Code Layout

The goal of code layout/formatting is to show logical structure.

Good layout is shows intention, is consistent, improves readability, and withstands modification.
Status #1

Did you get IntelliJ installed on your machine

A) Yes
B) No
Status #2

Did you order/do you have a copy of the book?

A) Yes
B) No
Which is best?

A) for(int i=0;i<str.length();i++){  
B) for (int i=0; i<str.length(); i++) {
C) for (int i = 0; i < str.length(); i++) {
D) for (int i = 0 ; i < str.length() ; i ++) {
E) for( int i = 0 ; i < str.length() ; i ++ ){

foo(a)  
for (  

Which is better?

A) for (int i = 0; i < args.length; i++)
B) for (int i = 0;
    i < args.length;
    i++)
C) Both are fine
D) Both are lacking
Which is better?

A) if (game[i][index] != c)
B) if ( game[i][index] != c )
C) Both are fine
D) Both are lacking
Which is better?

A) char [][] game = new char[3][3];
B) char [][] game = new char[ 3 ][ 3 ];
C) Both are fine
D) Both are lacking
Hmmm...

- I like spacing operands like the following:
  \[
  \text{int } x = a + b + c + d + 17;
  \]

- But in the below, I personally prefer the second option:
  \[
  \text{data}[i][i] = \text{data}[i-1][i-1];
  \]
  \[
  \text{data}[i][i] = \text{data}[i-1][i-1];
  \]
What is wrong with this?

int z = \( (b == 10) ? c : d ; \)

int foo = a + \( (b == 10 ? c : d) + e ; \)

- too much stuff
- requiring knowledge of precedence.
Which is better?

A) int parenthesis;
   parenthesis = 0;
B) int parenthesis = 0;

C) Both are fine
D) Both are lacking
Which is better?

A) \[ \text{int paren} = 0, \text{eqnLength} = \text{eqn.length}(); \]

B) \[ \text{int paren} = 0; \]
   \[ \text{int eqnLength} = \text{eqn.length}(); \]

C) Both are fine

D) Both are lacking
What is wrong with this?

```c
void aFunc(int j, int k) { 
j++; 
k++;
}
```
Which is better?

A)
```java
if (three) {
    System.out.println("Valid");
}
else {
    System.out.println("Invalid");
}
```

B)
```java
if (three) {
    System.out.println("Valid");
} else {
    System.out.println("Invalid");
}
```

C) Both are fine  
D) Both are lacking
Which is better?

A)  
```java
if (three) {
    System.out.println("Valid");
} else {
    System.out.println("Invalid");
}
```

B)  
```java
if (three)
    System.out.println("Valid");
else
    System.out.println("Invalid");
```

C) Both are fine
D) Both are lacking
Which is best?

A) if (prev_type==type&&type!=1&&type!=2) {
B) if (prev_type == type && type != 1 && type != 2) {
C) if ((prev_type == type) && (type != 1) && (type != 2)) {

D) All are fine
E) All are lacking
What is wrong with this?

```java
if (c.equals("*") ||
    c.equals("/") ||
    c.equals("+") ||
    c.equals("-"))) {
```
If statements

- What is wrong with:

```java
if (winner == true) {
    aFunctionCall(arg1, arg2);
}
```
Which is better?

A
if (count != 3) {
  return false;
}
return true;

B
if (count == 3) {
  return true;
}
return false;

C) Both are fine

D) Both are lacking
How can this be improved?

```java
if (func1(arg1, arg2)) {
    return true;
} else if(func2(arg1, arg2)) {
    return true;
}

if (func1(...)) || func2(...)) {
    return true;
}

if (func1) {
    if (x != null) & x . func1())
```
switch (type) {

    // checks if the type is an open parenthesis
    case "(" :

        // comment relating to the condition
        if (someCondition) {

            // comment explaining why this return value
            return false;
        }

        if (anotherCondition) {

            // another explanatory comment
            return aFunctionCall(arg1, arg2, arg3);
        }

    // comment relating to final return value possibility
    else return (anotherFunctionCall(anotherArg1));

    // if type is a closing parenthesis
    case ")":

        // comment relating to this different condition
        if (aDifferentCondition) {

            // explanation of this return value
            return true;
        }

        if (anotherConditionToConsider) {

            // comment relating to this return value
            return aDifferentFunctionCall(arg1, arg2, arg3);
        }

    // checks if the next value in the input is an operator and returns accordingly
    else return aValue;
}
switch (type) {

    // checks if the type is an open parenthesis
    case "(" :
        // comment relating to the condition
        if (someCondition) {
            // comment explaining why this return value
            return false;
        }
        if (anotherCondition) {
            // another explanatory comment
            return aFunctionCall(arg1, arg2, arg3);
        }
        // comment relating to final return value possibility
        else return (anotherFunctionCall(anotherArg1));

    // if type is a closing parenthesis
    case ")" :
        // comment relating to this different condition
        if (aDifferentCondition) {
            // explanation of this return value
            return true;
        }
        if (anotherConditionToConsider) {

            // comment relating to this return value
            return aDifferentFunctionCall(arg1, arg2, arg3);
        }
        // checks if the next value in the input is an operator and returns accordingly
        else return aValue;

8-16% blank lines is “optimal”
To do for Tuesday

- Read Ch. 22 of Code Complete (Testing).
- Learn JUnit 4 (I’ll provide an example video)
- Write tests for the class that you wrote.
  - Feel free to fix bugs in your code, but I’m more concerned with writing a solid test suite than having your code bug free yet.

(details to be posted in Piazza on how to submit.)