Answers to Chapter Review Problems

1. A good modular design allows alterations to focus on only the pertinent sections of the software. The use of local variables reduces side effects and thus reduces the chance of “corrections” introducing more errors, etc.

2. Evolutionary prototyping is the process of developing the final software system by enhancing the prototype—that is, the prototype evolves into the final product.

3. It results in a subjective discipline in which theories are debated but few precise, definitive conclusions can be reached.

4. No. The complexity of a software system is enhanced by the interaction between its parts and therefore would be greater than merely the sum of the complexities of its parts. (This is why beginning programmers often do not comprehend the problems faced by software engineers. They tend to see large software systems as merely bigger versions of the small programs they have written.) (This question is meant to invite the more mathematically inclined students to consider the mathematical properties that software metrics would have.)

5. Probably not. The structures of the underlying systems would probably be different. Feature X may fit conveniently into the system that already contains Y, but adding feature Y to a system already containing X may be problematic. (Here, again, this question is meant to invite the more mathematically inclined students to consider the mathematical properties that software metrics would have.)

6. Other fields of engineering tend to be based on exact science. Mechanical engineering, for example, has a strong foundation in physics.

7. There are several answers. Here are two that come to mind.
   a. The lack of flexibility means that mistakes can be very expensive.
   b. The waterfall model provides a well-defined process in which progress can be measured.

8. Open-source development is an example of bottom-up design in that the final system is built by expanding a simpler system. In fact, the functionality of the final system is usually not known at the beginning of the project. Instead, it evolves according to the desires of the developers.

9. Using a constant clarifies the role of the value. In addition, constants provide a single point of change. They are declared once and can be used in multiple places. Should a decision be made to change the value then only the declaration would have to be changed.

10. Coupling refers to the connections between modules, cohesion refers to the connectivity within a module. On the surface, one would like to minimize coupling (because that leads to independent modules that can be maintained individually) and maximize cohesion (because that leads to modules whose activities can be more easily understood).

11. The answer, of course, depends on the object that the student selects. As a general rule an entire device is usually only logically cohesive but individual components tend to be functionally cohesive. An entire automobile is an example of logical cohesion, whereas a component such as a single seat or the steering wheel is more functionally cohesive.
12. Although disliked for other reasons, the control coupling obtained by a simple goto statement is less complex than that of a subprogram call. In particular, the transfer is in only one direction. On the other hand, the careless use of goto statements can introduce a tangled mess that may never be understood.

13. Passing parameters by reference would probably be considered the more “complex” form of data coupling because it allows two-way communication between the program units whereas passing parameters by value does not allow the function to communicate back to the calling environment.

14. One problem would be that if the structure of one of the global elements was modified, it would be difficult to identify the portions of the program that access that element so they could be adjusted.

15. If an instance variable is private, no data coupling based on that variable can occur between objects. Therefore, declaring instance variables as private will reduce the potential for data coupling.

16. The problem of updating shared data as discussed in Section 5.6 of the previous chapter.

17. a. W if W called it, X if X called it
   b. Z.
   c. no
   d. yes, through the common use of data element a.
   e. element a.
   f. data coupling through data element a.

18. Answers will vary. The point is for students to understand structure charts and to think about how their modular designs would hold up against future modifications in the system. (Did they, for instance, minimize data coupling?)

19. How about something like this?

![Structure Chart](image)

20.

![UML Diagram](image)

21. UML stands for Unified Modeling Language. It is a standard for representing object-oriented designs. The word *modeling* indicates that it is used to define models for software. (The intent here is to have the student reflect on what constitutes a model and how it relates to software development. A model is often a representation of something yet to be built. For example, structural engineers may develop a scaled model of a bridge that will be later constructed. Thus in the context of software engineering, a model enables software developers to have a common understanding of how the software will be built.)

22. Answers will vary. The library should be represented as a large rectangle and the patron should be represented as a stick figure outside the rectangle. Two important uses that may be depicted are “check out book” and “return book.” Others might include “browse” and “access catalogue.”
23. Answers will vary. Perhaps the simplest would should a "solicit payment" message being sent to the customer followed by a "make payment" message being returned. On the other hand, some students may get creative and follow a longer sequence showing the customer refusing to pay and perhaps having his or her service cut off. The point is for the students to indicate that they understand collaboration diagrams.

24. Answers will vary. The important thing is that the students focus on the flow of data. One answer would be a diagram showing an "inventory record" being retrieved from the “inventory database” and merging with a “sales record” to form an “updated inventory record,” which flows back to the “inventory database.”

25. A class diagram represents relationships between classes, whereas a sequence diagram represents interaction between objects.

26. In a one-to-many relationship, each occurrence of one of the entities may be related to several occurrences of the other entity, but not vice-versa. In a many-to-many relationship, each occurrence of either entity may be related to several occurrences of the other entity.

27. There are many possible answers. An example of a one-to-many relationship is found between primary residences and individuals. (A residence may be the primary residence of several individuals but, by definition, each individual can have only one primary residence.) An example of a many-to-many relationship is found between stockholders and companies. (One person may own stock in many companies and a single company may have many stockholders.)

28. Answers will vary. The point is for the student to display an understanding of what a sequence diagram represents and the notation used. One sequence might be that the physician sends the patient a "request symptoms" message, the student responds by "reporting symptoms," and the physician replies with "prescribe action."

29.

![Diagram](image)

30. This problem is similar to problem 20, except it expresses the design with greater detail.

![Diagram](image)

31. Immediately before the bottom most arrow, insert duplicates of the second and third horizontal arrows.
32. a. X = User, Y = Manufacturer, and Z = Tool. A major clue is that tools are used by users.
   b. No
   c. No
   d. Yes
33. a. use case diagram  b. class diagram  c. sequence diagram
34. a. X  b. Z  c. No
35. The surrounding details will vary, but the core of the answer should look like this:

   ![Diagram]

36. A good way to relate this material to the previous topic of control structures back in chapters 5 and 6 is to point out that this is UML's way of representing a pretest loop. Ask your students to suggest a notation that UML could use to represent a posttest loop. (Move the "test condition" to the bottom of the interaction fragment.)

37. ![Diagram]

38. There can be a number of answers to this question. For SurgicalRecord, variables such as anesthesiologist, duration, and surgicalNotes are reasonable. For OfficeVisitRecord, variables such as prescriptions and typeOfVisit (with values brief, normal, and extended) are reasonable.

39. Careless use of inheritance results in a strong coupling between the parent and child classes that may lead to unforeseen errors later in software maintenance or may even need to be unraveled to implement future modifications.
40. One possible answer would be in the context of university schedules. Some follow the two-semester-summer-session pattern while others follow the quarter-system pattern. Another would be in television broadcasting where programs fit a variety of patterns such as 30-minute national news, 30-minute situation comedy, one-hour newsmagazine, etc.

41. By means of design patterns, software engineers hope to construct predefined building blocks from which large systems can be constructed.

42. The point here is for the student to realize that the control structures are exactly small scale design patterns. In chapters 4 and 5 we tended to use flowcharts to represent such patterns. The statement structures in a particular language are frameworks for those patterns. When a programmer writes a program, he or she customizes those frameworks to fit the specific needs of that particular program. In this sense the syntax diagrams for a programming language make up a cookbook. (Note that this is not only true for the control structures in a language. Indeed, each statement structure in a language is a framework for a small design pattern.)

43. All of them. The Pareto principle has a wide range of applications. (This could make an interesting discussion topic.)

44. Software engineers, according to the Pareto principle, expect errors to be concentrated in certain parts of a software system. Thus, they see large software systems as heterogeneous mixtures of errors.

45. Black-box testing tests the performance of software without regard for its internal construction. Glass-box testing is done with the knowledge of the system’s internal design.

46. Analogies to black-box testing would be a customer test driving a new automobile and another customer sampling a flavor of ice cream. An analogy to glass-box testing would be a health inspector evaluating the conditions under which an ice cream parlor prepared its ice cream.

47. In beta testing the tester is only allowed to test; in open-source development the "tester" is allowed to test and modify the software. Thus, beta testing is a black-box methodology whereas open-source development is a glass-box methodology.

48. Since half of the known errors were found, it’s reasonable to assume that half of all the errors were found. Thus, we conclude that there were a total of 400 errors in the system. 200 of these were found and removed. Another 50 were known but not found. These were removed also. Consequently, we would estimate that 150 errors remain in the system.

49. GOMS is a methodology for measuring the efficiency of a human-machine interface.

50. Ergonomics is the engineering discipline that deals with designing systems that harmonize with the physical capabilities of humans. Cognetics is the engineering discipline that deals with designing systems that harmonize with the mental capabilities of humans.

51. a. Due to the restrained size of the display screen on a smart phone, the use of sliders, menus, or toolbars is discouraged as they take away valuable display space needed to render the content of the application. Therefore alternatives like double touch are effective in maximizing the amount of display space that can be used for content.

b. A double touch expand on the display screen is a natural gesture to the way we think. Spreading the thumb and index finger is a gesture that visual communicates a meaning of expand (hence zoom).

52. Copyright laws are generally insufficient to prevent a competitor from independently creating a product with similar function.

53. A patent application may be denied if the software developer fails to demonstrate that the software is new, useful, or not obvious to others with similar backgrounds. A software developer will certainly be denied a patent in the event that a valid patent already exists and is held by another party.