|  |
| --- |
| **Exam 1 \_ Fall 2021**  **Intro to Programming Java** |
| **Topics:** Ch.3 (3.1-3.24)  Ch.4 (4.1-4.6) |

**BASIC OUTPUT LINES**

|  |
| --- |
| Print text with cursor ending to right of printed text:  System.***out***.print("");  Print text with cursor ending at next line:  System.***out***.println(""); |

**ESCAPE SEQUENCES**

|  |
| --- |
| Tab: move the cursor to the next tab stop: \t  Newline: newline - go to first column in next line: \n Return: return to first column in current line: \r |
| Print “ \”  Print ‘ \’  Print \ \\ |

**VARIABLES**

|  |
| --- |
| 1. **int**: Whole numbers that can be positive or negative 2. **long**: Also for whole #’s, but not as often used because it takes up more space 3. **double**: Negative or positive number with decimal places 4. **float:** Also for #’s with decimals but too small usually so we just use double 5. **boolean:** EX: true, false. We start to use this for either/ or situations and later for if/or 6. **char:** EX: ‘2’ , ‘a’, ‘A’, ‘+’ , ‘!’ . This is for a single character 7. **String:** More than one character. Only one that is capitalized because it is so powerful. Always needs double quotation marks in between: “ …. “ |

**STRING CONCANTENATION**

|  |
| --- |
| Char concantenation:  **char** first, last;  first = 'L';  last = 'Z';  System.***out***.println(first + last);  Printed text: LZ |
| String concantenation:  String one = "Hello cute";  String two = "doggie!";  System.***out***.println(one + two);  Printed text: Hello cute doggie! |

**CAST OPERATORS**

|  |
| --- |
| Instead of combining two operands, they compare them.   1. **Are two values equal to each other? ==** 2. **Do two values not equal each other?** != 3. **Is value on left less than value on right** < 4. **Is value on left greater than the one on right?** > 5. **Is value on left less than or equal to right?** <= 6. **Is value on left greater than or equal to left?** >= |

**ARITHMETIC**

|  |
| --- |
| 1. **Addition:** + 2. **Subtraction:** - 3. **Multiplication:** \* 4. **Floating-point division:** /   - Data types: float and double.  **EX:** 10 / 2 = 5   1. **Integer division:** / or %   - Data types: int and long  **EX:** 7 / 2 = 3 quotient  **EX:** 7 % 2 = 1 remainder |

**MIXED TYPE ARITHMETIC**

|  |
| --- |
| 1. **int <op> int** = int 2. **int <op> double** = double 3. **double <op> int** = double 4. **double <op> double** = double 5. **Except division and percentage**   - If you want to do division with two ints and want a double (want an answer with a decimal place, then you would have to do one of the following)  **First option**: Instead of 3/2 type any of the following below….  - 3.0/2  - 3/2.0  - 3.0/2.0  **Second option:** Instead of 9 % 2 type any of the following below….  - 9.0 % 2  - 9 % 2.0  - 9.0 % 2.0 |

**INCREMENT & DECREMENT**

|  |  |
| --- | --- |
| What to write:  x += 3  x -= 4  x \*= y  x /= 4  x % = 16 | What it equals:  x = x + 3  x = x - 4  x = x \* y  x = x / 4  x = x % 16 |
| x = x \* (y + 1) | |

**HOW TO FORCE A VARIABLE TO BE A FLOAT**

|  |
| --- |
| - Label as float and put f after.  EX: **float** gpa = 2.30f; |

**ORDER OF OPERATIONS**

|  |
| --- |
| 1. **Grouping with parentheses**    1. (<expression>) 2. **Unary operators [Unary]**    1. +x    2. -x    3. Cast operators: (<type>) x    4. x++    5. x--    6. !x 3. **Multiplication and division operators**    1. x \* y    2. x / y    3. x % y 4. **Addition and subtraction operators [Binary]**    1. x + y    2. x - y 5. **Less than and greater than relational operators**    1. x < y    2. x > y    3. x <= y    4. x >= y 6. **Equality operators**    1. x ==y    2. x != y 7. **“And” Logical operator**    1. x && y 8. **“Or” logical operator**    1. x || y |

**INPUT / SCANNER METHODS**

|  |
| --- |
| 1. **.next() :** (string) Skip leading whitespace until a token is found. Stops at next white space. Return the token as a String value. A token is a sequence of non-whitespace characters. **Ex:** gecko , lily, or 53B@a! 2. **.nextLine() :** (string) Takes input up until the next next line character (enter). Enter twice to avoid issues. This one causes problems. 3. **.nextInt() :** Skip leading whitespace until a token is found. Return the token as an int value. 4. **.nextLong() :** Skip leading whitespace until a token is found. Return the token as a long value. 5. **.nextFloat() :** Skip leading whitespace until a token is found. Return the token as a float value. 6. **.nextDouble() :** Skip leading whitespace until a token is found. Return the token as a double value. |

**BASIC CODE FOR USER INPUT (Scanner class)**

|  |
| --- |
| **The highlighted portions below are what need to be added to get user input:**  **import** java.util.Scanner;  **public** **class** programName  {  **public** **static** **void** main(String[] args)  {  Scanner stdIn = **new** Scanner(System.***in***);  System.***out***.print("Price of purchase item: ");  price =stdIn.nextDouble();  } //end main  } //end class |

**PULL DATA FROM OTHER DOCUMENT**

|  |
| --- |
| **import** java.util.Scanner;  **import** java.io.File; |

**AND / OR / NOT**

|  |
| --- |
| 1. And operator : **&&**   - Combines two conditions  - True only if first condition is true **and** second condition is true   1. Or operator: **||**   - Combines two conditions  - True if first condition is true **or** second condition is true   1. Not: **!**   - Makes the opposite of the statement fulfill program |

**BOOLEAN LOGIC**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **OR**   |  |  |  | | --- | --- | --- | | X | Y | X||Y | | True | True | True | | True | False | True | | False | True | True | | False | False | False |   **AND**   |  |  |  | | --- | --- | --- | | X | Y | X&&Y | | True | True | True | | True | False | False | | False | True | False | | False | False | False |   **NOT**   |  |  | | --- | --- | | X | !X | | True | False | | False | True | |

**IF/ IF ELSE/ IF ELSE IF**

|  |
| --- |
| 1. **If:** Use when you want to do one thing or nothing. 2. **Else if:** Use when you want to do one thing or another thing. 3. **Else:** This is the catch all if nothing else is working |

**STRING METHODS**

|  |
| --- |
| 1. **.length ()**   - total characters   1. **.charAt()**   - starts counting at 0   1. **.equals()** |

|  |
| --- |
| **Exam 2 \_ Fall 2021**  **Intro to Programming Java** |
| **Topics:** Ch.4 (4.8 - 4.15) Ch.5 (5.1 - 5.6, & 5.8) |

**IMPORT MATH**

**MATH CLASSES**

|  |
| --- |
| Ex: Math.min(4,5);   1. <callTheClass>. <method>(arguments passed into a method); |

**BOOLEAN OPERATORS**

|  |
| --- |
| 1. Can be created through a boolean expression or through comparison operators. 2. Will go to true or false. |

**SWITCH STATEMENT**

|  |
| --- |
| 1. Works similar to the “if, else if” 2. Determination of which path to take is based off of a single value unlike the ‘if, else if’ statement. |

**WHILE LOOPS**

|  |
| --- |
| 1. Condition is the same as an if statement’s condition. 2. Usually evaluates to true or false. 3. An if statement that keeps running until something is true/false. 4. Boolean is the first thing to be checked. |

**DO WHILE LOOPS**

|  |
| --- |
| 1. Similar to a while loop but this will execute at least once. 2. Boolean is the last thing to be checked |

**FOR LOOPS**

|  |
| --- |
| 1. for (create variable; boolean expression; something that updates the variable) 2. a.k.a.: for (initialization; boolean; update what was initialized) 3. most common loops used |

**WRAPPER CLASSES**

|  |
| --- |
| 1. Make a primitive into an object. |

**WRAPPER CLASS TYPES**

|  |  |
| --- | --- |
| **String to Primitive** | **Primitive to String** |
| Integer.parseInt(<string>) | Integer.toString(<#>) |
| Long.parseLong(<string>) | Long.toString(<#>) |
| Float.parseFloat(<string>) | Float.toString(<#>) |
| Double.parseDouble(<string>) | Double.toString(<#>) |

**FOR VS WHILE LOOPS**

|  |  |  |
| --- | --- | --- |
|  | **For** | **While** |
| **Use** | The ‘for’ loop used only when we already knew the # of iterations. | The ‘while’ loop used only when the number of iteration are not exactly known. |
| **Condition** | If the condition is not put up in ‘for’ loop, then loop iterates infinite times. | If the condition is not put up in ‘while’ loop, it provides compilation error. |
| **Initialization** | In ‘for’ loop the initialization once done is never repeated. | In while loop if initialization is done during condition checking, then initialization is done each time the loop iterate. |
| **Iteration statement** | In ‘for’ loop iteration statement is written at top, hence, executes only after all statements in loop are executed. | In ‘while’ loop, the iteration statement can be written anywhere in the loop. |

**EXAM 2 NOTES**

|  |
| --- |
| \***Arguments**\_The things/#’s plugged into a math method.  \***Parameters**\_A variable that doesn’t have content yet. The thing in the declaration of the method that gets substantiated with the argument. |
| **Parameter EX:**  Math.min(int a, int b);  **Argument EX:**  Math.min(4,3); |
| \* **Reference Variable**\_ Points to an object in memory.  \* String objects are kind of treated like primitives in Java, but they are actually references. |

|  |
| --- |
| \***Primitives**\_ All the things that start with a lower case letter like int, double, boolean, and float. |

|  |
| --- |
| **Final \_ Fall 2021**  **Intro to Programming Java** |
| **Topics:** Ch.6 Supplement Ch. 9.1-9.6 & Ch.9 Supplement 9.7 & 9.8 |

|  |  |
| --- | --- |
| **Array Element’s Type** | **Default Value** |
| Integer | 0 |
| Floating- point | 0.0 |
| Boolean | False |
| Reference (String) | null |