Lab 3 Comments

- name directories and files/programs as directed
  - no spaces!
  - case too!
- include your name and a description of the program in comments at the beginning of every program
- style
  - autoformat
  - split/wrap long lines
  - use blank lines to group and organize
  - variable names should start with lowercase letters

Lab 3 Comments

- only create one `Scanner` per program, even if you read lots of things

```
Scanner input = new Scanner();
String line1 = input.nextLine();
String line2 = input.nextLine();
…
```

- equality test
  - use == for only primitive types (`int`, etc)
  - use `s1.equals(s2)` for `String`
- you can use literals with `String` functions
  - e.g. `if ( beverage.equals(“juice”) ) { … }`

Lab 3 Comments

- using things not (yet) covered in class e.g. `Random` instead of `Math.random()`, subroutines, arrays –
  - is it not necessary to “do research” beyond the assigned reading and material covered in class
    - use the class resources first, as the information you need will be better organized for our purposes
    - if you are going beyond, do it because you are curious and interested in learning more, not because you don’t know how to solve the problem with what has been covered in class
  - if a TF or someone else is suggesting the thing, tell them that hasn’t been covered yet – you must understand help you receive, not just write down what someone else told you

Lab 3 Comments

- roulette
  - watch out for off-by-one problems – double-check > vs >=, < vs <=, random number generation (*36 or *37)

Lab 3 Comments

- elegance – roulette
  - two main structural options for the conditionals
    - each type of bet is separate consideration → a series of "do or not do" choices (`if` without `else` or `else if`)
    - low/high, red/black, first/second/third dozen are mutually exclusive variations of the same kind of criterion → three `ifs`, each with several cases
      - don't forget to consider whether "do nothing" is an option – it is, as 0 is not included in any of these bets → final `else if` rather than final `else`
  - the following `ifs` are the same logically
    - both are valid but in this case the nested `ifs` can be easier to understand (and get right)

```
if ( a && b ) {
  …
} else if ( a && !b ) {
  …
}
```

```
if ( a ) {
  if ( b ) {
    …
  } else {
    …
  }
}
```

Lab 3 Comments

- elegance – roulette
  - it is cleaner conceptually to separate different steps
    - e.g. determining the pocket color is separate from determining whether a red or black bet has been won → separate `ifs`

  ...but sometimes not repeating code is better
    - the two `ifs` would be very similar since the bet type is based on the pocket color – so combining is reasonable, though there may be extra hoops to jump through to keep the order that the lines of output are produced correct (low/high bets are considered before red/black bets)
  - the goal of the program is to print out the pocket color and the bets that would be won
    - it is simplest to just print the desired output in the `if` case where you determine the color/bet rather than storing a value so you can have one `System.out.println()` at the end
    - go with simple unless there's a more compelling reason to do otherwise

Lab 3 Comments

- elegance – drink order
  - since the kind of juice, milk, etc depends on the type of beverage, this naturally gives rise to nested `ifs` where the top level has a case for each kind of beverage (water, juice, milk, …) and an `if` inside each case deals with the variety (apple, orange, etc)
  - coffee and tea are similar
    - simplest is to have them as separate cases because they are different beverages
    - the simple solution is fine here!
    - in general, start with simple, but then consider how to avoid repeating code

Lab 3 Comments

- elegance – drink order
  - the goal of the program is to print out the drink order
    - it is simplest to just print the desired output in the `if` case where you determine the full order rather than storing a value so you can have one `System.out.println()` at the end
    - this is fine for the required version of the program – go with simple unless there's a compelling reason otherwise
    - to deal with the size, you don't want to repeat the code to get the size from the user for each kind of beverage, so you can't print the drink order in each case
      - then the solution is to introduce a variable containing whatever is different about each order – set the variable accordingly in each case, and print it (along with any common elements) at the end