Chapter 13

Programming style

Lesson page 13-1. Good Programming Practices

Question 1. True.

Question 2. True.

Question 3. True.

Question 4. True.

Question 5. True.

Question 6. True.

Question 7. The `javadoc` program will extract the specification of classes, methods, etc. from a collection of source files and create `html` files from them. These files describe the public and protected classes, inner classes, interfaces, constructors, methods, and fields that are defined in the source files.
The Java API specifications are generated using `javadoc`.

Lesson page 13-2. Naming conventions

Question 1. True.

Question 2. False.

Question 3. True.

Activity 13-2-1 Conventions for naming parameters

Question 4. Use short names for parameters, provided the specification of the method suitably defines the parameters.

Question 5. The specification of the method gives full details on the meanings of the parameters, and method bodies are usually quite short —less than a page long— so one can see any reference to a parameter and the method spec at the same time.

Question 6. The body of the method looks cluttered and harder to read.

Question 7. Perhaps `adj`, `opp`, and `hyp`.

Activity 13-2-2 Conventions for naming local variables

Question 8. Use a short name for a local variable, if the variable is properly specified in a comment near its declaration.

Question 9. Short names for local variables work because within a method body, a local-variable declaration and uses of the variable are quite close together.

Question 10. Here is the program with unnecessary comments crossed out.

```
// Used for demonstration only.
public class Test {
    //-This-is-method-main-
    public static void main(String args[]) {
        int a= 4;          // the number of ants
        int numOfBats= 1; // -the-number-of-bats-  (optional)
        int msqu= 500;     // the number of mosquitoes
        //-Print-out-the-ratio-of-bats-to-ants.-
            System.out.println(
                "Ratio of bats to ants is: "
                + numOfBats + ":" + msqu + ".");
    }
}
```

Activity 13-2-3 Conventions for naming instance variables and class variables

Question 11. To help the reader, use longer names that will help reduce the need to look at the definition.

Question 12. False. What does "S" stand for?

Question 13. When the description is too long to be a variable name.

Activity 13-2-4 Conventions for naming constants

Question 14. A constant is a variable that is declared with qualifier **final**, which indicates that the variable cannot be changed.

Question 15. Use all capital letters for constants. Separate words within a name using an underline character "_".

Question 16. Here's the class with the constant:

```
public class Fondue {
    public static final int CHOCOLATE_CHIP_BAG_WEIGHT= 250;
}
```

Activity 13-2-5 Review of conventions for variable names

Question 17. `NUM_STAPLES`.

Question 18. `height`. The name `x` might be okay, if it represented an x-coordinate, but `partNo` is unclear: `No` might mean "number", or it might indicate the word "no".

Question 19. `height`, `x`, and `partNo` are okay, provided that their real meaning is suitably defined in a specification or comment on a local variable.

Activity 13-2-6 Conventions for naming methods

Question 20. (1) A procedure name can be a command to do something. (2) A procedure name can be the name of an algorithm.

Question 21. (1) A function name can be a name for the result. (2) The name of a boolean function can be an abbreviation for the true-false statement that is its specification. (3) A function name may be the name of the algorithm used to compute the result.

Question 22. Procedures: `processInfo` (calls: `readInfo`, `sortInfo`, `disruptInfo` and `displayInfo`), `sortInfo`, `manipulateInfo`, `displayInfo`. Functions: `readInfo`.

Activity 13-2-7 Conventions for naming classes

Question 23. Use a name that is long enough to give some indication of what the class is for. (Or: use a noun phrase for the name —a list of adjectives followed by a noun— that describes an instance of the class.) Class names begin with capital letters.

Question 24. CityBlock, Building, House, CommercialBuilding, GovernmentBuilding, Person, Bike, Car. Note that these classes are singular. For example, "Building" and not "Buildings" because each building is a separate item, and will have a different amount of traffic.

Lesson page 13-3. Conventions for indentation

Activity 13-3-1 Why indent?

Question 1. (1) In a sequence of constructs, indent them all the same amount. (2) If a construct requires more than one line, indent its subconstructs.

Question 2. The properly indented code appears below. Now that we can actually read the program, we find that it has mistakes. For example, the first question goes unanswered, and the output should be flushed after printing out "Your question? ...".

```
// ask a Prof. questions until we're tired
private static void askQuestions (Professor prof)
    throws IOException
{
    final String LAST_QUESTION = "bye"; // ends input
    System.out.print("Your question? ('bye' for last) ");
    String question= in.readLine();
    while (!question.equals(LAST_QUESTION)) {
        System.out.print("Your question? ('bye' for last) ");
        String question= in.readLine();
        Prof.answer();
    }
}
```

Activity 13-3-2 Points to watch out for when indenting

Question 3. There is no answer to go with this non-question.

Lesson page 13-4. Guidelines for writing methods

Activity 13-4-1 The specification as a logical firewall

Question 1. The most important beneficiary of writing the specification before the method is the author of the code.

Question 2. `// = "a, b, and c are equal (using ==)"`

Question 3. The body of match is: **return** `a == b && b == c;`

Activity 13-4-2 Consistency of method body and method specification

Activity 13-4-3 The statement-comment

Question 4. Abstraction is the act or process of removing from consideration one or more qualities of a complex object in order to attend to others. For example, abstraction allows the programmer to deal with *what* a method does without having to deal with *how* the method does it every time the programmer wants to use the method.

Question 5. A statement-comment is a comment in a program, usually written in English, that describes precisely what a piece of code does; the piece of code is indented underneath it.

Activity 13-4-4 Discussion of the statement-comment

Question 6. Statement-comments (1) provide for different levels of abstraction, (2) make the design and structure visible, and (3) help the reader browse

the program.

Activity 13-4-5 Why indent?

Question 7. Indenting the implementation of a statement comment makes it possible to unambiguously find the end of that implementation; without such indenting, there is room for misinterpretation.

Question 8. To find the end of a refinement of a statement comment, just begin at the column in which the statement-comment begins and look down until nonblank material is found.

Question 9. We don't provide an answer to this question.

Lesson page 13-5. Describing variables

Activity 13-5-1 Describing instance variables

Question 1. Group variable declarations by logical togetherness (relationship). This is because one comment can usually be used to describe all the variables and their relationship and because one usually wants to understand all the logically related variables at the same time. For example, an array `b` and variable `n` might be related by "`b[0..n-1]` contains the temperatures that have been recorded so far."

Question 2. Here are the answers:

```
private int numShoppers; // number of shoppers in mall

// total cost all things bought by all shoppers
private int sumShoppers;

// number of strollers/carriages in mall
// OR
// number of people currently walking in mall
private int numStrollers;

private int numStairs; // number of staircases in mall

private int numStanding; // number of shoppers currently standing
```

Question 3. Parameters are described in the specification of the method, which appears in the comment that precedes the method.

Activity 13-5-2 The placement of local-variable declarations

Question 4. The "need to know" policy says to place a local variable declaration as close as possible to the first use of the variable.

Question 5. Place a local variable at the beginning of a method if that is the only way to have its scope include all places where it is going to be used.

Question 6. We can't do this exercise for you. Doing this exercise is a good way to become conscious of your own programming habits, so that you can begin improving them.