CS 1316 Programming Style Guidelines

Programming guidelines are conventions that programmers agree to follow so that their code style is consistent. Many software development groups and companies will adopt a coding guideline so that code produced by new employees matches the style of their existing code-base. All programmers are expected to follow the coding style guidelines, even though the style of the code has no real impact on how the code runs. The benefit is that when a programmer has to work on “strange” code, he already knows the conventions that the code will follow, and does not have to re-learn a different style or puzzle out confusing code. Each different group or company may have different (arbitrary) guidelines. These are the guidelines for CS 1316. You are expected to adhere to the following code style guidelines for CS 1316. Violations will result in (minor) penalties to your grade.

Naming:

- All regular variables and method names start with lowercase letters. Multi-word names use “Camal Case”, where the first letter of new words is capitalized. [A few special variables, such as constants are fully capitalized.]
- Class names start with capital letters. Class names are never plurals. If you want more than one instance of a class, you use an array or a list of class objects. Class names are typically nouns, and not verbs (although in some cases multi-word class names will contain verbs).
- Method names should describe verbs – what the object knows how to do.
- Accessors are typically named “set-” and “get-” the name of the field, such as “setAge(int a)” or “getAge()”.
- Variable, Method and Class names should be meaningful and describe their function or the data they reference. Exception: It is permissible to use single letter variable names for holding temporary data within a small function or block of code as long as the programmer can see the variable definition and all uses in a single screen. (However, the letter used should have some relation to the variable's function. l/w/h for length/width/height, x/y/z for coordinates, r/g/b for red/green/blue color values, i/j/k for counters, s for a string, etc)

Formatting of Blocks & Brackets

When starting a block of code with a bracket, you may place the bracket in one of two possible locations: First, it may be located directly after the statement that begins the block, separated by a space. Second, it may be located directly under the first letter of the statement that begins the block on the next line.

Regardless of where the beginning bracket is located, the bracket that ends the block must be indented so that it falls directly under the first letter of the statement that began the block. (See examples below.) The ending bracket must be on it's own line (see the single exception below).

Small Block Exception: When your block of code contains a single line, it is permissible to have both the starting bracket, the line of code, and ending bracket on the same line. The beginning bracket must be aligned with the first line of the statement that began the block.

Required Comments, Indenting & Visual Spacing:

All classes and non-trivial methods should have a comment located either directly before the header or directly after the header that indicates the purpose of the class/method. Comments that are more than 2 lines long MUST use a running comment that begins with a (/*) and ends with a (*/). Shorter comments may use a end-of-line comment that begins with a (//) and continues to the end of the line.

Complicated code or "magic numbers" should be labeled with an in-line comment (either on the line directly above the code, or as an "end-of-line" comment after the code).

End of block brackets that are visually separated from their corresponding beginning of block brackets must be labeled with an end of line comment. Visually separated means that you can not directly see the beginning of block bracket that corresponds to the end-of-block bracket without scrolling. Typically about 20 lines. Comments should start with the word "end" and then describe what the bracket is ending (for example, end class BLAH, end for p in pixels loop, end while button down loop, etc)

Code inside of blocks (for/while loops, if statements, class definitions) must be indented at least three spaces. All code inside a functional block must be indented the same amount.

White space (blank lines) should be used to separate each method definition from previous code. Additionally, when code begins a new operation or block, a blank line should be used to provide visual separation that indicates that something new is happening.

Example Class:

```
/*
      This class models a Person and implements a few behaviors
      that our modeled person can perform such as aging or eating
      food.
*/
public class Person {
  private int age;                // in years
  private Bool isHungry;

  public void Person( int a, Bool hungry)
  {
    this.age = a;
    this.isHungry = hungry;
  }

  public int getAge()
  { return( this.age); }

  public int setAge(int a)
  { this.age = a}

  // This method increments the person's age by one year, and makes them hungry!
  public void age()
  {
    this.age = this.age+1;
    this.isHungry = True;          //After a year, you get hungry!
  }

  public void eat()
  {
    this.isHungry = False;
    System.out.println("I'm full now, thanks!")
  }


} // end class Person
```