

## CS 302 Commenting Guide

# Program Commenting Guide

[File](#) | [Class](#) | [Method](#) | [Variables](#) | [Others](#)    **Related Pages:** [Style Guide](#) | [Example](#)

**Program comments are intended for programmers**, and they are ignored by the compiler. We want you to develop good commenting style so we'll require the following:

- [File Header - Main Class](#)
- [File Header - Other Classes](#)
- [Class Header](#)
- [Method Header](#)
- [Variable Declarations](#)
- [Other Method Comments](#)

## File Headers

**Requirement:** The source file for the **MAIN CLASS** must have a main file header comment located at the beginning of the file containing the following:

```

////////////////////////////////////
//                               ALL STUDENTS COMPLETE THESE SECTIONS
// Title:                        (program's title)
// Files:                        (list of source files)
// Semester:                     (course) Spring 2016
//
// Author:                       (your name)
// Email:                        (your email address)
// CS Login:                     (your login name)
// Lecturer's Name:             (name of your lecturer)
// Lab Section:                  (your lab section number)
//
//////////////////////////////////// PAIR PROGRAMMERS COMPLETE THIS SECTION //////////////////////////////////
//
//                               CHECK ASSIGNMENT PAGE TO see IF PAIR-PROGRAMMING IS ALLOWED
//                               If pair programming is allowed:
//                               1. Read PAIR-PROGRAMMING policy (in cs302 policy)
//                               2. choose a partner wisely
//                               3. REGISTER THE TEAM BEFORE YOU WORK TOGETHER
//                               a. one partner creates the team
//                               b. the other partner must join the team
//                               4. complete this section for each program file.
//
// Pair Partner:                 (name of your pair programming partner)
// Email:                        (email address of your programming partner)
// CS Login:                     (partner's login name)
// Lecturer's Name:             (name of your partner's lecturer)
// Lab Section:                  (your partner's lab section number)
//
//////////////////////////////////// STUDENTS WHO GET HELP FROM OTHER THAN THEIR PARTNER //////////////////////////////////
//                               must fully acknowledge and credit those sources of help.
//                               Instructors and TAs do not have to be credited here,
//                               but tutors, roommates, relatives, strangers, etc do.
//
// Persons:                      Identify persons by name, relationship to you, and email.
//                               Describe in detail the the ideas and help they provided.
//
// Online sources:               avoid web searches to solve your problems, but if you do

```

```
//          search, be sure to include Web URLs and description of
//          of any information you find.
//          ////////////////////////////////// 80 columns wide //////////////////////////////////
```

**Requirement:** Source files for **OTHER CLASSES** must have a **file header comment** located at the beginning of the file containing the following (see [example](#)):

```
////////////////////////////////////
//          ALL STUDENTS COMPLETE THESE SECTIONS
// Main Class File:  (name of main application class)
// File:            (name of this class's file)
// Semester:        (course) Spring 2016
//
// Author:          (your name and email address)
// CS Login:        (your login name)
// Lecturer's Name: (name of your lecturer)
// Lab Section:     (your lab section number)
//
//          ////////////////////////////////// PAIR PROGRAMMERS COMPLETE THIS SECTION //////////////////////////////////
//
// Pair Partner:    (name of your pair programming partner)
// Email:           (email address of your programming partner)
// CS Login:        (partner's login name)
// Lecturer's Name: (name of your partner's lecturer)
// Lab Section:     (your partner's lab section number)
//
//          ////////////////////////////////// STUDENTS WHO GET HELP FROM OTHER THAN THEIR PARTNER //////////////////////////////////
//          fully acknowledge and credit all sources of help,
//          other than Instructors and TAs.
//
// Persons:         Identify persons by name, relationship to you, and email.
//                  Describe in detail the the ideas and help they provided.
//
// Online sources:  avoid web searches to solve your problems, but if you do
//                  search, be sure to include Web URLs and description of
//                  of any information you find.
//          ////////////////////////////////// 80 columns wide //////////////////////////////////
```

## Class Header

**Requirement:** Each class must have a **header comment** located immediately before the class declaration containing the following (see [example](#)):

```
/**
 * (Write a succinct description of this class here. You should avoid
 * wordiness and redundancy. If necessary, additional paragraphs should
 * be preceded by <p>, the html tag for a new paragraph.)
 *
 * <p>Bugs: (a list of bugs and other problems)
 *
 * @author (your name)
 */
```

This form for the class header is the standard for documenting java programs. It is referred to as a "javadoc" comment because it is used by the **javadoc** program to automatically generate documentation in HTML format. You will not be required to run **javadoc** on your programs but it is a useful capability to know about. For more information about **javadoc**, you should visit Sun's website: [How to Write Doc Comments for the Javadoc Tool](#)

## Method Header

**Requirement:** Each method must have a header comment located immediately before the method declaration (see [example](#)). Include the information below but leave out parameter and/or return-value comments if the method has none. The method description must state preconditions and postconditions. Preconditions are requirements that must be met before entering the method, and postconditions are results from executing the method. Do not state the obvious such as "an object needs to be created before calling this instance method". Instead specify things that may be overlooked or unexpected.

```
/**
 * (Write a succinct description of this method here.  If necessary,
 * additional paragraphs should be preceded by <p>, the html tag for
 * a new paragraph.)
 *
 * @param (parameter name) (Describe the first parameter here)
 * @param (parameter name) (Do the same for each additional parameter)
 * @return (description of the return value)
 */
```

This form for the class header is the standard for documenting java programs. It is referred to as a "javadoc" comment because it is used by the **javadoc** program to automatically generate documentation in HTML format. You will not be required to run **javadoc** on your programs but it is a useful capability to know about. For more information about **javadoc**, you should visit Sun's website: [How to Write Doc Comments for the Javadoc Tool](#)

## Variable Declarations

**Requirement:** Variable declarations must be commented briefly describing their use. This includes object and primitive variable declarations of:

- **class** data members and constants
- **instance** data members and constants (see [example](#))
- **local** variables and constants (see [example](#))

**Requirement:** Primitive variable declarations must also specify their range of values if it is a subset of the full range for their data type. (see [example](#)).

Temporary variables and loop counters do not need to be commented.

## Other Comments

**Requirement:** Use comments within the body of methods to:

- highlight the major steps of your algorithm
- explain long calculations or conditions
- clarify convoluted or unusual code
- mark locations where you suspect a bug may exist
- mark locations where improvements or enhancements are planned

**Do not use comments in methods to explain things that are obvious to programmers.** For example, it is not useful to provide comments such as "this assignment statement assigns 22 to the variable x". Instead use comments to point out things that are not immediately evident or to highlight sections of code.

© 2000-2010 Jim Skrentny,  
 © 2001-2005 Deb Deppeler,  
 © 2007 Beck Hasti