# LBMS

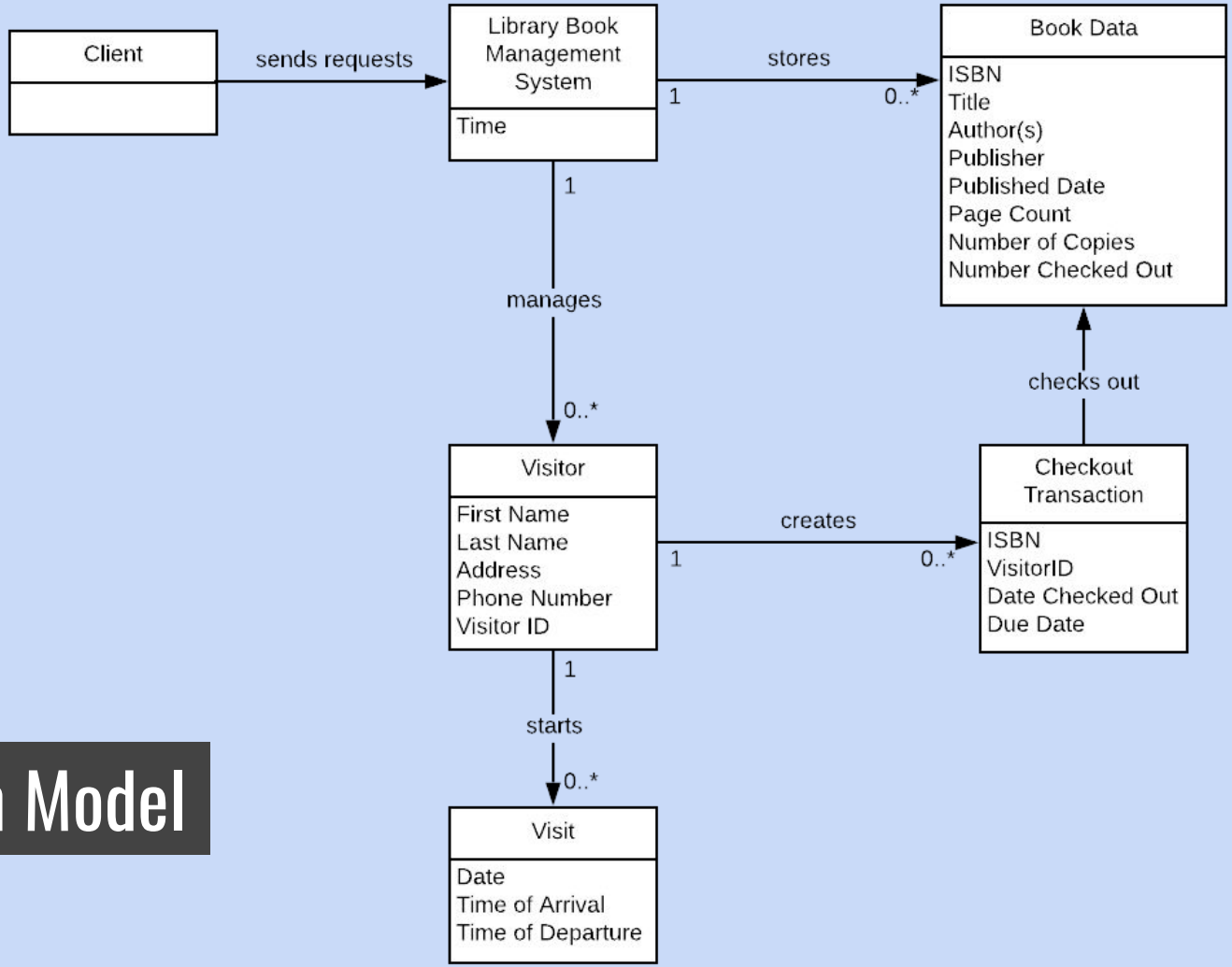
Team Banana

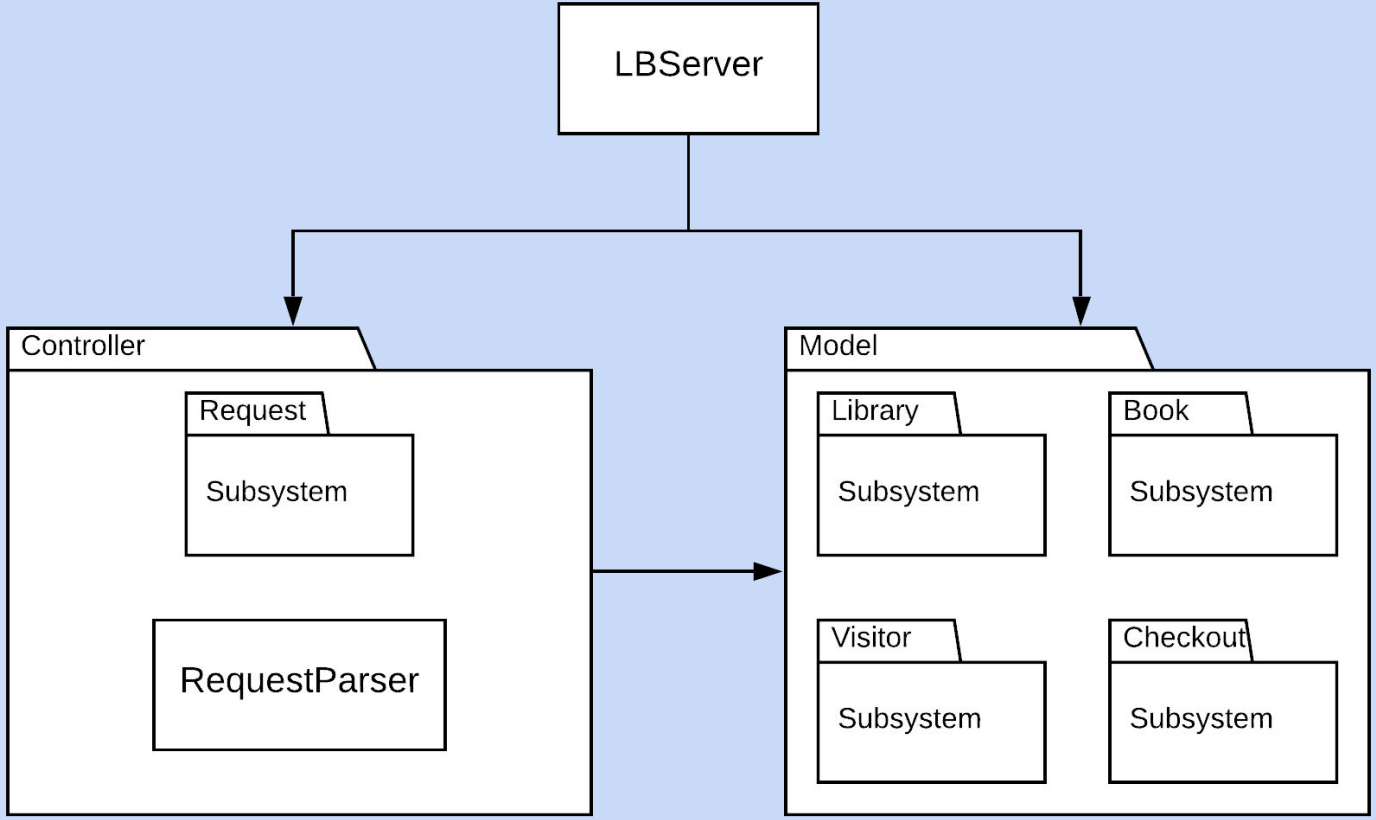
# Problem Statement/Summary/Overview

The Book Owl Library wants to automate their system for tracking books and visitors.

We were tasked with designing and implementing a system that meets their requirements. Our initial version of the system is run through the command line.

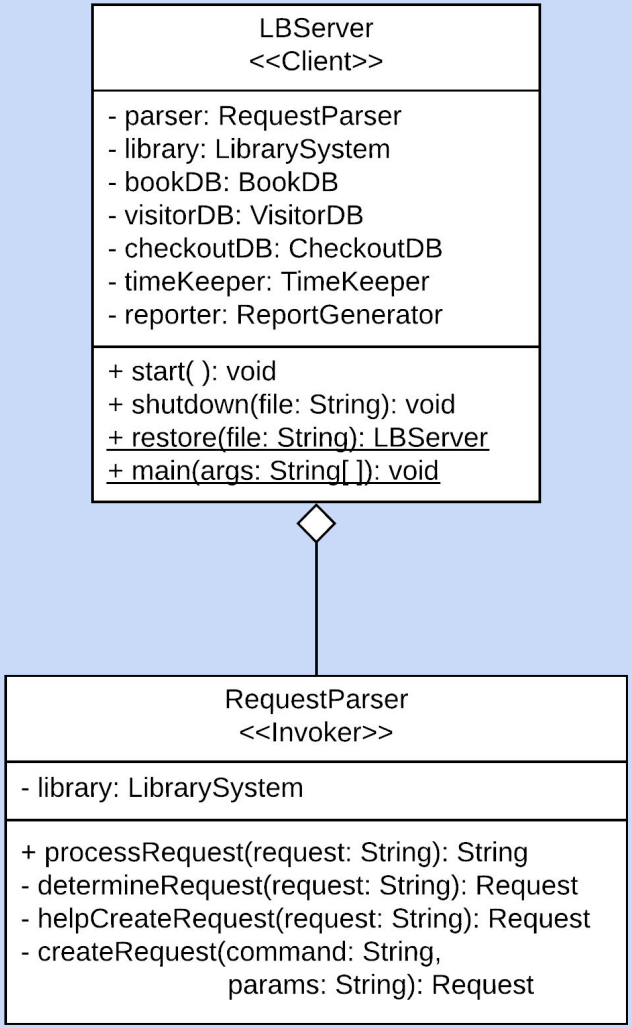
# Domain Model

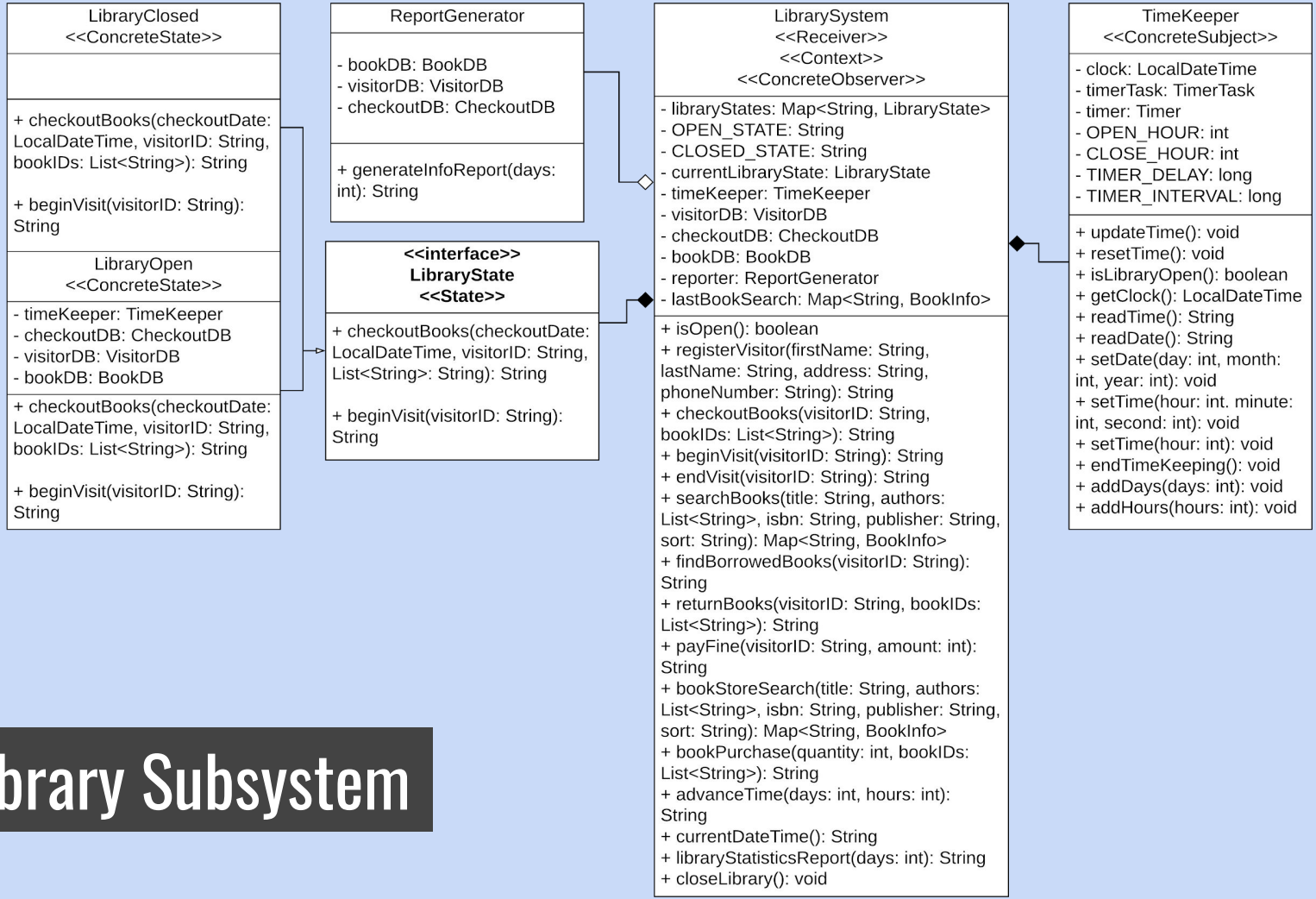




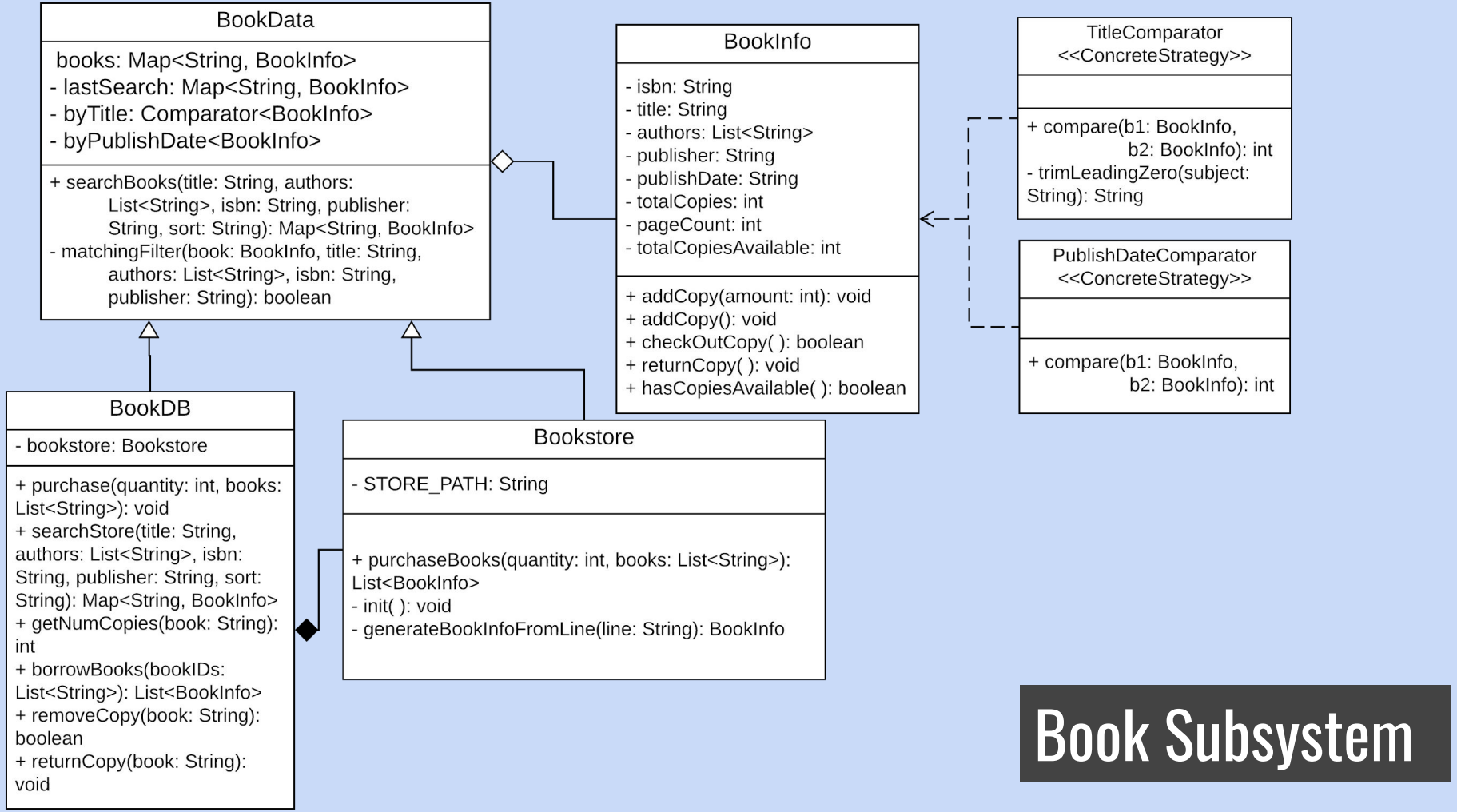

# System Architecture

# High-Level Structures

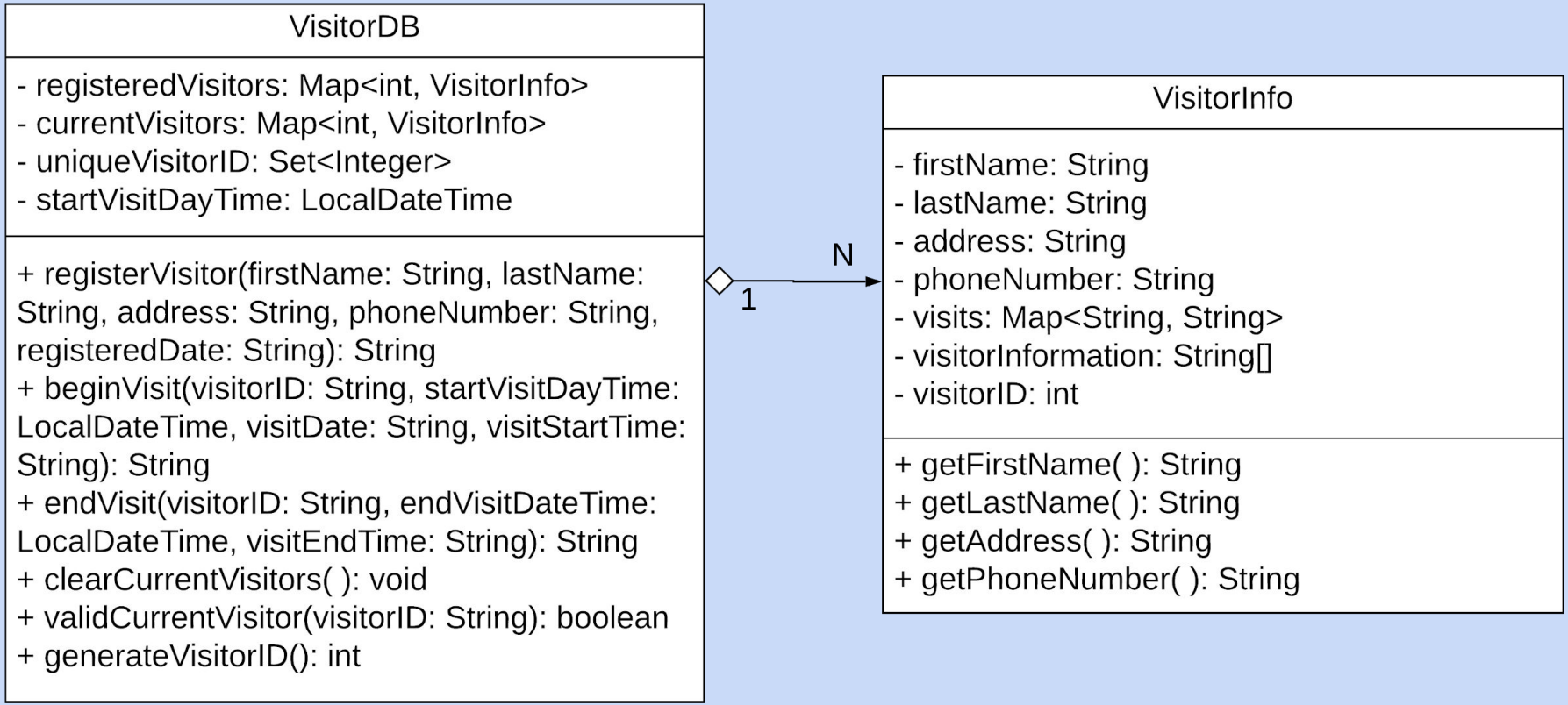




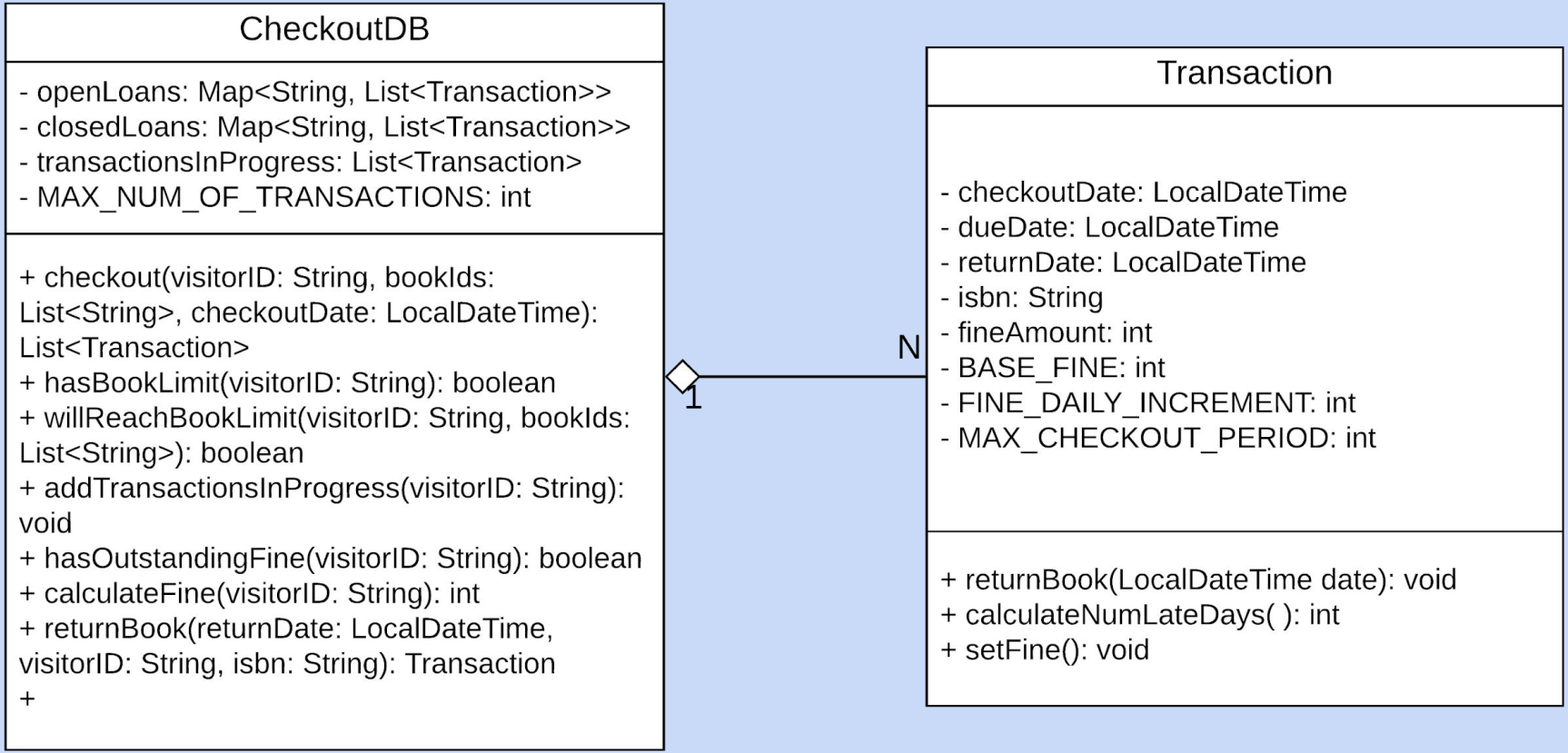

# Library Subsystem



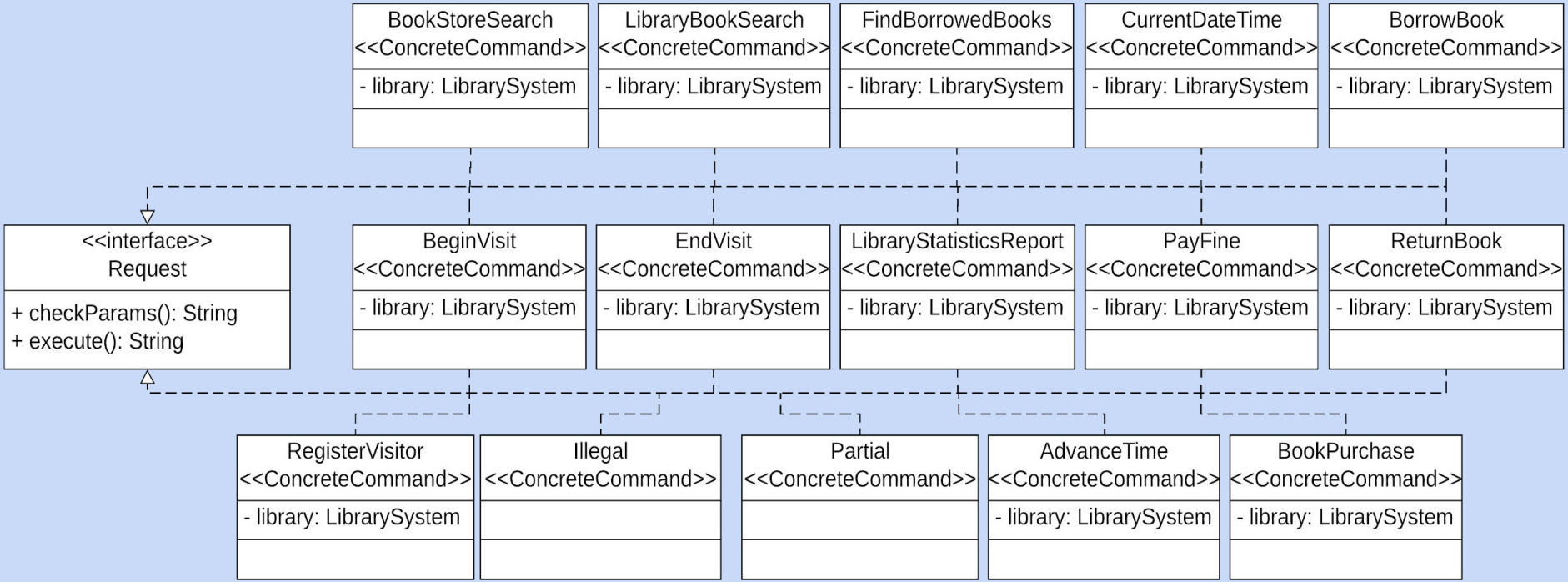

# Book Subsystem




# Visitor Subsystem




# Checkout Subsystem




# Request Subsystem

# Design Principles

- **Law Of Demeter:** Data must flow through the LibrarySystem class in a sequence of delegated method calls.
- **Composition over Inheritance:** The LibrarySystem is composed of a LibraryState of open or closed rather than having an inherited Open and Closed class.
- **Single Responsibility:** Each database deals with only one object's data.
- **Dependency Inversion / Open-Closed:** Interfaces are used for States and Commands to allow for easier maintenance and extension.

# Other Design Choices

- The TimeKeeper has two asynchronous constructs to track time. A Timer (Dynamic) updates a LocalDateTime (Static). This allows time to be “simulated” rather than be tied to actual system time.

- The states of the Databases are all stored as serialized objects in a file after the Library is shut down. The states can be restored by running the program with the file name as a command line argument

# State Design Pattern

The Library has two possible states: Open and Closed. Both states alter the way that the Library behaves relative to the rest of the system.

# MVC Design Pattern

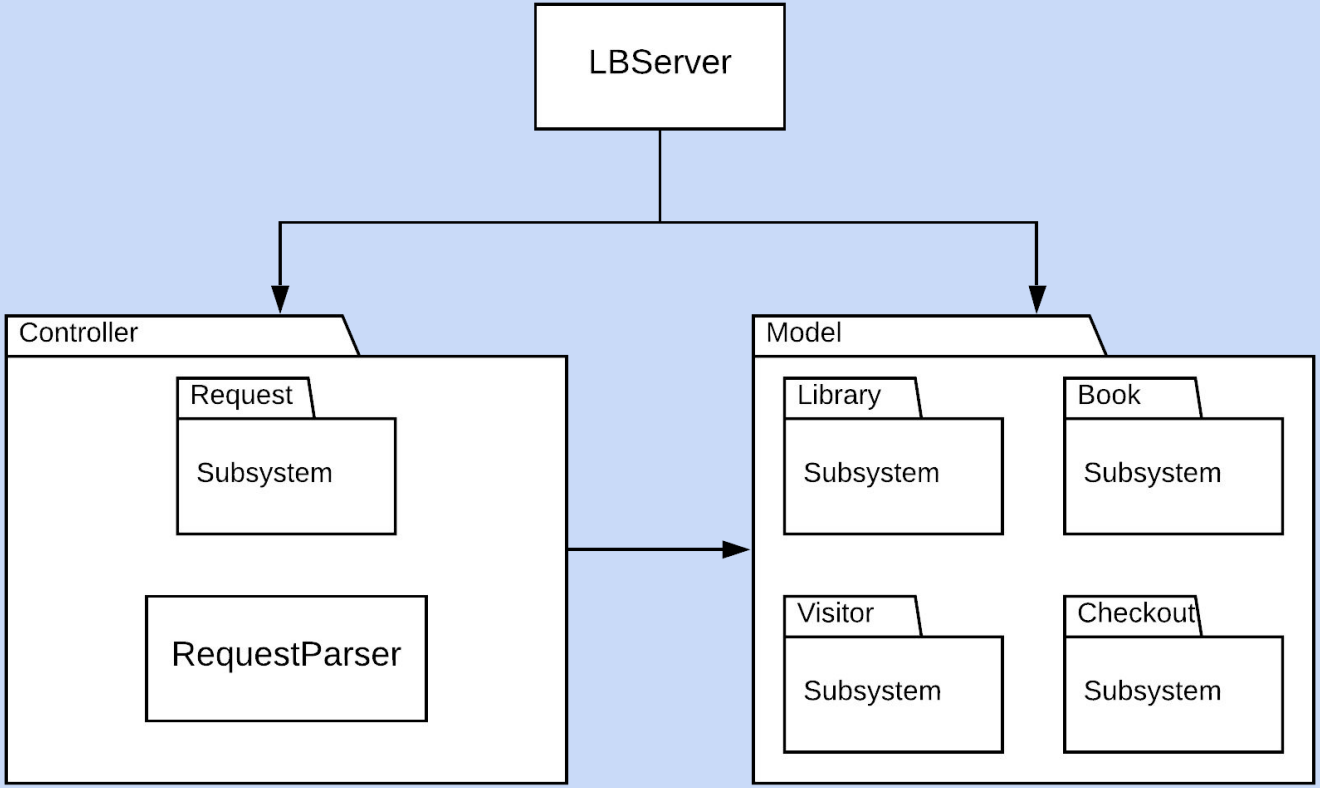
There is currently no graphic interface, but the implemented classes can be easily made compatible with one.

All logic associated with the storage of information is dealt with in the Model package.

The controller package includes all the Request related classes.

# Command Design Pattern

The Request interface defines the execute() method for all Request objects. Each object represents a different command made by the client.




# System Architecture

# Strategy Design Pattern

Each of the sorting filters when searching for books is a different algorithm to sort the books by a certain attribute.

# Difficulties Encountered

- Testing the system during the early stages of implementation
- Deciding how to perform request execution
- Realizing the request-response format page would dictate state and behavior
- Connecting the controller to the model classes without changing the initial design of the API

# Possible Improvements/Extra Features

Lava Flow: Some small methods and variables may be unused? □□♀

BookStores must be compatible with the BookData structure we created to be properly used by the Library to purchase books.

There are vague/similar names in our system that should be changed to be more specific (LibrarySystem, BookData/BookInfo).

GUI: Our code is compatible with a GUI, but does not contain one yet.

# Questions