Lab 12

Demo Points (50 Marks)

Stories 31-44 (5 Marks Each)

- JUnit test (2 Marks)

- Implementation (3 Marks)

Story 45 (35 marks)

- JUnit tests (15 marks)

- Implementation (20 marks)

This Week

This week you will add a Command Line Interface (CLI). CLI's were common in the early days of computing. They provided the most common way to interact with the computer before the Macintosh. Their are useful because they are easy to implement, the commands can be put together into programs, and computationally cheap to run. Their disadvantage is they impose too great a burden on human memory. Competition between software vendors has reduced to advantage of easy programing. The widespread use of computers by unsophisticated user has reduced the advantage of constructing programs from the commands. The dramatic fall in the cost of computing has reduced the advantage of computationally cheap programs. Therefore CLIs are now relatively rare. They are found most often in development environments.

The command line interface presents a prompt, then reads a command. It them executes the command and prompts for the next one. It terminates when the command "quit" is entered. It returns true when it executes a command, false, when "quit" is entered. Execute calls the appropriate methods to read, write, load, store information in the database.

User interfaces should be separated from the rest of the program because they are difficult to test automatically. Since they interface with humans, they require a user to fully test them. Many errors in a user interfaces, such as spelling or layout are difficult for a program to catch. They should be separated from the rest of the program so that errors in the user interface can be separated from errors in the rest of the program. Such separation ensures that errors found by the user can be fixed in the interface; they cannot indicate errors in the rest of the program.
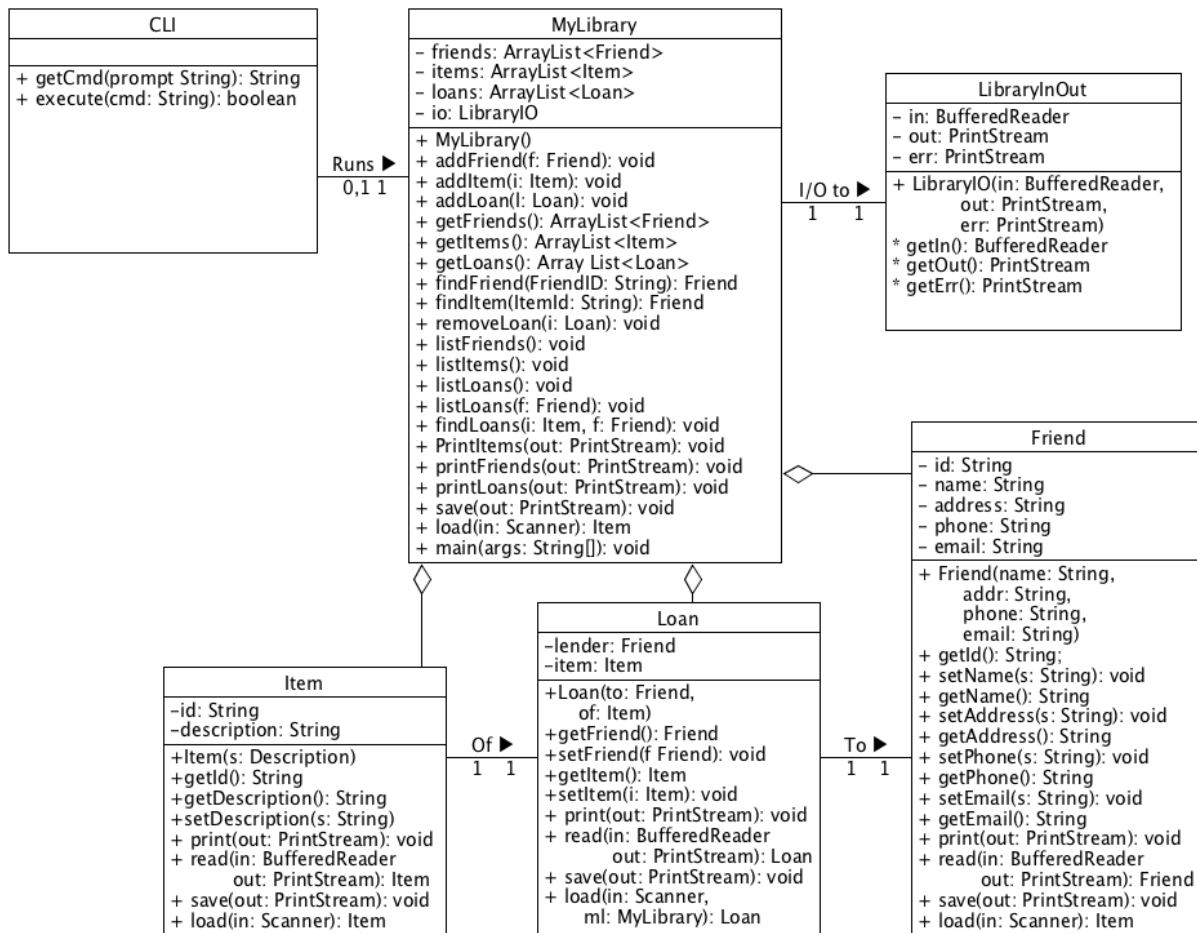
CLI

+ getCmd(prompt String): String
+ execute(cmd: String): boolean

MyLibrary

− friends: ArrayList<Friend>
− items: ArrayList<Item>
− loans: ArrayList<Loan>
− io: LibraryIO

+ MyLibrary()
+ addFriend(f: Friend): void
+ addItem(i: Item): void
+ addLoan(l: Loan): void
+ getFriends(): ArrayList<Friend>
+ getItems(): ArrayList<Item>
+ getLoans(): Array List<Loan>
+ findFriend(FriendID: String): Friend
+ findItem(ItemId: String): Friend
+ removeLoan(i: Loan): void
+ listFriends(): void
+ listItems(): void
+ listLoans(): void
+ listLoans(f: Friend): void
+ findLoans(i: Item, f: Friend): void
+ PrintItems(out: PrintStream): void
+ printFriends(out: PrintStream): void
+ printLoans(out: PrintStream): void
+ save(out: PrintStream): void
+ load(in: Scanner): Item
+ main(args: String[]): void

Runs ▶
0,1 1

LibraryInOut

− in: BufferedReader
− out: PrintStream
− err: PrintStream

+ LibraryIO(in: BufferedReader,
          out: PrintStream,
          err: PrintStream)
* getIn(): BufferedReader
* getOut(): PrintStream
* getErr(): PrintStream

I/O to ▶
1   1

Item

−id: String
−description: String

+Item(s: Description)
+getId(): String
+getDescription(): String
+setDescription(s: String)
+ print(out: PrintStream): void
+ read(in: BufferedReader
      out: PrintStream): Item
+ save(out: PrintStream): void
+ load(in: Scanner): Item

Of ▶
1   1

Loan

−lender: Friend
−item: Item

+Loan(to: Friend,
     of: Item)
+getFriend(): Friend
+setFriend(f Friend): void
+getItem(): Item
+setItem(i: Item): void
+ print(out: PrintStream): void
+ read(in: BufferedReader
      out: PrintStream): Loan
+ save(out: PrintStream): void
+ load(in: Scanner,
      ml: MyLibrary): Loan

To ▶
1   1

Friend

− id: String
− name: String
− address: String
− phone: String
− email: String

+ Friend(name: String,
        addr: String,
        phone: String,
        email: String)
+ getId(): String;
+ setName(s: String): void
+ getName(): String
+ setAddress(s: String): void
+ getAddress(): String
+ setPhone(s: String): void
+ getPhone(): String
+ setEmail(s: String): void
+ getEmail(): String
+ print(out: PrintStream): void
+ read(in: BufferedReader
      out: PrintStream): Friend
+ save(out: PrintStream): void
+ load(in: Scanner): Item

Illustration 1: Design with addition of CLI

In addition, We will add a main() method to the MyLibrary class so that it we can start it from the command line.

Next Week's Demo Points (40 marks)

Stories 46-55 (5 marks each)

- JUnit test (2 Marks)

- Implementation (3 Marks)