## READ theSE RULES below for CODE ALIGNMENT and INDENTATION, and FOLLOW them STRICTLY WHEN CODING

Please follow these rules when coding in Eclipse and Greenfoot. In addition, please read textbook side notes “Scope coloring and indentation” in page 26~27, which describes additional code indentation tools in Greenfoot.

**CODE ALIGNMENT and INDENTATION RULES:**

* **RULE 1: if the opening curly brace { starts a new line, then the closing curly brace } must be ALIGNED in the same column as the opening curly brace, such as**

while (count != 12)

{

//body of the loop goes here. And notice that code INDENT one level inside the pair of curly braces.

}

**or if the opening curly brace { is placed at the end of a line, then the closing curly } brace must be ALIGNED in the same column as the first column of the line where the opening curly brace { locates, and in this example, it is the w character of the while keyword, such as**

while (count != 12){

//body of the loop goes here. And notice that code INDENT one level inside the pair of curly braces.

}

* **RULE 2: each pair of opening and close curly braces { } is called a block, and indent EQUALLY for all lines inside a block UNLESS a new block is encountered – then indent one level more inside the new block. Such as:**

while (count != 12) {

//indent one more level because it meets a new pair of block. The loop body goes here.

score = 10; // indent equally as the line above

if (score > 8) // again, indent equally as the line above

{

// indent one more level, because a new pair of curly braces (or a new block)is encountered.

}// end of the if block

else{ // this line indent equally as the if block above

// indent one more level than the else line, because a new pair of block is encountered.

// the second line indent equally as the first line in this block

for (int i=0; i< 20; i++){ // indent equally as the line above

// indent one more level, because a new pair of curly braces is encountered.

} // end of for loop

} // end of the else block

}// end of the while loop

The sample source code provided above has comments starting with // , to help illustrate this indentation rule, and you don’t need to have comments in each line of your code.

* **RULE 3: Use even number of SPACES to indent inside a block (2 or 4 spaces is suggested), and use the same number of spaces for all indentations.** You should either **use space key only** or **use tab key only** for indentation. D**O NOT** **mix space and tab key together** to achieve indentation, and the reason is: tab key may be defined as different number of spaces in different platforms or different editors, thus it will result in unaligned issue in different platforms or editors. **Also, the number of spaces you used for indentation must be consistent across your code.** If one level indents 2 spaces, and the next level indents 4 spaces, this is not consistent indentation.
* **RULE 4: a line of source code should not exceed 120 columns.** Reason: if a line of source code is more than 120 columns, then the reader of the source code usually needs to scroll left and right in the editor, in order to read the whole line, which is not considered as reader-friendly. Keeping the source code less than 120 columns per line makes it possible to display the whole line in one screen.

One good news is that: you don’t need to count the column number in source code, because with text editors, the column number is always indicated at the bottom information bar of the editor windows. You can verify this by moving the editing cursor from left to right, one space at a time, and you can see the column number indicator increases one at time, accordingly.

What if a source code statement needs to exceed 120 columns? ---- Solution is to break a long line into multiple short lines, with each short line not exceeding 120 columns. For example:

System.out.printf("After processed, " +

"your application is %s, and the code is %d. \n",

applicationStatus,

reasonCode);

In the above example, a long statement is separated into 4 lines, and each line is less than 120 columns. Pay attention to the alignment rule in the above example: **the second continuous line and the following continuous line(s) should indent one more level with respect to the first line, to indicate that they are the continuous lines of the first line**, as shown in the example source code above.

* **RULE 5: there is this 80-column norm:** for terminal output generated by **System.out.println,** or **System.out.println ,** or **System.out.printf , the output content should not exceed 80 columns per line.** Reason: usually in most terminals, the width is 80 columns, and if you generate more than 80 columns in one line, the output content that is beyond 80 columns will wrap in the consecutive line(s).

If we take the println method in Rule 4 as an example, its output result is as below, and we can verify that its length is less than 80 columns.

*After processed, your application is approved, and the code is 9.*

If you have a terminal output which has more than 80 columns in content, you may want to output the content in multiple lines, so that each line only output less than 80 columns. Multiple-line output can be achieved by inserting the newline \n symbol into the output stream of one **System.out.print** statement, or **System.out.printf** statement. Or you can simply use multiple **System.out.println** statement, with each statement output a separate line. The ***ln*** at the end of the **System.out.println** means that: output a newline symbol \n at the end of each output item, so that it naturally starts a new line for the following output.

The table below gives examples of both incorrect (left column) and correct (right column) coding:

|  |  |
| --- | --- |
| A sample output line of more than 80 columns. Incorrect. | Break one long line of output into multiple short lines, and each line is less than 80 columns. Correct. |
| String s1 = "this is the first statement. ";  String s2 = "this is the second statement. ";  String s3 = "this is the third statement. ";  System.out.printf("%s %s %s", s1, s2, s3); | String s1 = "this is the first statement. ";  String s2 = "this is the second statement. ";  String s3 = "this is the third statement. ";  System.out.printf("%s \n %s \n %s \n", s1, s2, s3); |

* **RULE 6: We should always have a pair of curly braces to enclose the body of a for loop, a while loop, a do-while loop, and a conditional branch in if-“else if”-else statement, even though there is as few as one statement only or zero statement inside the body.**

Please look at the table below, which summarizes both the correct and the incorrect format.

|  |  |
| --- | --- |
| **Incorrect coding format, not recommended** | **Correct coding format, recommended** |
| for(int i =0; i< 20; i++)  System.out.printf("%4d \n", i); | for(int i =0; i< 20; i++){  System.out.printf("%d \n", i)  } |
| int i = 0;  while(i<20)  System.out.printf("%4d \n", i++); | int i = 0;  while(i<20){  System.out.printf("%4d \n", i++);  } |
| int keepSearching = 1;  Scanner input = new Scanner(System.in);  do  keepSearching = input.nextInt();  while(keepSearching == 1); | int keepSearching = 1;  Scanner input = new Scanner(System.in);  do{  keepSearching = input.nextInt();  }while(keepSearching == 1); |
| if(speed >= 80)  fine = 200;  else if(speed >= 60)  fine = 100;  else  fine = 0; | if(speed >= 80) {  fine = 200;  }  else if(speed >= 60) {  fine = 100;  }  else {  fine = 0;  } |

Even though in the above table, the code in the left column can still compile, but they are not recommended for coding practice, as stated in Google and Oracle’s Java coding standards.

In addition, **omitting the pair of curly braces makes your code error prone**. Look at the example below:

|  |  |
| --- | --- |
| **Incorrect coding format without curly braces, even though the code compiles, it creates logical error or infinite loop. Should be avoided.** | **Correct coding format with curly braces. Recommended.** |
| int total = 0;  for(int i =0; i< 20; i++)  total += i;  System.out.printf("%d\n", total); | int total = 0;  for(int i =0; i< 20; i++){  total += i;  System.out.printf("%4d \n", total);  } |
| int i = 0;  while(i<20)  System.out.printf("%4d \n", i);  i++; | int i = 0;  while(i<20){  System.out.printf("%4d \n", i);  i++;  } |
| if(speed >= 80)  fine = 200;  else if(speed >= 60)  fine = 100;  else  fine = 0;  speeding = false; | if(speed >= 80) {  fine = 200;  }  else if(speed >= 60) {  fine = 100;  }  else {  fine = 0;  speeding = false;  } |

Both left and right column in the above table can compile, but the left columns have serious errors:

* In the for loop at the left column, because there is no pair of curly braces enclosing the ***printf*** statement, this statement is not included in the body of the for loop, thus it does not print out the value of variable total in each loop iteration, resulting a logical error.
* In the while loop at the left column, because there is no pair of curly braces enclosing the ***i++*** statement, this statement is not included in the loop body, therefore variable i will never be increased, so it will always stays at its original value 0, and the while loop will become infinite loop that does not terminate.
* In the else statement at the left column, because there is no pair of curly braces enclosing the statement: ***speeding = false*** , therefore this statement is not included in the body of the else branch, and variable ***speeding*** will not be set to false when the speed is less than 60, and that is a logical error.

In other cases, without the pair of curly braces, the code won’t even compile.

|  |  |
| --- | --- |
| **Without curly braces enclosing the body of for loop, while loop, do-while loop, or the if statement, or “else if” statement, the code does not compile.** | **With curly braces enclosing the body of do-while loop, or the if statement, or the “else if” statement, the code compiles.** |
| int total = 0;  for(int i =0; i< 20; i++)  System.out.printf("%4d \n", i);  total += i; | int total = 0;  for(int i =0; i< 20; i++){  System.out.printf("%4d \n", i);  total += i;  } |
| int i = 0;  do  System.out.printf("%4d \n", i);  i++;  while(i < 20); | int i = 0;  do{  System.out.printf("%4d \n", i);  i++;  }while(i < 20); |
| if(speed >= 80)  fine = 200;  speeding = true;  else if(score >= 60)  fine = 100;  speeding = true;  else  fine = 0; | if(speed >= 80) {  fine = 200;  speeding = true;  }  else if(score >= 60) {  fine = 100;  speeding = true;  }  else {  fine = 0;  } |

In the above table, the left column does not compile, because:

* In the for loop at the left column, statement ***total += i;*** is not included in the for loop, because there is no curly braces enclosing it, and variable i is only defined inside the for loop, thus this line won’t compile.
* In the do-while loop, without the enclosing curly braces, only the first statement after the ***do*** keyword is included in the loop body, and the second statement with the followings are not included, and they won’t compile.
* In the conditional if and “else if” statement, the code won’t compile, because the first statement ***speeding = true***; cannot be included into the if statement, and the if statement can only include statement ***fine = 200***; thus the following “else if” statement does not have a leading if statement, which results in grammar error.

**The above rules are simple but it is also very IMPORTANT to follow them, so that your code looks beautiful and readable!**

Code alignment and indent contains simple rules, and they are not difficult or sophisticated for students to follow and learn. It is just a matter of habit in coding. Hopefully through this class, you develop a good habit for coding.

And remember: do not copy java source code example from any WORD document into Eclipse or Greenfoot, because WORD document for formatted, thus it contains characters not recognized by Eclipse or Greenfoot.

Question: where are these rules coming from?

Answer: they are a subset of Java coding standard rules from Google and Oracle:

<https://google.github.io/styleguide/javaguide.html>

<http://www.oracle.com/technetwork/articles/javase/codeconvtoc-136057.html>