CS206 Assignment 2 Style Grading Rubrics

General

10 points are allocated to fairly mechanical rules on naming/comments/indentation - these should be easy to check off. Another 15 points are allocated to more creative practices, as explained below. Consult the formatting guide for details to check for under each category

Print student programs from Emacs, via “postscript print buffer” menu option. Editors and OS mess with tabs and newlines. Students programs need to check out on our Linux server and under Emacs.

Code formatting (**10 points total**)

1. Naming Conventions: **3 points**
    a. if any of the rules are violated
2. Whitespace: **1 point**
    a. inconsistent spacing (excessively) - - if just one place, point it out but don’t take off
3. Comments: **5 points**
    a. File header missing or malformatted
    b. Uncommented instance variables
    c. Uncommented methods (getters and setters can have no comments, when appropriately named)
    d. Method comments that do not conform to javadoc style
    e. Uncommented complex blocks of code
    f. Unhelpful comments
4. Indentation: **1 point**
    a. inconsistent indentation (excessively) - if just one single line, point it out but don’t take off

Design principles (**15 points total**)

206 Assignment 2 (ArrayList and inheritance)

1. private Instance variables and getters **1 point**
    a. Any non-private instance variables, including missing modifier
    b. Missing getters, even if not used
2. `public static final` constants instead of integer/double literals - any literal that has reason to be changed later should be a constant **1 point**
    a. Cases noted
        i. Using `"00000"` directly in code
        ii. Using `[0]`, `[1]`, `[2]` … directly in code after calling split
3. Constructor must initialize all instance variables **2 point**
    a. Check `Place, LocatedPlace` and `PopulatedPlace` constructors
    b. `LocatedPlace` and `PopulatedPlace` constructors must call `super`

appropriately

4. Reasonable designs for `Place`, `LocatedPlace`, `PopulatedPlace`, `LoopupZip` and no additional classes (besides `Main` of course) **5 points**
    a. `Place` has zipcode, town and state instance variables (as `String`) and no additional. Has `toString` overridden
    b. `LocatedPlace` has latitude and longitude instance variables as `double`, not `String` and no additional. `toString` appropriately overridden. Preferrably by calling `super.toString()` first (don't take off though, just point it out)
    c. `PopulatedPlace` has population instance variable as `int`, not `String`, and no additional. `toString` appropriately overridden. Preferrably by calling `super.toString()` first (don't take off though, just point it out)
    d. `LookupZip` doesn't have instance variables (constants are not instance variables and they should have them!) and holds the methods `parseLine` (if exists), `readZipCodes` and `lookupZip`
5. Method designs and data weaving **3 point**
    a. `parseLine`, `readZipCodes` and `lookupZip` should have reasonable designs - any abuse/overcall/redundant use gets -1:
        i. It is acceptable to not have a `parseLine` and merge the functionality into `readZipCodes` directly. Another approach is to write two different versions of `parseLine`, one for each file. `parseLine` (if there is one) should NOT have a loop
        ii. `readZipCodes` should process both files
            1. Both files are read only once
            2. Creates and returns the final ArrayList
        iii. `lookupZip` is called in a while loop in `main`, once per lookup/user input
            1. Scanner for user input is created once outside of the loop, not over and over again. This breaks redirection.
6. Only one correctly-sized ArrayList of `Place` used and created only once **3 points**
    a. An `ArrayList<Place>` of the appropriate size is created only once after uszipcodes.csv is read. It holds either `Place` or `PopulatedPlace` objects.
    b. When reading ziplocs.csv, replace `Place` with `LocatedPlace` or update `PopulatedPlace` objects in ArrayList with setters
    c. Any additional data structure -1
        i. This includes creating ArrayList in a loop over and over again