs to be true for the OR expression to be true | a || b = trueb || b = false |

* The order of operations with logical operators is Not, And, Or
* So *a || b && c* means we first AND b and c, then we OR that result with a. b&&c = False, so true || false = *true*
* Be comfortable filling in truth tables for various boolean expressions
* deMorgans Law states:
	+ !(a && b) = !a || !b
	+ !(a || b) = !a && !b

**Random Numbers**

* Math.random() – generates a random double value between 0 and 1 (0.000001 to 0.9999999)
* To generate a random integer in a given range*: int num = (int) (Math.random() \* 100) + 1;*
	+ 100 is the number of possible value that could be generated
	+ +1 at the end is the lowest possible value in the range.
* Example # 1: *int num = (int) (Math.random() \* 10) + 15;*
	+ This creates a random integer starting at 15 with 10 possible values.
	+ The range is from 15 to 24
* Example # 2: *int num = (int) (Math.random() \* 20) -10;*
	+ This creates a random integer starting at -10 with 20 possible values.
	+ The range is from -10 to 9

**if statements**

* also called condition statements
* Used when we have code that we only want to execute under certain conditions
* Braces follow an if statement and if the condition is met, all code inside the braces is executed
* Braces can be omitted if there is only a single statement to be executed
* Can use logical operators to create more sophisticated conditions.
* Can be used to validate user entered data

You are writing a program to determine voting eligibility. You want a user to enter their age but obviously, you can't have a negative age and if the user does enter a negative value for age, you will inform them that an invalid age was entered:

 int age = s.nextInt(); //assume s is the Scanner name

 if (age < 0) System.out.println("Invalid age entered");

 <program continues>

|  |  |  |
| --- | --- | --- |
| **if Condition operator** | **Syntax** | **Example** |
| greater than | > | if( x > 10) |
| less than | < | if (testPercentage < 60) |
| equal to | == | if(password == 1234) |
| not equal to | != | if(password != 1234) |
| greater than or equal | >= | if(age >=65) |
| less than or equal | <= | if(iq <=70) |

* Instead of writing conditions with your if statements, you can evaluate a boolean variable
* When using a boolean variable in your if statement, you will only use == or !=.
* You can actually omit the equality operators if using boolean variables in if statements
* Imagine a boolean variable called *passed. passed* holds a true value if you passed the test, otherwise false;

 if(passed) System.out.println("You passed."); 🡨 is the same thing as (if passed == true)

 if(!passed) System.out.println("You failed."); 🡨 is the same thing as (if passed == false)

**if…else Statements**

* This gives us the ability to do something if the condition isn't satisfied.
* When using if..else statements, one of the two conditions must be met.

 if (userNumber%2 == 0) System.out.println(“Your number is even.”);

 else System.out.println(“Your number is odd.”);

**if..elseif…else statements**

* We use these when we have more than two possible conditions to be factored in
* You can have any number of elseif statements, each with its own code
* The else statement at the end is optional, but is often used to catch "all other possibilities".
* Once java finds a statement in which the if condition is satisfied, it doesn't check any more elseif statements.

You want to determine what class your high school is in for athletics, which is based on student population.

* Class A has 893 or more students.
* Class B has 429 to 892 students
* Class C has 207 to 428
* Class D has 206 or fewer students

 int studentPopulation = s.nextInt();

 if (studentPopulation >=893) System.out.println("Class A");

 elseif (studentPopulation >=429) System.out.println("Class B");

 elseif (studentPopulation >=207) System.out.println("Class C");

 else System.out.println("Class D"); //Only possibility left is class D, so use the else statement

**Nested if statements**

* If you have multiple conditions that you want to check, you can place if statements inside other if statements.
* The inside if statement is only executed if the outer condition is met.
* Indentation is critical to making these types of statements readable and to minimize errors with braces.
* Assume you have to be 18 years old and have $100 in cash to get into a casino.

 if (age >=18){

 if (moneyInWallet > 100.0) { //We only get to this if statement if age is 18 or more

 System.out.println(“You can enter the casino.”);

 }

 }