**Discuss the ways in which inheritance promotes software reuse, saves time during program**

**development and helps prevent errors.**

**ANS:** Inheritance allows developers to create derived classes that reuse code declared already in a base class. Avoiding the duplication of common functionality between several classes by building an inheritance hierarchy to contain the classes can save developers a considerable amount of time. Placing common functionality in a single base class, rather than duplicating the code in multiple unrelated classes, helps prevent the same errors from appearing in multiple source-code files and makes debugging easier. If several classes each contain duplicate code containing an error, the software developer has to spend time correcting each source-code file with the error. However, if these classes take advantage of inheritance, and the error occurs in the common functionality of the base class, the software developer needs to modify only the base class’s

code.

**Some programmers prefer not to use protected access because they believe it breaks the encapsulation of the base class. Discuss the relative merits of using protected access vs. using**

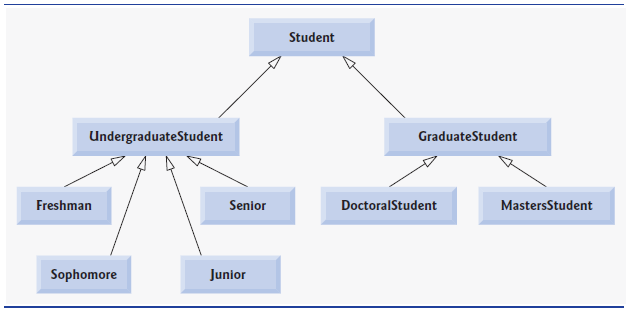
**private access in base classes.**

**ANS:** private data members are hidden in the base class and are accessible only through the public or protected member functions of the base class. Using protected access enables the derived class to manipulate the protected members without using the access functions of the base class. If the base-class members are private, the member

functions of the base class must be used to access the data. This may result in a decrease in performance due to the extra function calls, yet accessing and modifying private data in this indirect manner helps ensure that the data in the base class remains consistent.

**Draw an inheritance hierarchy for students at a university. Take Student as the base class of the hierarchy, including classes UndergraduateStudent and GraduateStudent that derive from Student. Continue to extend the hierarchy as deep (i.e., as many levels) as possible. For example, Freshman, Sophomore, Junior and Senior might derive from UndergraduateStudent, and DoctoralStudent and MastersStudent might derive from Graduate- Student. After drawing the hierarchy, discuss the relationships that exist between the classes. [*Note:* You do not need to write any code for this exercise.]**

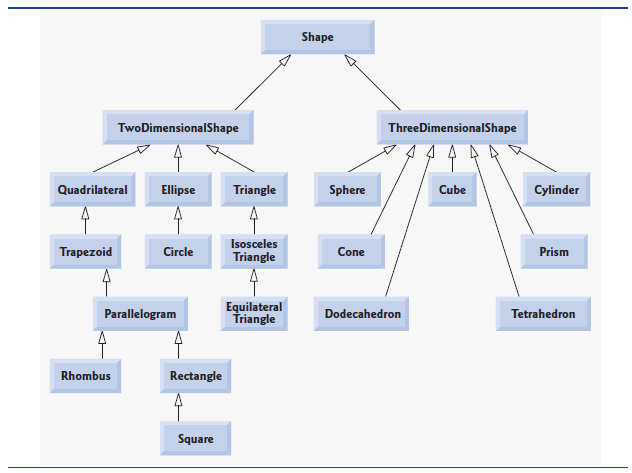
**ANS:** This hierarchy contains many “is-a” (inheritance) relationships. An UndergraduateStudent *is a* Student. A GraduateStudent *is a* Student, too. Each of the classes Freshman, Sophomore, Junior and Senior *is an* UndergraduateStudent and *is a* Student. Each of the classes DoctoralStudent and MastersStudent *is a* GraduateStudent and *is a* Student.



**Using Inheritance write down all the shapes you can think of—both two-dimensional and three-dimensional—**

**and form them into a more complete Shape hierarchy with as many levels as possible. You hierarchy should have the base class Shape from which class TwoDimensionalShape and class ThreeDimensionalShape are derived. [*Note:* You do not need to write any code for this exercise.]**

**ANS:** [*Note:* Solutions may vary.]



**Draw an inheritance hierarchy for classes Quadrilateral, Trapezoid, Parallelogram, Rectangle and Square. Use Quadrilateral as the base class of the hierarchy. Make the hierarchy as deep as possible.**

