**Design Classes**

The analysis model defines a set of analysis classes (Chapter 10). Each of these classes describes some element of the problem domain, focusing on aspects of the problem that are user visible. The level of abstraction of an analysis class is relatively high.

As the design model evolves, you will defi ne a set of design classes that refi ne the analysis classes by providing design detail that will enable the classes to be implemented, and implement a software infrastructure that supports the business solution. Five different types of design classes, each representing a different layer of the design architecture, can be developed [Amb01].

1. **User Interface Classes**: These classes define abstractions necessary for human-computer interaction (HCI). They often implement the HCI using metaphors and manage how users interact with the system.
2. **Business Domain Classes**: These classes refine the analysis classes related to elements of the business domain. They identify attributes and services (methods) needed to implement specific business functionalities.
3. **Process Classes**: These classes implement lower-level business abstractions that manage the behaviour and interactions between business domain classes. They are crucial for the operational flow and logic of the application.
4. **Persistent Classes**: These classes represent data stores such as databases. They manage data persistence beyond the execution of the software and encapsulate the logic for storing and retrieving data.
5. **System Classes**: These classes implement software management and control functions necessary for the system to operate within its computing environment and interact with the outside world. They handle system-level concerns like communication protocols, hardware interfaces, etc.

Each type of design class plays a specific role in translating high-level analysis models into detailed design specifications that can be implemented. They encapsulate different aspects of the system's functionality and technical implementation details, ensuring that the system is both well-structured and maintainable over time.

Example related to a hypothetical software application for managing a library: