form of menus, input screens, report screens, etc. Control classes control the sequencing of events, typically the sequences of events in the execution of a set of use case scenarios. In a system of any size you would expect to find a boundary object and a control object for each use case.

Visibility. When we talk about visibility in modelling, and in programming, we are essentially talking about the ability to limit the accessibility of certain features of the model or the program. The more features we can make inaccessible, the more control we have over isolating the effect of changes. However, we cannot make everything inaccessible. In an object-oriented system, objects must communicate with each other to carry out tasks; they can only do so by using operations they know about, i.e. operations that are part of some class's public interface. In the UML, any attribute or operation can be declared to be *public* (+), *private* (−) *or protected (#).* A class can have some operations that are public, which are there to provide services to other classes, and some that are private because they are only for the class's internal use. Programming languages interpret these visibility indicators slightly differently. The UML leaves the precise interpretation to be determined by the implementation language. However, a public operation is usually interpreted to be one that can be used by instances of any class. A private operation is usually interpreted to be one that can only be used by an instance of the owning class. A protected operation is usually interpreted to be one that can only be used by instances of the owning class or a subclass (or other descendent) of the owning class. As far as attributes are concerned, a public attribute can be accessed[1] by any object and it can be inherited; a private attribute cannot be accessed by any other object, it cannot be inherited; a protected attribute cannot be accessed by any other object, but it can be inherited.[2]

Packages and dependencies. As we move towards implementation, we are adding a great deal of extra detail to our models. The class diagram is bulging with new classes as we graft on boundary classes, control classes and collection classes.[3] Decisions need to be made about what software and hardware will be used – the software and hardware platforms. All this information has to be built into existing models and new models added to document our decisions. If the models are to be useful, they need to be organized and managed – we can't really deal (physically or intellectually)

1. *Viewed or modified.*
2. *In Java, protected visibility, for an operation or an attribute, gives public access to other classes in the same package, but is private to classes outside it.*
3. *Collection classes are described in Chapter 10.*