Pearson Pseudocode to Java – Part 1

1. The tables below should help students convert from *Pearson pseudocode* to *Java*. Take note that Java requires a semicolon (`;`) after statements (unless the statement ends with a closing curly brace). Please pay particular attention to the relative indentation of the code.

Declaring Variables

Pearson Pseudocode Syntax	Java Syntax
dataType variableLabel; • `INTEGER index` • `BOOLEAN isEnabled` • `CHARACTER firstLetter` • `REAL weightInKg` • `STRING lastName` *In Pearson pseudocode, there is no need to declare variables, you simply assign a value to a label; however, it may help the reader understand your code if you do declare them at the start of your code.*	***dataType variableLabel*`;`** • `int index;` • `boolean isEnabled;` • `char firstLetter;` • `double weightInKg;` • `String lastName;` *Note: for data types, primitive types use camel case, while classes use Pascal case (upper camel case).* `String` *is upper case because it is a class and not a primitive type. The semicolon is required at the end of these statements.*

Assigning Values To Variables

Pearson Pseudocode Syntax	Java Syntax
`SET` ***variableLabel*** **`TO`** ***expression*** • `SET counter TO 0` • `SET myString TO 'Hello'` *Note: data types are not required for Pearson pseudocode, so we simply assign values to any variable label without any regard to the type of data we are storing. If we do declare the type of variables, it is done at the top of the pseudocode on a separate line.*	***dataType variableLabel*** **`=`** ***expression*`;`** • `int index = 0;` • `boolean isEnabled = true;` • `char firstLetter = 'N';` • `double weightInKg = 42.6;` • `String lastName = "Nielsen";` ***variableLabel*** **`=`** ***expression*`;`** • `index = n + 1;` • `isEnabled = false;` • `firstLetter = 'C';` • `weightInKg = 43.0;` • `lastName = "Chen";` *Note: literals for the data type* `char` *must be enclosed in single quotes* (`'`)*, while string literals must be enclosed in double quotes* (`"`).

Output

Pearson Pseudocode Syntax	Java Syntax
`SEND` ***expression*** **`TO DISPLAY`** • `SEND 'Hello' TO DISPLAY`	**`System.out.print(`** ***expression*** **`);`** • `System.out.print("Hello");` • `System.out.print(isEnabled);` **`System.out.println(`** ***expression*** **`);`** • `System.out.println("Hello");` • `System.out.println(index);` *Note: after* `println`*, the next output will be printed on a new line; whereas after* `print`*, the next output will continue on the same line.*

Pearson Pseudocode to Java – Part 1

Selection

Pearson Pseudocode Syntax	Java Syntax
IF ***condition*** **THEN** ***statement(s)*** **END IF** • `IF (x > 0 ) THEN` `SEND 'Positive' TO DISPLAY` `x = 0` `END IF` *Note: Pay attention to proper indentation.*	**if (** ***condition*** **) {** ***statement(s)*****;** **}** • `if (x > 0) {` `System.out.print("Pos");` `x = 0;` `}` *Note: The curly braces are optional only in the case when there is a single command statement. However, in most cases it is considered good practice to always include the curly braces.* **if (** ***condition*** **)** ***singleStatement*****;** ***nextStatement(s)*****;** • `if (x > 0) {` `System.out.print("Pos");` `y = 0;` *Note: in the case immediately above, the statement* `y = 0;` *is outside of the* `if` *statement, so it will be executed regardless of the value of* `x`*.*
IF ***condition*** **THEN** ***statement(s)*** **ELSE** ***statement(s)*** **END IF** • `IF (x >= 0 ) THEN` `SEND 'asset' TO DISPLAY` `ELSE` `SEND 'debt' TO DISPLAY` `END IF` *Note: Pay attention to proper indentation.*	**if (** ***condition*** **) {** ***statement(s)*****;** **} else {** ***statement(s)*****;** **}** • `if (x > 0) {` `System.out.print("asset");` `} else {` `System.out.print("debt");` `}` *Note: Any number of statements can be contained within the curly braces.* **if (** ***condition*** **)** ***singleStatement*****;** **else** ***singleStatement*****;** ***nextStatement(s)*****;** *Note: again, curly braces are optional, but still recommended for cases when there is only a single statement contained within the* `else` *code block.*

Pearson Pseudocode to Java – Part 1

Iteration

Pearson Pseudocode Syntax	Java Syntax
WHILE *condition* **DO** *statement(s)* **END WHILE** • WHILE (x > 0) DO SEND x TO DISPLAY x = x – 1 END WHILE	**while (** *condition* **) {** *statement(s)***;** **}** • while (x > 0) { System.out.print(x); x = x - 1; } *Note: As with the* `if` *statement, the curly braces for a* `while` *loop are optional only in the case when there is a single command statement in the code block.*
REPEAT *statement(s)* **UNTIL** *condition* • REPEAT SEND x TO DISPLAY x = x - 1 UNTIL x > 0	**do {** *statement(s)***;** **} while (** *condition* **);** • do { System.out.print(x); x = x – 1; } while (x > 0); *Note: curly braces are optional when there is only a single statement contained within the code block.*

Pearson Pseudocode to Java – Part 1

Subprocesses (Java methods)

Pearson Pseudocode Syntax	Java Syntax
PROCEDURE *procedureLabel* **(** *parameter(s)* **)** **BEGIN PROCEDURE** *statement(s)* **END PROCEDURE** • `PROCEDURE printName(name)` `BEGIN PROCEDURE` `SEND name TO DISPLAY` `END IF`	**public static void** *methodLabel* **(** *parameter(s)* **) {** *statement(s)* ***}*** • `public static void printName(String name)` `{` `System.out.println(name);` `}` *Note: There is no distinction between procedures and functions in Java. If a method does not return a value, we use the keyword* `void`*, as shown above.*
procedureLabel **(** *parameter(s)* **)** • `printName("Chris")`	*procedureLabel* **(** *parameter(s)* **)** • `printName("Chris")`
FUNCTION *functionLabel* **(** *parameter(s)* **)** **BEGIN FUNCTION** *statement(s)* **RETURN** *expression* **END FUNCTION** • `FUNCTION area(length, width)` `BEGIN FUNCTION` `value = length * width` `RETURN value` `END FUNCTION`	**public static** *returnDataType* *methodLabel* **(** *parameter(s)* **) {** *statement(s)* ***}*** • `public static int area(int l, int w)` `{` `int value = l * w;` `return value;` `}` *Note: parameters are given in a comma-separated list, and the data type must be specified for each parameter in the parameter list.*
SET *variableLabel* **TO** *functionLabel* **(** *parameter(s)* **)** • `SET a TO area(5, 6)` *Note: the function call should ideally be written on a single line.*	*variableLabel* **=** *functionLabel* **(** *parameter(s)* **);** • `a = area(5, 6);` *Note: in this case, the variable,* `a`*, must have been declared as type* `int` *since the function returns a value of type* `int`*.* *Note: methods can be used anywhere where an expression can be used. The method text is replaced by the method's return value once execution of the method is complete.*