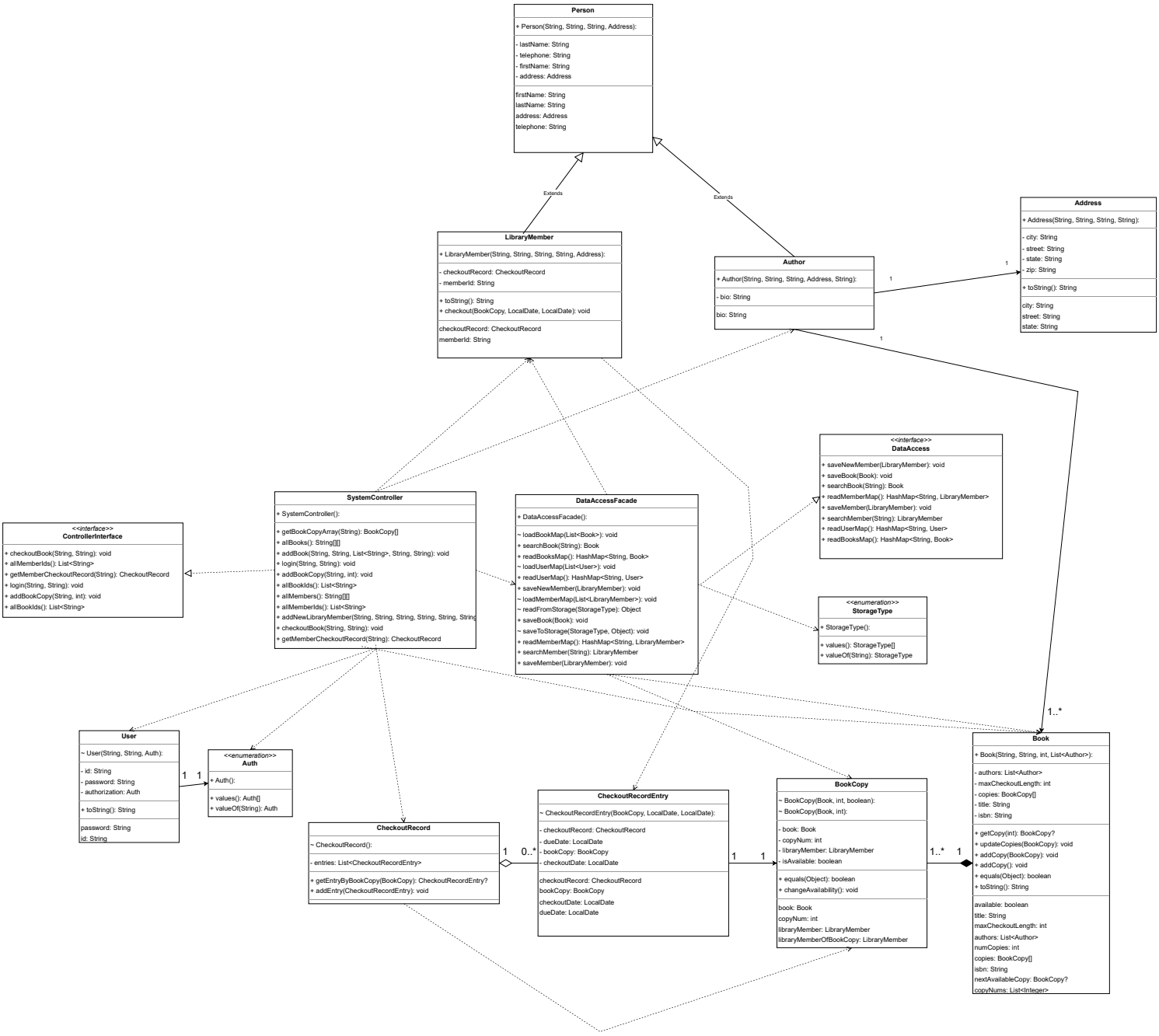
Person
+ Person(String, String, String, Address):
- lastName: String
- telephone: String
- firstName: String
- address: Address
firstName: String
lastName: String
address: Address
telephone: String
Extends
Extends
LibraryMember
+ LibraryMember(String, String, String, String, Address):
- checkoutRecord: CheckoutRecord
- memberId: String
+ toString(): String
+ checkout(BookCopy, LocalDate, LocalDate): void
checkoutRecord: CheckoutRecord
memberId: String
Author
+ Author(String, String, String, Address, String):
- bio: String
bio: String
1
1
Address
+ Address(String, String, String, String):
- city: String
- street: String
- state: String
- zip: String
+ toString(): String
city: String
street: String
state: String
1
<<interface>>
DataAccess
+ saveNewMember(LibraryMember): void
+ saveBook(Book): void
+ searchBook(String): Book
+ readMemberMap(): HashMap<String, LibraryMember>
+ saveMember(LibraryMember): void
+ searchMember(String): LibraryMember
+ readUserMap(): HashMap<String, User>
+ readBooksMap(): HashMap<String, Book>
SystemController
+ SystemController():
+ getBookCopyArray(String): BookCopy[]
+ allBooks(): String[][]
+ addBook(String, String, List<String>, String, String): void
+ login(String, String): void
+ addBookCopy(String, int): void
+ allBookIds(): List<String>
+ allMembers(): String[][]
+ allMemberIds(): List<String>
+ addNewLibraryMember(String, String, String, String, String, String
+ checkoutBook(String, String): void
+ getMemberCheckoutRecord(String): CheckoutRecord
DataAccessFacade
+ DataAccessFacade():
~ loadBookMap(List<Book>): void
+ searchBook(String): Book
+ readBooksMap(): HashMap<String, Book>
~ loadUserMap(List<User>): void
+ readUserMap(): HashMap<String, User>
+ saveNewMember(LibraryMember): void
~ loadMemberMap(List<LibraryMember>): void
~ readFromStorage(StorageType): Object
+ saveBook(Book): void
~ saveToStorage(StorageType, Object): void
+ readMemberMap(): HashMap<String, LibraryMember>
+ searchMember(String): LibraryMember
+ saveMember(LibraryMember): void
<<interface>>
ControllerInterface
+ checkoutBook(String, String): void
+ allMemberIds(): List<String>
+ getMemberCheckoutRecord(String): CheckoutRecord
+ login(String, String): void
+ addBookCopy(String, int): void
+ allBookIds(): List<String>
<<enumeration>>
StorageType
+ StorageType():
+ values(): StorageType[]
+ valueOf(String): StorageType
1..*
User
~ User(String, String, Auth):
- id: String
- password: String
- authorization: Auth
+ toString(): String
password: String
id: String
1
1
<<enumeration>>
Auth
+ Auth():
+ values(): Auth[]
+ valueOf(String): Auth
CheckoutRecord
~ CheckoutRecord():
- entries: List<CheckoutRecordEntry>
+ getEntryByBookCopy(BookCopy): CheckoutRecordEntry?
+ addEntry(CheckoutRecordEntry): void
1
0..*
CheckoutRecordEntry
~ CheckoutRecordEntry(BookCopy, LocalDate, LocalDate):
- checkoutRecord: CheckoutRecord
- dueDate: LocalDate
- bookCopy: BookCopy
- checkoutDate: LocalDate
checkoutRecord: CheckoutRecord
bookCopy: BookCopy
checkoutDate: LocalDate
dueDate: LocalDate
1
1
BookCopy
~ BookCopy(Book, int, boolean):
~ BookCopy(Book, int):
- book: Book
- copyNum: int
- libraryMember: LibraryMember
- isAvailable: boolean
+ equals(Object): boolean
+ changeAvailability(): void
book: Book
copyNum: int
libraryMember: LibraryMember
libraryMemberOfBookCopy: LibraryMember
1..*
1
Book
+ Book(String, String, int, List<Author>):
- authors: List<Author>
- maxCheckoutLength: int
- copies: BookCopy[]
- title: String
- isbn: String
+ getCopy(int): BookCopy?
+ updateCopies(BookCopy): void
+ addCopy(BookCopy): void
+ addCopy(): void
+ equals(Object): boolean
+ toString(): String
available: boolean
title: String
maxCheckoutLength: int
authors: List<Author>
numCopies: int
copies: BookCopy[]
isbn: String
nextAvailableCopy: BookCopy?
copyNums: List<Integer>