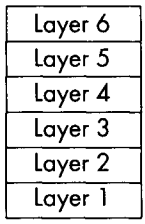

Figure 9.4 *A layered architecture*

Arranging packages in layers does not, in itself, limit dependencies, but it makes the developer aware of dependencies. It offers a logical structure to aim for. The layered approach only works if you have arranged your classes, packages and dependencies in a way that works as a layered architecture. If classes in layer 6 use classes in layer 3 or 1, or if classes in any layer use classes in the layers above them, the advantages of using a layered architecture are rapidly lost. The reason for structuring system models, and therefore ultimately the code elements generated by them, into a layered architecture is that it minimizes dependencies and the effects of dependencies. Being aware of, and carefully controlling, the dependencies between layers can make the difference between a robust system that is easy to maintain and one that is not.

Layers. Now that we understand how the layers work together, we need to think about what to put in each layer. If the layers are to be arranged in such a way that each layer only uses the services of the layer below, we are quite limited in where we place our packages. The normal arrangement is to put what is called the presentation layer near the top and the data storage layer towards the bottom, with the application layers in the middle. A simple four layer architecture might look like Figure 9.5. The presentation layer contains a package of boundary or interface classes, the application logic layer contains the control classes, the application layer contains the domain classes (entity and related classes) and the storage layer contains the database and related classes. This is a standard sort of arrangement because, generally, interface classes depend on control classes which in turn depend on application classes which depend on database classes. In a large and complex system there will be more layers and more packages in each layer.

Having decided on the layers, we have a basis for grouping our classes into packages. Figure 9.6 shows a possible arrangement which includes some implementation decisions.

We have grouped all of the interface classes into a package, 'Application UI classes', and put them in the presentation layer. If we decide to implement in Java we might use the Java Swing