Book

-Book_Id: int
-Author: string
-ISBN: string
-Title: string

+Book():void {Default Constructor}
+Book(Book_Id: int, Author: string,
  ISBN: string, Title: string): void

+getID(): int
+getAuthor(): String
+getISBN(): String
+getTitle(): String
+getLoans(): Loan[*]
+setAuthor(Author: String): void
+setISBN(ISBN: String): void
+setTitle(Title: String): void
+setID(Book_Id: int): void

+toString(): string

Loan

-Loan_Id: int
-Copy_Id: int
-User_Id: int
-Date_Out: Date
-Date_Due: Date
-Active: Boolean = True

+Loan(): void
+Loan(Copy_Id: int, User_Id: int,
  Date_Out: Date, Date_Due: Date)

+Loan(Loan_ID: int, Copy_Id: int,
  User_Id: int, Date_Out: Date,
  Date_Due: Date)

+ getID(): int
+ getCopyID(): int
+ getUserID(): int
+ getDateOut(): Date
+ getDateDue(): Date
+ isActive(): Boolean

+ setID(Loan_Id: int): void
+ setCopyID(Copy_Id: int): void
+ setUserID(User_Id: int): void
+ setDateOut(Date_Out: Date): void
+ setDateDue(Date_Due: Date): void
+ setActive(bool: Boolean): void

+ toString(): String

+User_Address
«static»

-User_Id:int
-Address_Id: int
-Street1: string
-Street2: string
-City: string
-Zip: string {5 digit numeric}
-State: string {2 char abbv}

+User_Address():void
+User_Address(Street1: string, Street2: string,
  City: string, Zip: string, State: string): void

+ setStreet1(Street1: string): void
+ setStreet2(Street2: string): void
+ setCity(City: string): void
+ setZip(Zip: string): void
+ setState(State: string): void

+ getStreet1(): String
+ getStreet2(): String
+ getCity(): String
+ getZip(): String
+ getState(): String

+ toString(): String

User

-User_Id: int
-Name_First: string
-Name_Last: string
-Phone_Nos: string[*] {10 digit numeric}
-Type: string = "User" {User type}

+User():void
+User(Name_First: string, Name_Last: string):void

+ getPhoneNos(): string[*]
+ getName(): string[1]
+ getFirstName(): string[1]
+ getLastName(): string[1]

+ setFirstName(name: String): void
+ setLastName(name: String): void
+ setType(type: String): void

+ insertAddress(Address: User_Address): void
+ deleteAddress(Address_Id: int): void
+ insertPhone(Phone_No: string): void
+ deletePhone(Phone_Id: int): void

+ toString(): string
+ getType(): string

Patron

-Type: string = "Patron" {Patron type}
-Loans: Book.Loan[*]

+Patron(): void
+Patron(Name_First: string, Name_Last: string):void

Librarian

-Type: string = "Librarian" {Librarian type}

+Librarian(): void
+Librarian(Name_First: string, Name_Last: string):void

Copy

-Copy_Id: int
-Book_Id: int

+ Copy(): void
+ Copy(book: Book): void
+ Copy(Book_Id: int, Copy_Id: int): void

+ getID(): int
+ getBookID(): int

+ setID(Copy_Id: int): void
+ setBookID(Book_Id: int): void

+ toString(): String

DB_Access

- dbURL: string = "jdbc:sqlite:mydb.db"
- connection: java.sql.Connection

+ connect(): void throws SQLException
+ disconnect(): void throws SQLException
+ getConnection(): java.sql.Connection
+ existsDB(): boolean
+ createDB(): void throws SQLException

«abstract»
Table_Access<T>

# table_name: string
# connection:java.sql.Connection
# primary_key: string
- instances: Map<Class<? extends Table_Access<?»,
  Table_Access<?»

# Table_Access(): void
+ getInstance(connection: Connection):
  «T extends Table_Access<?»
+ getInstance(): «T extends Table_Access<?»

+ getTableName(): String
+ getPrimaryKey(): String
- getCallerClass(): Class<?>
# getColumnGetterMap(): Map<String, Method>
# getColumnSetterMap(): Map<String, Method>
# getTableSchema(): List<String>

# tableExists(): boolean throws SQLException
# createTable(): void throws SQLException

+ insert(record: T) throws SQLException
+ delete(Record_Id: int): void throws SQLException
+ delete(Record: T): void throws SQLException
+ read(Record_Id: int): T throws SQLException
+ readAll(): T[*] throws SQLException
+ update(record:T): void throws SQLException

+ find(Parameters: HashMap<String, String>):
  T[*] throws SQLException {Query DB for specified params}

User_Address_Access
{extends Table_Access<User.User_Address>}

# Table_Name: string = "UserAddress" {DB Table Name}
# primary_key: String = "UserAddressID"
- schema: List<String>
- columnGetterMap: Map<String, Method>
- columnSetterMap: Map<String, Method>

- Book_Access(): void

User_Access
{extends Table_Access<User>}

# Table_Name: string = "User" {DB Table Name}
# primary_key: String = "UserID"
- schema: List<String>
- columnGetterMap: Map<String, Method>
- columnSetterMap: Map<String, Method>

- Book_Access(): void

Book_Access
{extends Table_Access<Book>}

# Table_Name: string = "Book" {DB Table Name}
# primary_key: String = "BookID"
- schema: List<String>
- columnGetterMap: Map<String, Method>
- columnSetterMap: Map<String, Method>

- Book_Access(): void

Copy_Access
{extends Table_Access<Copy>}

# Table_Name: string = "Copy" {DB Table Name}
# primary_key: String = "CopyID"
- schema: List<String>
- columnGetterMap: Map<String, Method>
- columnSetterMap: Map<String, Method>

- Book_Access(): void

Loan_Access
{extends Table_Access<Loan>}

# Table_Name: string = "Loan" {DB Table Name}
# primary_key: String = "LoanID"
- schema: List<String>
- columnGetterMap: Map<String, Method>
- columnSetterMap: Map<String, Method>

- Book_Access(): void